Proceedings BIOMA'2018
International Conference on Bioinspired Optimization Methods and their Applications

16-18 May 2018, Paris, France

Organization

Moroccan Society for Nature-Inspired Computing and Metaheuristic

SYNERGY Horizon 2020 GA No 692286

Institut des Systèmes Complexes

CRIStAL

Université de Lille

Jožef Stefan Institute Ljubljana, Slovenia

SYNERGY FOR SMART MULTI-OBJECTIVE OPTIMISATION

Table of contents

Collaborative Agent Teams (CAT): from the Paradigm to Implementation Guidelines

Marc-André Carle[1,2], Alain Martel[2], and Nicolas Zufferey[2,3]

[1] Département Opérations et Systèmes de Décisions, Université Laval, Québec, Canada, marc-andre.carle@osd.ulaval.ca
[2] CIRRELT (www.cirrelt.ca), Canada, alain.martel@cirrelt.ca
[3] GSEM – University of Geneva, Switzerland, n.zufferey@unige.ch

Abstract. We propose a general solution method framework based on a Collaborative Agent Teams (CAT) architecture to tackle large-scale mixed-integer optimization problems with complex structures. This framework introduces several conceptual improvements over previous agent teams' approaches. We discuss how to configure the three key components of a CAT solver for multidimensional optimization problems: the problem representation, the design of agents, and the information sharing mechanisms between agents. Implementation guidelines are also given.

Keywords: Multidimensional Optimization, Asynchronous Teams

1 Introduction

Despite of the continuous improvements of the commercial/academic solvers and of the exact and solution methods, many optimization problems are so complex that finding good-quality solutions remains challenging. These problems are too large (in terms of size) and complex (in terms of structure) to be solved directly through classical solution methods (e.g., [1, 2]). This paper extends the latter by generalizing and formulating the CAT (Collaborative Agent Teams) methodology for any optimization problem. We propose a general framework for CAT, an agent-based methodology based on the Asynchronous Teams (A-Teams) paradigm that is designed to tackle complex multi-dimensional optimization problems. We discuss how to design the three key components of a CAT solver: the problem representation; the design of the agents and their job description; the information sharing mechanisms between the agents.

"Decision problem" refers to a real-world issue requiring a solution as perceived by decision-makers. A decision problem can often be expressed qualitatively in terms of a choice between alternative options. An "optimization model" refers to a mathematical system formulated to represent a view of a decision problem. It is specified in terms of a set of decision variables and parameters. It incorporates objective function(s) and a constraint set. "Optimization algorithm" and "solution method" refer to programmable procedures developed to generate high-quality solutions for a given optimization model. The Simplex and

branch-and-bound methods, greedy heuristics and tabu search metaheuristics, are all examples of optimization algorithms, including simulation-optimization approaches [3]. A "heuristic" is a basic solution method that finds, in a reasonable amount of time, a "satisfying" solution to the considered optimization model. Optimality is generally not guaranteed. A "metaheuristic" (MH) is a higher-level methodology that guides underlying heuristics for solving the optimization problem. The term "solution" is used to designate a set of values for the decision variables that satisfy all the constraints of a given optimization model.

The paper is organized as follows. Section 2 presents a literature review of operations research approaches that contributed to design CAT. In Section 3, CAT and its components are presented. Section 4 concludes the paper.

2 Strategies to Tackle Complex Problems/Models

We describe here algorithmic strategies to solve complex decision problems and the optimization models used to represent them. We also position their relative strengths in achieving better performance or tackling more complex problems. A general description of relevant strategies is provided rather than a technical description of algorithms. Some strategies are not exclusive: they can be hybridized to tackle the most challenging problems (which opens the door for CAT).

2.1 Classical Approaches, Parallel Algorithms and Hybridization

Many optimization models are today "easy" to solve, even with the use of a solver (CPLEX/Gurobi) or with the help of filtering techniques to reduce the search space [4, 5]. However, several optimization models are hard to solve using solvers, especially when the model is nonlinear/stochastic. Models with a single category of binary decision variables and few constraint types are often solved to near-optimality in a reasonable amount of time with MHs (e.g., the tabu search MH is efficient for the simple facility location model [6]). MHs can usually successfully tackle such problems even if they have nonlinear objective functions or constraints. Local-search MHs are efficient on these problems because it is straightforward to create a new solution by applying a local transformation on a given solution. When multiple types of integer, binary and continuous decision variables are present in the model, these approaches may not be effective.

Most state-of-the-art commercial solvers (CPLEX, Gurobi) use several processors at a time if possible. According to [7], parallelism [in MHs] allows for improved solution quality and reduction in the resolution time. Two strategies are especially relevant: (1) parallelization of the search: several copies of a given MH work in parallel, each having its own parameter settings and possibly exchanging solutions synchronously or asynchronously during the search process; (2) parallelization of some of the most computationally-intensive tasks of the search process (typically solution quality evaluation or neighborhood exploration).

Hybridization refers to the combination of different types of algorithms into one methodology [8]. The first main technique combines various MHs [9], hoping

that one method's strengths compensate for the other method's weaknesses. Several hybridization strategies can be designed for any two given MHs, resulting in a larger number of potential solution methods. The second main hybridization technique combines MHs with exact methods [10]. These hybrid solution methods are often called matheuristics [11]. They effectively combine the ability of MHs to handle a large number of binary/integer decision variables, with the LP- (linear program) or MIP- (mixed integer program) based methods' ability to handle a large number of constraints and continuous decision variables.

2.2 Decomposition and Model-Based Strategies

Some methods use the optimization model's formulation to break it down into smaller/easier problems. Since the 1960s, decomposition-based solution methods (e.g., [12, 13]) are effective at solving large optimization models that exhibit a specific model structure. The efficiency of these methods lies in clever reformulation of the optimization model and the availability of a sub-model that can be solved very fast. However, when the decomposed sub-models they yield are themselves difficult to solve, these methods may not perform well.

Multilevel techniques [14] are another family of methods making use of the model formulation. They start from the optimization model, then iteratively and recursively generate a smaller and smaller model by coarsening until a relatively small model is obtained, creating a hierarchy of optimization models [9]. A solution to the smallest model is found by some optimization algorithm. Then, the solution to this problem is successively transformed into a solution to the model of the next level until a solution to the original optimization model is found.

Recently, a number of progressive variable fixing solution methods have been proposed to solve complex models. The simplest method, the LP-rounding strategy [15], uses a solver to obtain the LP relaxation of the model. The values of integer and binary decision variables that are fractional in the LP relaxation are then rounded to obtain an integer-feasible solution. In [16], a sequence of linear relaxations of the original optimization model is solved, and as many binary and integer variables as possible are fixed at every iteration. These methods are effective to solve problems with a small number of binary and a large number of continuous decision variables.

2.3 Distributed Decision Making and Agent Optimization

Another strategy to cope with model complexity is to work at the decision problem level rather than directly on the optimization model. The decision problem can often be partitioned into various interconnected sub-problems under the responsibility of distinct organizational units. For each sub-problem, a sub-model is formulated and solved using an optimization algorithm. This approach has many advantages and is even more suited to decision problems involving multiple decision-makers. According to [17], "distributed decision making can be useful in order to better understand/manipulate a complex decision situation".

Multi-agent systems (MAS) and agent-based optimization algorithms have been used recently to model/analyze complex decision problems. MAS formalize complex decision problems as networks of simpler decision problems, each of these problems being tackled by a separate agent [17]. Depending on the degree of sophistication of the approach, the agent may use basic rules to make decisions, or formulate an optimization model which is then solved with an appropriate (exact or heuristic) optimization algorithm. An example of this approach is A-Teams [18], a cooperative MAS used for various problems (e.g. [1, 19–21]).

2.4 Towards an Integrated Optimization Framework

Many approaches have integrated the above strategies to solve complex optimization models. Their strengths are often complementary: a MAS is indeed well suited to implement parallel and potentially hybrid optimization algorithms. A hybrid matheuristic can couple two algorithms working in parallel rather than sequentially. Despite these advantages, very few tools have been proposed to combine the strengths from all these strategies into one solution system. The five following elements should be present in an optimization framework designed for complex decision problems. (1) Draw inspiration from the decision problem and alternative optimization model formulations to design adapted solution methods, instead of one perspective. (2) Use partitioning strategies through organizational decomposition (at the problem level) or mathematical decomposition (at the model level), while working on each partition simultaneously in parallel. (3) Use the type of optimization algorithm that works best for each sub-model. (4) Share information/solutions between different optimization strategies. (5) Combine good solutions from sub-models into good solutions to the complete model. An optimization framework based on these characteristics is now proposed.

3 CAT as an Agent-Based Solution Method

CAT is a hybrid distributed agent-based solution method to solve complex decision problems and associated optimization models that cannot be efficiently addressed using classical MHs or mathematical decomposition methods. The approach builds on the A-Teams paradigm [18], and it relies on the foundations of Subsection 2.4. We use the location-routing problem (LRP) [22] to illustrate the CAT concepts. It involves decisions on the number and location of distribution centers (DCs) in order to serve customers at minimum cost, and finding the best delivery schedules and vehicle routes to serve the customers. "An asynchronous team is a team of software agents that cooperate to solve a problem by dynamically evolving a shared population of solutions" [18]. Agents are autonomous: they incorporate their own representation of the problem, and rules to choose when to work, what to work on and when to stop working. The approach is well suited to implement multiple representations of a problem, such as advocated above. Previous work suggests that A-Teams can host a large variety of optimization algorithms. Whereas some applications [1] use simple heuristics

and linear and integer programming, recent applications [20, 21] employ MHs (e.g., tabu search). When facing complex optimization models, it makes sense to use the best tools for each (sub-)model. A MAS allows for that much flexibility.

3.1 Problem Solving Approach

The following steps are required to solve optimization problems with CAT: (1) identify different relevant points of view (dimensions) to examine the decision problem; (2) formulate optimization models and sub-models for these dimensional views; (3) design optimization algorithms to solve each sub-model; (4) design optimization algorithms to integrate solutions from sub-models into solutions of the complete optimization models. These steps are explained below.

Views and Sub-Problems, Models and Sub-Models. Complex decision problems can be analyzed from different views, that is, a filter/lens which emphasizes, reduces or reshapes some aspects of the decision problem to solve. It can reflect a stakeholder's perceptions. The integrated view refers to a holistic apprehension of the complete decision problem, that is, one that looks at all relevant facets from a centralized standpoint. Problem solving with CAT requires addressing the problem with an integrated view, and with alternative dimensional views. Dimensional views are rearrangements of the problem into systems of interrelated sub-problems. These sub-problems may cover only a subset of the objectives and decisions of the original problem and they may involve a reduction of some of its facets. Dimensional views are used to reduce the complexity of the problem by providing effective partitioning schemes. Dimensional views must be selected before optimization models can be formulated. A dimensional view may require the definition of several sub-problems. A sub-problem contains a portion of the decisions and context associated with the decision problem. The number of sub-problems used and the exact definition of each of them are critical issues. Useful sub-problems possess the following characteristics. First, they make sense from a business standpoint (i.e., they are easily understandable by a decision-maker). Second, the set of all the sub-problems associated with a dimensional view must constitute a valid representation of the complete decision problem. For the LRP, the following two dimensional views could be defined. First, a functional view is associated with the types of decisions (location, customer allocation to facilities, vehicle routing) associated with the decision problem. The problem can then be partitioned into a DC location sub-problem, a customer-to-DC allocation sub-problem, and a transportation or route design sub-problem. Second, the LRP has an inherent spatial dimension. Indeed, the customers served by a company may cover a large territory, and logistics decisions may be made on a national or sales region level instead of globally. In this context, the problem can be partitioned into several regional sub-problems. These dimensional views and the associated sub-problems are easily understandable by a decision-maker. Each regional sub-problem contains all decision types, and each functional sub-problem contains decisions for all regions. Thus, they both constitute a valid representation of the whole problem.

The integrated view leads to the formulation of a "complete" optimization model to represent the decision problem. This model is generally difficult to solve, but it will be used for various purposes. For each dimensional view, sub-models are formulated to represent sub-problems. These formulations are usually expressed in terms of partitions of complete model decision variable vectors and parameter matrices. They may also be based on alternative modeling formalisms: (e.g., a constraint programming sub-model can be defined even if the complete optimization model is a MIP). A sub-model is useful if it can be solved efficiently. It is usually the case if the sub-model: (1) corresponds to a generic class of decision models studied in depth in the literature (e.g., bin packing, facility location); (2) can be solved to optimality using generic LP-MIP solvers, or dynamic programming, or simple enumeration (explicit or implicit); (3) isolates a homogeneous group of binary/integer variables and their associated constraints.

Optimization Algorithms. Once the sub-models have been formulated, optimization algorithms must be designed to solve them. In CAT, optimization algorithms are implemented as a set of autonomous software agents. Solutions to sub-models are recorded and subsequently used to build complete solutions. The following guidelines are useful to select a solution method. (1) Develop greedy heuristics to construct feasible solutions for profit maximizing or cost minimizing sub-models. (2) When the sub-model has been studied in the literature, published solution methods can be integrated into CAT. (3) Purely linear sub-models can be solved using a LP-solver library. (4) Sub-models involving a homogeneous group of binary/integer variables can usually be solved effectively with a local search MH since it is rather straightforward to define a neighborhood in this context. In the LRP context, some of the sub-models formulated and the solution methods selected could be the following: a pure facility location sub-model solved with a MIP solver such as CPLEX; a location-allocation sub-model solved with a Lagrangean heuristic [23]; a vehicle routing sub-model solved with a tabu search MH [24]; a regional LRP sub-model solved with a tabu search [25].

Integration Sub-models. Integration refers to combining the solutions of the sub-models associated with one view into solutions to the full optimization model. It is done by exactly/heuristically solving an integration sub-model. Integration sub-models are restricted versions of the full optimization model obtained by fixing the value of several decision variables. The fixed values are provided by the solutions to the dimensional sub-models. By solving the integration sub-model, the optimal value of the non-fixed decision variables is found, and a solution to the complete model is produced. We refer to the set of decision variables to optimize in an integration sub-model as integration variables. Integration variables not present in any dimensional sub-model are linking variables, and those present in various dimensional sub-models are overlapping variables.

Integration is also used as a search strategy. For a specific dimensional view, the choices of integration variables lead to different integration sub-models. When the dimensional sub-models solutions are mutually exclusive, the integration sub-model contains only linking variables, and optimizing these variables provides a feasible solution for the complete model. When the dimensional sub-

models solutions are overlapping, a merging integration sub-model is obtained. Since it is unlikely that the overlapping variables have the same value in all partial solutions, the integration sub-model must find the optimal value of these variables. The search space created by a merging sub-model can be enhanced by including more than one partial solution from a given dimensional sub-model. This adds all the variables from that sub-model to the set of overlapping variables. If the resulting integration sub-model is difficult to solve, one can constrain the integration sub-model by fixing the values of the overlapping variables that are identical in all partial solutions or restricting the values of the overlapping variables to those found in the partial solutions, resulting in a smaller model.

Depending on the partial solutions chosen for integration, the resulting sub-model may be infeasible. When it occurs, an alternative integration sub-model that seeks to find a feasible solution while keeping most of the partial solutions' characteristics is used. In these sub-model's, the original objective function is replaced with the minimization of the number (or amplitude) of decision variable changes when compared with the values found in the sub-problems.

To conclude our LRP example, using the pure location sub-model and the vehicle routing sub-model solutions, one would formulate a merging integration sub-model as follows. The depot location decision variables are fixed using the solution to the pure min-cost location sub-model. Several vehicle routing sub-model solutions are also considered. The resulting integration sub-model selects a set of feasible routes among the routes provided by the vehicle routing sub-models. It is a capacitated set partitioning model for which many methods exist.

3.2 CAT System Structure

The structure of the CAT system incorporates a blackboard, utility agents and optimization agents. The blackboard acts as a memory and a hub for communications, and it is the repository of all solutions (to the complete optimization model and to all sub-models). Agents communicate solely through the blackboard. New complete or partial solutions are placed on the blackboard and existing solutions are retrieved when necessary. Utility agents provide functionalities required by all agents, such as building mathematical model files for solvers, formatting instance data, and compiling solution statistics. The optimization agents are the most important: (1) construction agents create new solutions from scratch; (2) improvement agents take existing solutions and try to improve them; (3) destruction agents remove unwanted solutions from the repository; (4) integration agents combine high-quality solutions from various dimensional sub-models into solutions to the complete optimization model. These agent roles are now defined.

3.3 Agent Jobs Descriptions

As pointed out by earlier works [20, 26], a few key questions must be answered when designing a multi-agent optimization system. How many agents should be used? What should their role be? How should they decide when to act, what to act on, and how to act? For all their advantages, agent teams are complex

to design and implement. Indeed, if the system uses several algorithms that are similar in nature (e.g., simulated annealing variants) on the same sub-model, it is likely that one of the optimization algorithms (usually the best) will be largely responsible for the team's performance. Also, on a computer with limited resources (memory or processor power), it is likely that adding agents will deteriorate performances. To avoid these pitfalls, it is advised in [18] to start with a small number of agents, and to add new agents with different skills as needed. According to the literature and to our experience in developing CAT systems, an agent team needs four important basic skills. (1) Quickly obtain feasible solutions to the complete optimization model. Although these may not be of high quality, they provide a basis for other agents to work upon. (2) Improve existing solutions. This can be done at the complete model level or agents can work on specific parts of the problem. (3) Remove unwanted or poor solutions from the population to control its size. (4) Efficiently combine features from solutions originating from different methods or dimensions. The nature of these skills is now discussed.

Construction agents. Feasible solutions can be obtained quickly with simple heuristics (e.g., greedy or hill-climbing algorithms, or even randomly) for several classes of optimization models. Another option is to use generic LP/MIP heuristics (e.g., feasibility pump [27]). This approach tends to produce solutions that are very different from those obtained with greedy methods. The key goals at this task are speed and diversity, rather than solution quality. Using various methods usually results in a more diverse initial population of solutions, yielding a higher potential for improvement and collaboration, and reducing the need for specific diversification strategies. If the complete optimization model is difficult to solve but it is easy to find a feasible solution, one can generate solutions to the complete model then infer initial solutions for sub-models from these solutions, thus reducing the number of algorithms and agents needed for this role.

Improvement agents. For complex decision problems, it is recommended to work on sub-models and not on the complete model. Since defining a neighborhood (or a set of neighborhoods covering the complete model's range of variables) may be challenging, local search is difficult to use. Evolutionary computing provides generic crossover operators, but solution encoding is complex and on highly constrained problems, developing effective repair functions may be problematic. To design a good set of improvement agents, the solution methods used to solve sub-models must be carefully selected. If the sub-model is a LP, existing LP-solvers can be used. If it has only one type of binary/integer variable (allowing for the construction of neighborhoods), a local search MH can be developed. If it is a variant of a well-known problem, the best available method can be implemented. It may also be worthy to investigate alternative sub-model reformulations. Many strategies can tackle complex sub-models (e.g., an initial solution obtained with a simple heuristic may give a hot-start for a MIP-solver). Nowadays, commercial solvers incorporate several generic MIP heuristics [28]. When MHs are not efficient, generic MIP heuristics often are. In [29], a review of heuristics based on mathematical programming is provided. To ensure that

the system continuously works on each sub-model (or, at least, looks for opportunities to work on it), a dedicated agent should be assigned to its solution. The creation of "super-agents" performing several tasks should be avoided. Such super-agents tend to use too much resource and require complex scheduling rules.

Destruction agents. Solution destruction is as important as solution creation in agent-based optimization [18]. In some situations, the choice of solutions to destroy is obvious, such as when duplicates exist in the population. Aside from maintaining some control on the size of the population, destruction serves two purposes: removing poor quality solutions and maintaining diversity in terms of solution characteristics. At the beginning of the search, the solutions in the population are quite diverse. As improvement agents work, the solution quality of the best solutions in the population improves rapidly. At this stage, the destruction agent should focus on removing solutions that are of poor quality. A simple rule such as choosing a solution at random from those in the 4th quartile in terms of solution quality is appropriate. However, as the overall quality of solution improves, newly created solutions tend not to be competitive in terms of quality compared to those which have been improved by several agents. They should have a chance to be improved before they are discarded. Furthermore, as the population improves, working on the same solutions tends to accelerate convergence. As the search progresses, a destruction agent shifts its focus from removing poor solutions to either: (1) removing a random solution which has been improved at least (I - 2) times and is in the bottom half in terms of performance, where I (parameter) is the number of improvements made on the solution that has been improved most frequently in the population; (2) finding the two most similar solutions in the population, and destroying the worst one; (3) finding the solution which has been used the most frequently to create new solutions among the solutions in the 4th quartile in terms quality, and destroying it. These rules can be encapsulated in destruction agents, and they work equally well on a population of complete solutions or on a population of partial solutions (solutions to a specific sub-problem). The metrics necessary to implement them are detailed in Subsection 3.4. Alternatively, some solutions can be "protected" and be immune to deletion for a certain amount of time. These solutions may be the status quo or solutions provided by a decision-maker.

Integration agents. Although the destruction agent works toward maintaining variety, additional diversification strategies may be needed. It is possible to add an agent whose sole objective is to provide the population with radically different solutions than those currently in the population. This agent should maintain a record of what has been proposed in the past, so it does not produce solutions similar to those already removed from the population due to poor solution quality. The integration of partial solutions from sub-models into complete solutions is a key component of an efficient agent team. At least one optimization algorithm should be provided for each integration sub-model. If two methods are available, they can both be used if they generate different high quality solutions. The number of agents to use depends on the relative speed at which the improvement agents generate new solutions to sub-models and the amount of

computation effort required to solve the integration sub-models. Integration can be used in flexible ways. Integration of solutions to sub-models from different dimensional views can be desirable, as long as the resulting merging integration sub-models are not too difficult to solve. Solving these models often requires the design of specific heuristics or the use of a generic approach (see above). These heuristics are easily implemented using a MIP solver such as CPLEX or Gurobi. This approach is in line with scatter search and path-relinking MHs, and is an effective way of reaping the most benefits from using multiple dimensional views. As this type of integration is slightly different from the type of integration sub-models required to assemble complete solutions from partial solutions, these sub-models should be assigned to a different integration agent.

3.4 Decision Rules and Metrics: Solution Ancestry and Similarity

An agent needs formal rules to determine which solution to work on. A trivial option is to select a random solution from the population, but this does not give very good results. Obviously, an agent does not want to select a solution that it has recently worked on. A simple yet effective decision rule is that the agent waits that at least three others agents have improved the solution before attempting to work on it again. Some improvement agents such as local search MHs may want to push that rule a little further: since a local search explores thoroughly a restricted portion of the search space, an agent may want to select a solution that is significantly different from the one it just worked on. For more sophisticated decision rules, metrics can be computed as follows. Agents need an effective way to determine which solutions they recently worked on. In a cooperative context, this information should be accessible to all agents. A simple metric to achieve this objective is solution's ancestry. Simply put, a solution's ancestry is its genealogical tree. Each solution keeps track of the solutions used for its creation, or as a basis for its improvement, and the agents that worked on it. An improvement agent can then use this information to determine if it has worked on a solution recently, or on any of its parents. Tied to each solution is a list of agents that have worked on it, and whether this attempt at improving it succeeded. This list is sorted in reverse order. A similar mechanism is used to determine whether a solution has transmitted its characteristics to other solutions in the population. Anytime a solution is used to create a new solution or to alter an existing solution, its characteristics are propagated through the population. The new solution is linked to its parent solution(s) through an acyclic directed graph structure, so that it is easy to find all the parents or all offspring of a given solution. A propagation index is calculated for each solution, which is set to 0 when the solution is created. When a new solution s_0 is created, if it has one or more parent solutions, it parses its solutions digraph and updates the values of its parents' propagation index in a recursive manner.

There are occasions when an agent wishes to find similar, or very different, solutions in the population. A well-known metric to do this is the Hamming distance, which is the number of binary variables with different values in two

solutions. Although it is useful in some contexts, that measure can be misleading for mixed-integer linear models. In most decision problems, some decisions have more importance than others. Often, a group of binary or integer variables is larger but of less significance. In the LRP, many more decision variables are associated with the vehicle routing decisions than the location decisions, despite the fact that location decisions have a more lasting impact on the quality of the solution. For this problem, two solutions could have the exact same depot locations but have a high Hamming distance, which would not reflect the importance of location decisions adequately. In order to obtain a more accurate distance metric, one can measure the percentage of variables of each type that have the same value. Different types of variables can even be weighted in order to account for their relative importance.

4 Conclusion

This paper gives a generic methodology and implementation guidelines to model and solve complex real-world decision problems. It shows how to look at decision problems from different point-of-views, and how to partition the problem as well as the associated optimization method into dimensional sub-models. We propose a general formulation of CAT, a new agent-based solution method designed to benefit from the complexity reductions resulting from the multi-dimensional views of the problem. CAT is scalable since its execution can easily be distributed over multiple computers. CAT is easily extendable by adding new agents or processing power as needed or by allowing some of the agents to work using more than one processor at a time.

References

1. Murthy, S., Akkiraju, R., Goodwin, R., Keskinocak, P., Rachlin, J., Wu, F.: Cooperative Multiobjective Decision Support for the Paper Industry. Interfaces **29 (5)** (1999) 5 – 30
2. Carle, M.A., Martel, A., Zufferey, N.: The CAT metaheuristic for the solution of multi-period activity-based supply chain network design problems. International Journal of Production Economics **139 (2)** (2012) 664 – 677
3. Silver, E., Zufferey, N.: Inventory control of an item with a probabilistic replenishment lead time and a known supplier shutdown period. International Journal of Production Research **49** (2011) 923–947
4. Hertz, A., Schindl, D., Zufferey, N.: Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. 4OR **3 (2)** (2005) 139 – 161
5. Zufferey, N.: Heuristiques pour les Problèmes de la Coloration des Sommets d'un Graphe et d'Affectation de Fréquences avec Polarités. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland (2002)
6. Michel, L., Hentenryck, P.V.: A simple tabu search for warehouse location. European Journal of Operational Research **157** (2004) 576 – 591

7. Melab, N., Talbi, E.G., Cahon, S., Alba, E., Luque, G.: Parallel Metaheuristics: Models and Frameworks. In: Parallel Combinatorial Optimization. Hoboken: Wiley (2006) 330
8. Talbi, E.G.: A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics **8 (5)** (2002) 541–564
9. Blum, C., Puchinger, J., Raidl, G., Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing **11** (2011) 4135 – 4151
10. Raidl, G.R., Puchinger, J.: Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization. In: Hybrid Metaheuristics. Springer (2008) 31 – 62
11. Maniezzo, V., Sttzle, T., Voss, S.: Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. Springer (2009)
12. Dantzig, G.B., Wolfe, P.: Decomposition Principle for Linear Programs. Operations Research **8** (1960) 101 – 111
13. Benders, J.F.: Partitioning Procedures for Solving Mixed Variables Programming Problem. Numerische Mathematik **4** (1962) 238 – 252
14. Walshaw, C.: Multilevel Refinement for Combinatorial Optimization Problems. Annals of Operations Research **131** (2004) 325 – 372
15. Melo, M., Nickel, S., da Gama, F.S.: An efficient heuristic approach for a multi-period logistics network redesign problem. TOP **22** (2014) 80 – 108
16. Thanh, P., Péton, O., Bostel, N.: A linear relaxation-based heuristic approach for logistics network design. Computers & Industrial Engineering **59** (2010) 964 – 975
17. Schneeweiss, C.: Distributed Decision Making. Springer (2nd Edition). (2003)
18. Talukdar, S., Murthy, S., Akkiraju, R.: Asynchronous Teams. In: Handbook of Metaheuristics. Kluwer Academic Publishers
19. Keskinocak, P., Wu, F., Goodwin, R., Murthy, S., Akkiraju, R., Kumaran, S.: Scheduling solutions for the paper industry. Operations Research **50 (2)** (2002) 249 – 259
20. Aydin, M., Fogarty, T.: Teams of autonomous agents for job-shop scheduling problems: An experimental study. Journal of Intelligent Manufacturing **15** (2004) 455 – 462
21. Ratajczak-Ropel, E.: Experimental Evaluation of the A-Team Solving Instances of the RCPSP/max Problem. Lecture Notes in Computer Science **6071** (2010) 210 – 219
22. Nagy, G., Salhi, S.: Location-routing: issues, models and methods. European Journal of Operational Research **177** (2007) 649 – 672
23. Beasley, J.E.: Lagrangean heuristics for location problems. European Journal of Operational Research **65** (1993) 383 – 399
24. Cordeau, J.F., Laporte, G.: Tabu search heuristics for the vehicle routing problem. In: Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search. Springer (2005) 145 – 163
25. Wu, T.H., Low, C., Bai, J.W.: Heuristic solutions to multi-depot location-routing problems. Computers & Operations Research **29** (2002) 1393 – 1415
26. Zufferey, N.: Optimization by ant algorithms: Possible roles for an individual ant. Optimization Letters **6 (5)** (2012) 963 – 973
27. Achterberg, T., Berthold, T.: Improving the Feasibility Pump. Discrete Optimization **4** (2007) 77 – 86
28. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, Series A **102** (2005) 71–90
29. Ball, M.: Heuristics based on mathematical programming. Surveys in Operations Research and Management Science **16** (2011) 21 – 38

Collaborative Variable Neighborhood Search

Nicolas Zufferey[1] and Olivier Gallay[2]

[1] Geneva School of Economics and Management, GSEM - University of Geneva
Blvd du Pont-d'Arve 40, 1211 Geneva 4, Switzerland, n.zufferey@unige.ch
[2] Faculty of Business and Economics (HEC Lausanne), University of Lausanne
Quartier UNIL-Dorigny, 1015 Lausanne, Switzerland, olivier.gallay@unil.ch

Abstract. Variable neighborhood search (VNS) is a well-known metaheuristic. Two main ingredients are needed for its design: a collection $M = (N_1, \ldots, N_r)$ of neighborhood structures and a local search LS (often using its own single neighborhood L). M has a diversification purpose (search for unexplored zones of the solution space S), whereas LS plays an intensification role (focus on the most promising parts of S). Usually, the used set M of neighborhood structures relies on the same type of modification (e.g., change the value of i components of the decision variable vector, where i is a parameter) and they are built in a nested way (i.e., N_i is included in N_{i+1}). The more difficult it is to escape from the currently explored zone of S, the larger is i, and the more capability has the search process to visit regions of S which are distant (in terms of solution structure) from the incumbent solution. M is usually designed independently from L. In this paper, we depart from this classical VNS framework and discuss an extension, Collaborative Variable Neighborhood Search (CVNS), where the design of M and L is performed in a collaborative fashion (in contrast with nested and independent), and can rely on various and complementary types of modifications (in contrast with a common type with different amplitudes).

Keywords: Metaheuristics; Variable Neighborhood Search.

1 Introduction

As depicted in [1], modern methods for solving complex optimization problems are often divided into exact methods (e.g., dynamic programming, branch and bound) and metaheuristics [2]. An optimal solution can always be found with an exact method in a finite amount of time. Unfortunately, most real-life optimization problems are NP-hard, and therefore, exact methods would require too much computing time to find an optimal solution. For such difficult problems, it is thus better to quickly find a satisfying solution. A streamline heuristic can be used if solution quality is not a crucial issue. Otherwise, a more advanced metaheuristic is recommended. There are mainly two classes of metaheuristics: local search and population based methods. The former algorithms work iteratively on a single solution (e.g., descent local search, tabu search, variable neighborhood search), whereas the latter manage a set of solutions (e.g., genetic algorithms, ant colonies, adaptive memory algorithms).

A local search starts from an initial solution. Next, in each iteration, a *neighbor* solution s' is generated from the *current* solution s by performing a *move* on s (i.e., the structure of s is slightly modified to get s', according to predefined rules). In tabu search, to try to avoid cycling (i.e., coming back to an already visited solution), a *tabu list* forbids to perform the reverse of recently performed moves. The best non-tabu move is generally performed in each iteration. In most local search algorithms, only one neighborhood structure is used (i.e., a solution can only be modified according to a dedicated technique with a fixed amplitude, like changing one component of the solution). In contrast, Variable Neighborhood Search (VNS) [3] uses sequentially different neighborhood structures. A generic version of VNS is given in Algorithm 1, where $N_1, N_2, \ldots, N_r$ denote a finite set of neighborhoods, $N_i(s)$ is the set of solutions in the i^{th} neighborhood of solution s, and L is the neighborhood structure used in the local search LS. In a classical VNS, the neighborhood structures $N_1, \ldots, N_r$ actually rely on the same type of move, but used with different amplitudes. For example, if a solution s is a vector, N_i consists in changing the value of i components of s. The resulting collection M of neighborhood structures are thus dependent (i.e., they rely on the same type of modification) and nested (i.e., N_i is included in N_{i+1}).

Algorithm 1 Variable Neighborhood Search (VNS)

Generate an initial solution s and set $i = 1$

While no stopping criterion is met, **do**

1. *Shaking* (diversification): generate a neighbor solution s' in $N_i(s)$.
2. *Local search* (intensification): apply some local search method (with neighborhood L) with s' as initial solution, and let s'' be the returned solution.
3. *Relocate the search*: if s'' improves s, move there (set $s = s''$), and continue the search with N_1 (set $i = 1$); otherwise set $i = i + 1$, but if $i > r$, set $i = r$.

In this paper, starting from such a VNS framework, we discuss how the design of the neighborhood structures $N_1, \ldots, N_r$ and L can be enhanced in order to be performed in a collaborative and integrated fashion (note that integrated collaboration also appears in some ant algorithms [4], but within a different framework). The resulting VNS is called CVNS (for Collaborative VNS). In contrast with the standard literature on VNS, depending on the involved problem structure, the following features can appear in CVNS: (A) a strategic use of *destroying* neighborhood structures in M (i.e., moves which eliminate some pieces of the solution); (B) the use of a *central memory* Mem containing all the local minima encountered during the search, which is employed to design the stopping condition of LS. On the one hand, feature (A) allows for the joint action of moves of different types, resulting in a collaborative solution improvement process. On the other hand, feature (B) offers a way to collaboratively improve the performance of LS thanks to the sharing of information at a global level.

In this contribution, we discuss the use of such features, and the performance of CVNS is highlighted with the use of three problems belonging to different fields: (1) job scheduling with time-window penalties (Section 2 relying on [5]); (2) nonlinear global optimization (Section 3 relying on [6]); (3) network design (Section 4 relying on [7]). Only a baseline study is given for each problem, and the reader is referred to the above three references to have more detailed information on the complexity issues, the literature review, the parameter setting, and a finer-grained presentation of the experiments, including the experimental conditions like the computer type, the programming language, etc. The main numerical results are highlighted in this work, which allows to observe the good performance of CVNS. A conclusion is provided in Section 5. For recent VNS variants, the reader is referred to [8–10].

2 Job scheduling with time-window penalties

2.1 Presentation of the Problem (P)

Make-to-order production systems are relevant to face the customized products requested at the clients level [11]. The associated just-in-time paradigm appears as a relevant approach to reduce the inventory costs. In such a context and because of the limited production capacity, scheduling jobs at the plant level can result in rejecting some orders [12]. Surprisingly, the literature on order acceptance problems involving earliness or tardiness penalties is limited [13, 14]. Let (P) denote the considered NP-hard single-machine scheduling problem. It has the following features: sequence-dependent setup times and costs, earliness and tardiness penalties, and rejection penalties associated with the rejected jobs.

(P) can be presented as follows [15]. n jobs can be performed on a single machine, but two jobs cannot be processed concurrently. With each job j, the following information is associated: a due date d_j, a deadline $\bar{d_j}$, a rejection penalty u_j, an available date $\bar{r_j}$, a release date r_j, and a processing time p_j. Let S_j (resp. C_j) denote the starting time (resp. completion time) of job j. The following constraints are imposed for each job j: $S_j \geq \bar{r_j}$ and $C_j \leq \bar{d_j}$. If an accepted job j is not fully performed in time-window $[r_j, d_j]$, a penalty is encountered: if $S_j < r_j$ (resp. $C_j > d_j$), an earliness (resp. tardiness) penalty $E_j(S_j)$ (resp. $T_j(C_j)$) is paid, where $E_j(\cdot)$ (resp. $T_j(\cdot)$) is a non-increasing (resp. non-decreasing) function. In addition, a setup time (resp. cost) $s_{jj'}$ (resp. $c_{jj'}$) is encountered if two jobs j and j' of different families are consecutively performed. Idle times are allowed (indeed, they can have a positive impact on the earliness penalties), but preemptions are forbidden. Let $\sigma(s)$ (resp. $\Omega(s)$) be the sequence (resp. set) of accepted (resp. rejected) jobs associated with solution s. In order to measure the earliness/tardiness penalties of any solution s, it is necessary to first determine a starting time for each job of $\sigma(s)$. This is performed with a *timing procedure* [15] (this task is complex as idle times are allowed). The objective function to minimize is $f(s) = \sum_{j \in \sigma(s)} \left[E_j(S_j) + T_j(C_j) + c_{p_s(j)j}\right] + \sum_{j \in \Omega(s)} u_j$, where $p_s(j)$ is the predecessor of job j in $\sigma(s)$ (the predecessor of the first job is a dummy job representing the initial state of the machine).

2.2 CVNS for (P)

VNS has been applied to single-machine scheduling problems with different production environments [16]. When it is forbidden rejecting jobs, the neighborhood structures often consist in slightly changing the production sequence with move REINSERT or with move SWAP (as defined below). The strategic use of the move DROP (consisting in rejecting some jobs) is proposed in CVNS for (P).

Two methods are proposed for (P) in [15]: GR (a greedy heuristic) and TS (a tabu search using GR to generate an initial solution). GR consists in two phases: (1) sort the jobs by increasing slack times ($\bar{d_j} - \bar{r_j} - p_j$); (2) sequentially insert the jobs in the solution s under consideration, at the position minimizing the augmentation of the costs (but a job is rejected if it is cheaper than to do it). Four types of moves are used in TS in order to modify the current solution s: ADD moves a job from $\Omega(s)$ to $\sigma(s)$; DROP moves a job from $\sigma(s)$ to $\Omega(s)$; REINSERT reschedules one job in $\sigma(s)$; SWAP exchanges the positions of two jobs in $\sigma(s)$. If a move leads to an unfeasible solution s' (as available dates or deadlines are not respected), s' is immediately repaired as follows: while s' remains unfeasible, the job whose rejection leads to the smallest cost is removed (and the starting/ending times are also updated with the timing procedure). Four tabu structures were employed after applying a move on solution s. The first forbids adding a dropped job for τ_1 (parameter) iterations. The second forbids dropping an added job for τ_2 iterations. The third forbids (during τ_3 iterations) moving again a job that has been swapped, reinserted or added. If j has been reinserted or swapped, the fourth tabu status forbids moving a job j between its two previous neighboring jobs (in $\sigma(s)$) for τ_4 iterations.

In CVNS for (P), the initial solution is also generated by GR. The way to switch from one neighborhood to another differs from the standard Algorithm 1. $N_i(s)$ consists in randomly dropping $i\%$ of the jobs from $\sigma(s)$ to $\Omega(s)$. Such a move DROP is used here to diversify the search. Parameter i is managed in order to focus the search away from the current solution when no improvement has been made for a long period. More precisely, the proportion of removed jobs grows exponentially with the number of iterations without improvement. In step (1) of Algorithm 1, the selected solution is the best among k (parameter) solutions generated randomly in $N_i(s)$. In step (2) of Algorithm 1, the above presented TS is applied for I (parameter) iterations, but without move DROP (as it appears in the shaking process).

2.3 Results

The uniform distribution was used to generate all the data. Two values are important for generating instances for (P): the number n of jobs, and a parameter α impacting the time interval within which release dates and due dates are generated. Formally, a value $Start$ is selected large enough, and End is computed as $Start + \alpha \sum_j p_j$. Next, each r_j (resp. d_j) is randomly chosen in $[Start, End]$ (resp. $[r_j + p_j, End]$). Linear and quadratic penalties are investigated. More precisely, the earliness (resp. tardiness) penalties are computed as $w_j(r_j - S_j)^{q_j}$

(resp. $w'_j(C_j - d_j)^{q'_j}$). The weights w_j and w'_j are randomly picked in $\{1, 2, 3, 4, 5\}$, whereas q_j and q'_j are selected in $\{1, 2\}$. p_j is an integer randomly generated in $[50, 100]$, and $u_j = \beta_j \cdot p_j$, where β_j is an integer randomly picked in interval $[50, 200]$. $\bar{d_j}$ and $\bar{r_j}$ are generated such that $T_j(\bar{d_j}) = E_j(\bar{r_j}) = u_j$. The number of job families is chosen randomly in $[10, 20]$. Finally, setup costs and setup times are related (as in practice): the setup time $s_{FF'}$ between jobs of families F and F' is selected randomly in $[50, 200]$, and the corresponding setup cost $c_{FF'}$ is computed as $\lfloor \gamma \cdot s_{FF'} \rfloor$, where γ is randomly chosen in interval $[0.5, 2]$.

The quick timing procedure proposed in [17] was adapted to evaluate a solution in TS and CVNS. GR, TS and CVNS were tested with a time limit $T(n)$ depending on n. As GR is quick, it is restarted as long as T is not reached, and it finally returns the best generated solution among the restarts. Table 1 summarizes the results. Column "Best-known" indicates the best-known objective function value for each instance (in \$). Next, for each method, the percentage gap between the average result (over 10 runs) and "Best-known" is given. Both local search approaches are better than GR, and CVNS outperforms TS. Indeed, CVNS obtains the best results for 11 instances out of 15, versus 5 for TS. In other words, a strategic use of move DROP appears to be a powerful exploration tool: the ingredients added to TS to derive CVNS are thus efficient.

Table 1. Results for a job scheduling problem

n	α	**Best-known [\$]**	**GR [% gap]**	**TS [% gap]**	**CVNS [% gap]**
25	0.5	46,860	0.4	0.13	0.05
	1	35,866	6.5	0	0
	2	8,172	21.25	0.75	1.33
50	0.5	137,567	6.47	4.26	2.32
	1	69,671	44.34	10.15	11.26
	2	6,123	166.39	30.91	19.52
75	0.5	198,633	19.68	6.52	6.06
	1	126,052	33.93	5.15	0.6
	2	11,199	246.3	41.58	32.86
100	0.5	332,731	21.32	8.36	6.63
	1	175,237	50.36	25.65	4.6
	2	20,459	124.39	39.34	17.98
150	0.5	561,422	23.49	3.92	4.8
	1	320,225	53.85	11.76	15.6
	2	66,585	16.34	63.2	9.59
Average			**55.67**	**16.78**	**8.88**

3 Nonlinear global optimization

3.1 Presentation of the Problem (P)

Problem (P) consists in finding a global minimum of the nonlinear optimization problem $\min_{x \in \mathbb{R}^n} f(x)$, where function $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, but has no special structure. Most of the literature on nonlinear optimization [18–21] is usually dedicated on the global convergence of algorithms toward a local

optimum, with a fast local convergence. A point $x^\star$ is a *global* (resp. *local*) minimum of f if $f(x^\star) \leq f(x)$ for all $x \in \mathbb{R}^n$ (resp. if there exists $\varepsilon > 0$ such that $f(x^\star) \leq f(x)$ for each x such that $\|x - x^\star\| \leq \varepsilon$). An algorithm is *globally* (resp. *locally*) convergent if it converges to a (local) minimum from any starting point (resp. when it is converging to a (local) minimum when the starting point x_1 is in a given neighborhood of $x^\star$).

3.2 CVNS for (P)

The employed local search LS is able to prematurely stop its search if the iterates are converging to an already identified local minimum or if they are reaching an area of the solution space where no important improvement can be expected. LS relies on a trust region framework [20]. It is interrupted if one of the following conditions is verified: (1) a maximum number of iterations is reached; (2) LS has converged to a local minimum up to the desired precision; (3) LS seems to converge to an already identified local minimum; (4) the gradient norm is not large enough when the objective function value is far from the value at the best iterate; (5) a significant improvement of the objective function is not encountered. An efficient use of available information on f can strongly impact the design of the neighborhood structures. It was proposed to analyze the curvature of f at x based on the analysis of the eigenstructure of the Hessian matrix H, the approximation of the second derivatives matrix of f at x.

The main loop of CVNS is designed as follows. Let x be the current solution (which is the best visited solution, as in any classical VNS approach). Five neighborhood structures $N_1, \ldots, N_5$ are used (from the smallest N_1 to the largest N_5), and each time a neighborhood structure N_k is used, p candidates $x_1, \ldots, x_p$ (parameter tuned to 5) are generated in it and then improved with LS. If the five so performed LS have been prematurely stopped, a quick option consists in restarting the process with N_{k+1}. Otherwise (i.e., at least one local search application converged to a local minimum), the list Mem of local minima is updated. If the best local minimum of Mem is better (resp. not better) than x, the process is restarted with N_1 (resp. N_{k+1}), which represents a success (resp. a failure). The overall algorithm stops if N_5 has failed.

3.3 Results

CVNS was performed 100 times for each instance, and a run is successful if CVNS finds a global minimum. Two measures of performance are considered: the average percentage of success and the average number of function evaluations among the successful runs. This second criterion is very important when it is computationally cumbersome to evaluate a solution [22]. CVNS is compared with the following methods: (1) Direct Search Simulated Annealing (DSSA) [23]; (2) Continuous Hybrid Algorithm (CHA) [24]; (3) Simulated Annealing Heuristic Pattern Search (SAHPS) [25]; (4) Directed Tabu Search (DTS) [26]. Table 2 provides the number of successes over the 100 runs for 25 problems (left information), and the average number of function evaluations for successful runs

on the same 25 problems (right information). Some of the cells associated with competitors are empty if the corresponding information was not available. First, CVNS appears to be the most robust method as it gets a success rate of 100% for almost all instances. Second, CVNS has the lowest average number of function evaluations for most instances. Interestingly, the efficiency of CVNS on Zakharov (Z_n) and Rosenbrock (R_n) functions is improving when the dimension n of the problem augments from 2 to 10. CVNS is also able to significantly reduce the average number of f-evaluations for instances R_{10} and Z_{10}.

Table 2. Results for nonlinear global optimization

Problem	CVNS		CHA		DSSA		DTS		SAHPS	
RC	100	153	100	295	100	118	100	212	100	318
ES	100	167	100	952	93	1442	82	223	96	432
RT	84	246	100	132	100	252			100	346
SH	78	366	100	345	94	457	92	274	86	450
DJ	100	104	100	371	100	273	100	446	100	398
HM	100	335			100	225				
GR_6	100	807			90	1830				
CV	100	854			100	1592				
DX	100	2148			100	6941				
Z_2	100	251	100	215	100	186	100	201	100	276
Z_5	100	837	100	950	100	914	100	1003	100	716
Z_{10}	100	1705	100	4291	100	12501	100	4032	100	2284
Z_{50}	100	17932	100	75520			0	177125		
$H_{3,4}$	100	249	100	492	100	572	100	438	95	517
$H_{6,4}$	100	735	100	930	92	1737	83	1787	72	997
$S_{4,5}$	100	583	85	698	81	993	75	819	48	1073
$S_{4,7}$	100	596	85	620	84	932	65	812	57	1059
$S_{4,10}$	100	590	85	635	77	992	52	828	48	1035
R_2	100	556	100	459	100	306	100	254	100	357
R_5	100	1120	100	3290	100	2685	85	1684	91	1104
R_{10}	100	2363	83	14563	100	16785	85	9037	87	4603
R_{50}	100	11934	79	55356			100	510505		
R_{100}	100	30165	72	124302			0	3202879		

4 Network Design

4.1 Presentation of the Problem (P)

In the context of large-scale production-distribution networks, the considered problem (P) is an extension of the two-echelon multicommodity CFLPSS (capacitated facility location problem with single sourcing) with alternative facility configurations, direct shipments from manufacturing facilities, and inventory holding costs. The design of a supply chain network implies strategic decisions for: (1) opening/closing production and distribution centers; (2) reconfiguring some of these centers; (3) specifying their mission according to (a) the products they have to produce or stock and (b) the customers they should deliver. The related recent literature, usually on simpler problems, includes [27–30].

Consider a network made of sites for potential PDCs (production-distribution centers) $u \in U$ and DCs (distribution centers) $w \in W$. They represent locations

where a facility could be opened, or alternatively, existing facilities. The plants are able to manufacture a set of finished products $p \in P$. A product is actually a group of items needing the same type of production capacity. For each p, it may be possible to produce it only on a subset of sites $U_p \subseteq U$. The facilities have to deliver external demand zones (group of ship-to-points located in a specified geographical area) $d \in D$. Only a subset $P_d \subseteq P$ of products might be requested from a demand zone d. Finished products can be stocked in the PDCs, for which the mission consists in supplying the DCs and some demand zones (direct shipments). Each demand zone has to be delivered by a single source (either a PDC or a DC). In addition, in order to satisfy some predefined service criteria (e.g., next day delivery), a facility $s \in S$ could deliver only a subset of demand zones $D_s \subseteq D$ or, conversely, only a subset $S_d \subseteq S = U \cup W$ of the sites are positioned to supply a given demand zone $d \in D$.

The production/storage capacity and the fixed/variable costs characterize the configuration of each existing facility. Alternative configurations can be implemented for each potential site, corresponding to: (a) the addition of new space and/or equipment to augment its capacity; (b) a re-engineering of current equipments/layouts; (c) other facility specifications for the new sites. Therefore, a set J_s of possible configurations can be implemented for each site $s \in S$ of the potential network. For the considered planning horizon, each configuration $j \in J_s$ is characterized by the following information: a production capacity, a flexible storage capacity, a fixed exploitation cost, and a variable throughput cost (covering the relevant procurement/reception/production/handling/shipping expenses). The objective function to minimize is the sum of the following costs: configuration costs for the facilities (fixed + variable), inventory holding costs, and transportation costs. The constraints to satisfy are: the capacity constraints, the flow equilibrium in each node of the network, and the clients' demand satisfaction.

4.2 CVNS for (P)

First, note that only feasible solutions are generated. The following options are considered when performing a move: (1) each demand from zones $d \in D$ is supplied by any of the open center $s \in S_d$ while respecting the service criteria; (2) the capacity constraints (i.e., minimum and maximum) of the used configurations are all satisfied; (3) if there is a demand from a zone d which can only be supplied from a single center (i.e., if $|S_d| = 1$), then the center is always set as open during the search process. Moreover, if for each $p \in P_d$, the demand x_{pd} from a zone $d \in D_w$ is reassigned to a DC $w \in W$, then a new requirement (equal to x_{pd}) is created for the opened PDCs that can ship product p to DC w. The same kind of additional requirements can be designed when an existing PDC is closed. In both cases, the requirements induced at the first echelon are assigned to the second echelon center u with the lowest production and transportation cost, given that a configuration j with sufficient capacity can be employed. If there is not enough capacity, the outstanding requirement is attributed to the next best plant. This implies that the DCs can be delivered by various plants.

Let $W(v) \subseteq W$ (resp. $U(v) \subseteq U$) be the subset of opened DCs (resp. PDCs) associated with solution v. In other words, a pair $(W(v), U(v))$ characterizes each solution v. In the shaking phase of CVNS, the best neighbor solution is chosen, and the stopping condition is a time limit. Five neighborhood structures are used, denoted as N_1 to N_5. (1) $v' \in N_1(v)$ if a PDC is closed but another is opened. (2) $v' \in N_2(v)$ if an additional PDC is opened. (3) $v' \in N_3(v)$ if a PDC is closed. (4) $v' \in N_4(v)$ if an additional DC is opened. (5) $v' \in N_5(v)$ if a DC is closed. As the number of potential DCs is usually far above the number of PDCs, it is appropriate to test many possibilities for $W(v)$. Therefore, the W-shift moves (i.e., a DC is closed but another is opened) will be employed within the local search LS of CVNS.

Two important points related to the design of the above neighborhood structures should be raised: (1) which demand zones should be attributed to a center that is newly available (this is identified with add/shift moves); (2) to which centers must the demands of a closed center (identified with a drop-move or a shift-move) be reassigned? In both cases, the involved costs are the configuration/transportation/production/inventory costs. In order to tackle issue (2), suppose that center s' has to be opened. It is appropriate to assign demand zone $d \in D_{s'}$ to s' instead of its current supplier s if the sum of the costs is decreased, and if the minimum capacity constraint remains satisfied for s. It was however observed that such a reassignment of demands often leads to an infeasible solution s' according to its minimum capacity constraint. To repair it, additional clients are given to s' as follows. While the minimum capacity constraint of s' is violated, a demand zone $d \in D_{s'}$ that is not already delivered by s' is randomly chosen. If there exists an assignment (s'', d) (involving an already open s'') that leads to a solution with superior costs than the assignment (s', d), then d is assigned to s'. If such an assignment does not exist, s' cannot be opened. Issue (1) is tackled as follows. For each demand zone d associated with the investigated center to be closed, d is simply reassigned to the best (according to the costs) possible open center s_b. In the two cases, the tightest center configuration is chosen (i.e., the feasible capacity configuration with the smallest fixed cost).

Let v denote the current solution. The five neighborhood structures are used as follows, starting with $M = \{N_1, \ldots, N_5\}$. In the shaking phase of Algorithm 1, instead of initially choosing $i = 1$, i is randomly picked in $\{1, 2, \ldots, 5\}$, and the best solution v' in $N_i(v)$ is chosen. LS is then applied on v' to get v''. Next, if v'' outperforms v, M is set to $\{N_1, \ldots, N_5\}$ and v is updated (i.e., set $v = v''$). Otherwise: if $|M| > 1$, N_i is removed from M; but if $|M| = 1$, M is set back to $\{N_1, \ldots, N_5\}$. The next neighborhood structure in the shaking phase is then randomly chosen in M.

The employed LS is a tabu search using the W-shift moves (w, w') such that w and w' can supply common demand zones. When a W-shift move (w, w') is performed, it is then forbidden to close (resp. open) w' (resp. w) for a certain number of iterations. LS is stopped when a maximum number I of iterations without improving the best solution encountered so far is reached. Note that the use of filtering techniques [31] might be very helpful to reduce the search space.

4.3 Results

CVNS was tested on a 32-bit 2 GHz Dual Core computer with 1 GB of RAM. An exact method relying on CPLEX was also developed. Random instances with various sizes and cost structures were generated, based on realistic cases documented in [32]. A uniform distribution is used to generate the demand for the different demand zones, with lower/upper bounds based on the total production capacity of the network. It was always assumed that demand zones had to be delivered from facilities located at a distance up to 530 miles from its centroid. Different sizes were obtained by modifying the potential PDCs (4 or 6, with four possible configurations for each PDC), the potential DCs (60 or 100, with two possible configurations for each DC), the demand zones (500 or 1000), and the number of product families (3 or 20).

Average results (with computing times indicated in minutes) are summarized in Table 3, depending on the instances characteristics. The percentage gaps of CVNS are computed with respect to the optimal costs. The focus is put on four components: the number $|P|$ of products (3 or 20), the number $|D|$ of demand zones (500 or 1000), the number of centers ($|U| = 4$ or 6 PDCs, $|W| = 60$ or 100 DCs). Anytime a component is fixed, the three other components vary. One can conclude that on average: (1) CPLEX requires 375 minutes to find an optimal solution; (2) CVNS is able to find very competitive solutions (as the average gap is 0.84%) in 13 minutes. Moreover, CVNS appears to be very efficient with a large number of products (indeed, the average gap gets close to 0.30% in such cases). Finally, CVNS is more competitive if more decision variables are involved, which is a good indicator if larger instances have to be tackled.

Table 3. Results for a network design problem

	CPLEX		**CVNS**	
Characteristics	**Opt. cost [$]**	**Time**	**Gap [%]**	**Time**
$\|D\| = 500$ demand zones	69,207,275	341.58	0.89	12.5
$\|D\| = 1000$ demand zones	134,420,183.3	409.17	0.79	14.17
$\|P\| = 3$ products	25,726,628.25	217.92	1.43	13.33
$\|P\| = 20$ products	177,900,830	532.83	0.26	13.33
$(\|U\|, \|W\|) = (4, 60)$ centers	103,763,688.3	297	0.99	10
$(\|U\|, \|W\|) = (6, 100)$ centers	99,863,769.92	453.75	0.7	16.67

5 Conclusion

The performance of a metaheuristic can be evaluated according to several criteria [1]: (1) quality (value of the obtained results according to a given objective function); (2) speed (time needed to get competitive results); (3) robustness (sensitivity to variations in problem characteristics and data quality); (4) ease of adaptation; (5) ability to take advantage of problem structure. The CVNS methodology has a good overall behavior according to these criteria. Indeed, for

the above presented applications, the solution encoding and the employed moves account for the problem specific features. Next, CVNS is easy to adapt because it only relies on two ingredients (which have to be designed in a collaborative fashion): a local search LS and a collection M of neighborhood structures. In addition, the strategic use of M plays a key role in robustness. The quickness of LS leads to the quickness of CVNS (it is usually the case if an aggressive method is used, such as tabu search). Finally, quality is ensured because of the intensification capability of LS combined with the diversification ability of M. Among the future works on CVNS, we can mention the integration of other learning mechanisms [33] to better guide the search.

References

1. Zufferey, N.: Metaheuristics: some Principles for an Efficient Design. Computer Technology and Applications **3 (6)** (2012) 446 – 462
2. Gendreau, M., Potvin, J.Y.: Handbook of Metaheuristics. Volume 146 of International Series in Operations Research & Management Science. Springer (2010)
3. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24** (1997) 1097–1100
4. Zufferey, N.: Optimization by ant algorithms: Possible roles for an individual ant. Optimization Letters **6 (5)** (2012) 963 – 973
5. Thevenin, S., Zufferey, N.: Variable Neighborhood Search for a Scheduling Problem with Time Window Penalties. In: Proceedings of the 14th International Workshop on Project Management and Scheduling (PMS 2014), Munich, Germany (April 2014)
6. Bierlaire, M., Thémans, M., Zufferey, N.: A heuristic for nonlinear global optimization. INFORMS Journal on Computing **22 (1)** (2010) 59 – 70
7. Amrani, H., Martel, A., Zufferey, N., Makeeva, P.: A Variable Neighborhood Search Heuristic for the Design of Multicommodity Production-Distribution Networks with Alternative Facility Configurations. Operations Research Spectrum **33 (4)** (2011) 989 – 1007
8. dos Santos, J.P.Q., de Melo, J.D., Neto, A.D.D., Aloise, D.: Reactive search strategies using reinforcement learning, local search algorithms and Variable Neighborhood Search. Expert Systems with Applications **41** (2014) 4939 – 4949
9. Li, K., Tian, H.: A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. Applied Soft Computing **43** (2016) 469 – 479
10. Stenger, A., Vigo, D., Enz, S., Schwind, M.: An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transportation Science **47 (1)** (2013) 64 – 80
11. Mansouri, S.A., Gallear, D., Askariazad, M.H.: Decision support for build-to-order supply chain management through multiobjective optimization. International Journal of Production Economics **135** (2012) 24 – 36
12. Oguz, C., Salman, F.S., Yalcin, Z.B.: Order acceptance and scheduling decisions in make-to-order systems. International Journal of Production Economics **125** (2010) 200 – 211
13. Atan, M.O., Akturk, M.S.: Single CNC machine scheduling with controllable processing times and multiple due dates. International Journal of Production Research **46** (2008) 6087 – 6111

14. Shabtay, D., Gaspar, N., Yedidsion, L.: A bicriteria approach to scheduling a single machine with job rejection and positional penalties. Journal of Combinatorial Optimization **23** (2013) 395 – 424
15. Thevenin, S., Zufferey, N., Widmer, M.: Tabu search for a single machine scheduling problem with discretely controllable release dates. In: 12th International Symposium on Operations Research in Slovenia (SOR 2013). (2013) 1590 – 1595
16. Liao, C.J., Cheng, C.C.: A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. Computers & Industrial Engineering **52** (2007) 404 – 413
17. Hendel, Y., Sourd, F.: An improved earliness-tardiness timing algorithm. Computers & Operations Research **34** (2007) 2931 – 2938
18. Bertsekas, D.P.: Nonlinear Programming. 2nd edn. Athena Scientific, Belmont (1999)
19. Bierlaire, M.: Introduction à l'optimisation différentiable. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland (2006) In press.
20. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-Region Methods. Series on Optimization. MPS-SIAM, Philadelphia, USA (2000)
21. Nocedal, J., Wright, S.J.: Numerical optimization. Operations Research. Springer Verlag, New-York (1999)
22. Silver, E., Zufferey, N.: Inventory control of an item with a probabilistic replenishment lead time and a known supplier shutdown period. International Journal of Production Research **49** (2011) 923–947
23. Hedar, A., Fukushima, M.: Hybrid simulated annealing and direct search methods for nonlinear unconstrained global optimization. Optimization Methods and Software **17** (2002) 891–912
24. Chelouah, R., Siarry, P.: Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. European Journal of Operational Research **148** (2003) 335–348
25. Hedar, A., Fukushima, M.: Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. Optimization Methods and Software **19** (2004) 291–308
26. Hedar, A., Fukushima, M.: Tabu search directed by direct search methods for nonlinear global optimization. European Journal of Operations Research **170** (2006) 329–349
27. Ahuja, R.K., Orlin, J.B., Pallattino, S., Scaparra, M.P., Scutella, M.G.: A multi-exchange heuristic for the single-source capacitated facility location problem. Management Science **50 (6)** (2004) 749 – 760
28. Barahona, F., Chudak, F.A.: Near-optimal solutions to large-scale facility location problems. Discrete Optimization **2** (2005) 35 – 50
29. Michel, L., Hentenryck, P.V.: A simple tabu search for warehouse location. European Journal of Operational Research **157** (2004) 576 – 591
30. Zhang, J., Chen, B., Ye, Y.: A multi-exchange local search algorithm for the capacitated facility location problem. Mathematics of Operations Research **30 (2)** (2005) 389 – 403
31. Hertz, A., Schindl, D., Zufferey, N.: Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. 4OR **3 (2)** (2005) 139 – 161
32. Ballou, R.H.: Business logistics management. Prentice Hall, New Jersey (1992)
33. Schindl, D., Zufferey, N.: A Learning Tabu Search for a Truck Allocation Problem with Linear and Nonlinear Cost Components. Naval Research Logistics **62 (1)** (2015) 32 – 45

Shape Optimization of I-section Beam-columns with the Crow Search Algorithm

Hakan Ozbasaran

Eskisehir Osmangazi University, Department of Civil Engineering, Eskisehir/TURKEY
e-mail: ozbasaran@ogu.edu.tr, Tel: +90 (222) 239 37 50, Fax: +90 (222) 229 14 18

ABSTRACT

The standard hot-rolled sections are used to form the skeletal structural system in most of the steel structures. The steel manufacturers produce various hot-rolled shapes with different names depending on the country. The cross-section geometry of a member should be determined to provide structural material economy and construction ease. Even if the most appropriate hot-rolled section is selected for a member, a significant amount of structural material may be wasted. In this case, designing a built-up section with a better material distribution may provide a more economical solution compared to any hot-rolled standard shape.

Since the requirements of an engineering problem can mostly be satisfied by more than one value sets for the variables, it is not wrong to say that most of the design problems have multiple solutions. The modern procedures accept the best possible solution as the final design, which directs the designer towards structural optimization. It should be reminded that the definition of the "best" word mostly interpreted as "lightest" in structural engineering. There are three structural optimization types as size, shape and topology. Size optimization is to seek the best design by taking the size of the structural members as the design variables. On the other hand, topology optimization deals with the best material distribution. One of the best definitions for the shape optimization concept, which is the main concern of the study, is searching the best locations for the nodes of the finite element mesh of a structure without changing the connectivity information.

The research on the shape optimization of bars has a relatively long history compared to the other structural optimization problems. The works of Keller [1], Taylor [2] and Simitses et al. [3], which involve the shape optimization of columns, are the earlier studies in the field. As for the studies conducted on beam-columns in the last decade, Gil-Martín et al. presented the proportioning of steel beam-columns based on Reinforcement Sizing Diagrams (RSD) optimization methodology with code-based constraints [4]. Cheng et al. studied the optimum design of clamped beam-columns under the thermal load that maximizes the buckling temperature and the fundamental natural frequency of transverse vibrations [5]. Wang et al. presented the shape optimization of simply supported singly-symmetric cold-formed beams and beam-columns [6].

This study presents the performance of the Crow Search Algorithm (CSA) [7], which is one of the recently published metaheuristic algorithms, for the shape optimization of prismatic I-section beam-columns considering a set of structural and geometrical constraints. The CSA is inspired by the food seeking behavior of the crows. The crows try to steal each other's food by following another random picked crow to its food-hiding place. If they realize that they are being followed, they lead the follower to another random place that is different than their food-

hiding location [7]. The structure of the algorithm is simple. Two lists, which can be called as "position" and "memory" matrices, store the crow information. The former (position) is for the current position of the crows. The latter (memory) holds the best solutions that the crows have discovered throughout the optimization procedure. The candidate positions for the next iteration are calculated by a modest movement operator. Then, each crow moves to its candidate position if the solution represented by the new candidate position is feasible and the memory matrix is updated. The algorithm continues by repeating these steps until the stopping criteria have been met.

The first structural constraint of the optimization process prevents the yielding of the material. The von Mises yield criterion is considered to determine the full elastic capacity of the bars. The transversal loads on a member produce both normal and shear stresses. However, the influence of the shear stress is small compared to the normal stress on the I-section members with large spans. With these assumptions, it becomes easy to determine the limiting von Mises stress occurred in the critical cross-section of the member. The second constraint controls the excessive deflections. It is well-known that the deflection curve of a member that is under axial load(s) and transverse bending cannot be determined by superposition. Fortunately, it is practical to utilize energy methods to obtain the approximate deflection curve of the member considering the axial load - transverse bending interaction. The third constraint deals with the global out-of-plane stability of the members. The first (critical) global buckling mode of a bar varies according to the structural material properties, section geometry and loading case. The compression members may buckle by deflecting laterally (flexural buckling), twisting (torsional buckling) or both (flexural-torsional buckling). On the other hand, there is a buckling mode for the bending members such as beams and beam-columns called lateral-torsional buckling where the transversally deformed bar finds another equilibrium state by twisting and deflecting laterally (Fig. 1). Note that this study uses the equation provided by Pi and Trahair [8,9] to calculate the buckling loads of the members.

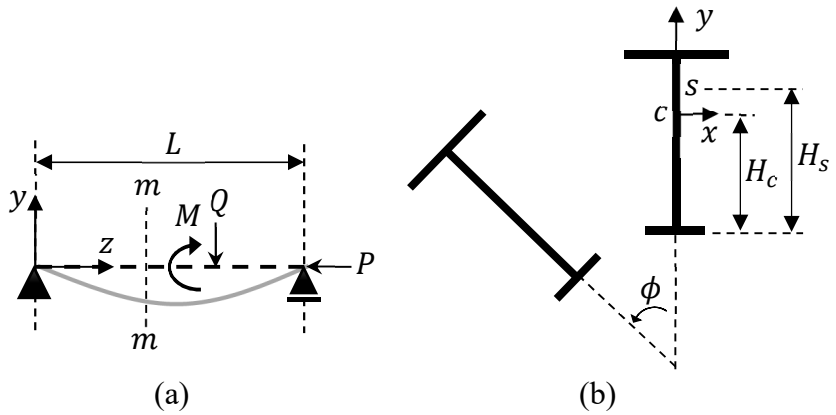


Fig. 1. A buckled beam-column a) Side view, b) m-m section

In Fig. 1, c and s are the center of mass and shear center of the section, respectively. The conventional design rules recommend using relatively thick plates for the flanges and a thinner plate for the web to construct an efficient I-section. On the other hand, slender webs are highly prone to buckling and therefore, the plate that forms the web of a built-up I-section should have

adequate strength to prevent local instability. For this reason, a geometric limitations set, which does not belong to any specific design code, is considered in the study as the fourth constraint.

The performance of the CSA for the shape optimization of the members under axial and transversal loads is demonstrated over beam-column design examples including simply-supported and cantilever configurations. The weights of the near-optimal designs found by a number of consecutive runs for each problem are adequately close to each other. The differences between the best and the average/worst results are acceptable and the standard deviations of each solution set are sufficiently small. Considering these remarks, it can be said that the CSA performed well in the introduced optimization problem. Another important note on the results is that the design problems (both simply-supported and cantilever configurations) except a few cases were controlled by the deflection and buckling constraints. Since the shape optimization attempts mostly leaded to slender designs, the buckling constraints dominated the problem and the stress constraints were implicitly satisfied even for the low-grade steels.

Keywords: *I-section, beam-column, shape optimization, metaheuristics, crow search algorithm*

References

[1] J.B. Keller, The shape of the strongest column, Arch. Ration. Mech. Anal. 5 (1960) 275–285. doi:10.1007/BF00252909.

[2] J.E. Taylor, The Strongest Column: An Energy Approach, J. Appl. Mech. 34 (1967) 486. doi:10.1115/1.3607710.

[3] G.J. Simitses, M.P. Kamat, C.V. Smith Jr., Strongest Column by the Finite Element Displacement Method, AIAA J. 11 (1973) 1231–1232. doi:10.2514/3.50571.

[4] L.M. Gil-Martín, M. Aschheim, E. Hernández-Montes, Proportioning of steel beam–column members based on RSD optimization methodology, Eng. Struct. 30 (2008) 3003–3013. doi:10.1016/j.engstruct.2008.04.004.

[5] G. Cheng, N. Yu, N. Olhoff, Optimum design of thermally loaded beam-columns for maximum vibration frequency or buckling temperature, Int. J. Solids Struct. 66 (2015) 20–34. doi:10.1016/j.ijsolstr.2015.04.008.

[6] B. Wang, G.L. Bosco, B.P. Gilbert, H. Guan, L.H. Teh, Unconstrained shape optimisation of singly-symmetric and open cold-formed steel beams and beam-columns, Thin-Walled Struct. 104 (2016) 54–61. doi:10.1016/j.tws.2016.03.007.

[7] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, Comput. Struct. 169 (2016) 1–12. doi:10.1016/j.compstruc.2016.03.001.

[8] Y.L. Pi, N.S. Trahair, Prebuckling Deflections and Lateral Buckling. I: Theory, J. Struct. Eng. 118 (1992) 2949–2966. doi:10.1061/(ASCE)0733-9445(1992)118:11(2949).

[9] Y.L. Pi, N.S. Trahair, Prebuckling Deflections and Lateral Buckling. II: Applications, J. Struct. Eng. 118 (1992) 2967–2985. doi:10.1061/(ASCE)0733-9445(1992)118:11(2967).

Hybrid weighted barebones exploiting particle swarm optimization algorithm for time series representation

Antonio Manuel Durán-Rosal*, David Guijo-Rubio, Pedro Antonio Gutiérrez, and César Hervás-Martínez

University of Córdoba, Dept. of Computer Science and Numerical Analysis, Córdoba, Spain

Abstract. The amount of data available in time series is recently increasing in an exponential way, making difficult time series preprocessing and analysis. This paper adapts different methods for time series representation, which are based on time series segmentation. Specifically, we consider a particle swarm optimization algorithm (PSO) and its barebones exploitation version (BBePSO). Moreover, a new variant of the BBePSO algorithm is proposed, which takes into account the positions of the particles throughout the generations, where those close in time are given more importance. This methodology is referred to as weighted BBePSO (WBBePSO). The solutions obtained by all the algorithms are finally hybridised with a local search algorithm, combining simple segmentation strategies (Top-Down and Bottom-Up). WBBePSO is tested in 13 time series and compared against the rest of algorithms, showing that it leads to the best results and obtains consistent representations.

Keywords: Time series representation, segmentation, barebones particle swarm optimization, hybrid algorithms

1 Introduction

Nowadays, the exponential increase of time series and their big amount of data hamper their processing [1]. Time series data mining (TSDM) includes several tasks such as the reconstruction of missing values [2], clustering [3], classification [4], forecasting [5] or segmentation [6]. Different areas of application can significantly benefit from efficient TSDM algorithms, including climate [7] or finances [8], among others.

Time series segmentation consists in dividing the time series into consecutive parts or points, trying to satisfy different objectives. There are two points of view that time series segmentation is focused on. On the one hand, segmenting time series is used for discovering patterns in them. On the other hand, there is another objective when segmenting time series, which tries to reduce the number

* Corresponding author at: Email: i92duroa@uco.es; Tel.: +34 957 218 349; Fax: +34 957 218 630.

of points of the time series (i.e. its dimensionality). With respect to this second objective, one of the main problems is the difficulty of processing and mining large time series, their dimensionality making them very difficult to analyse. Due to this fact, several algorithms have been proposed trying to simplify time series, which are also known as time series representation procedures. Keogh et al. [9] proposed different algorithms using piecewise linear approximations (PLA), which try to discover a set of points whose interpolations are the representation of the segments. Two PLA well-known algorithms are Top-Down and Bottom-Up, which iteratively reduce the error of the approximation. Other time series representation algorithms are the piecewise aggregate approximation (PAA) or the adaptative piecewise constant approximation (APCA) [10].

In this work, the contribution is focused on PLA segmentation algorithms, trying to find the set of points whose interpolations minimize the error of the resulting approximation. To do so, we propose a new variant of the barebones exploiting particle swarm optimization algorithm (BBePSO) [11], using the weighted average values of the visited positions of the particles and the best one from all the generations, instead of considering only the current ones. PSO is another evolutionary algorithm which simulates the behaviour of a set of particles when looking for food, and it has been applied to a lot of problems, such as routing vehicle [12], video tracking [13], etc. BBePSO has been proposed to improve the convergence of the standard BBPSO (which used a normal distribution to decide the movement of the particles), adding an exploiter component. In BBePSO, the algorithm converges using the current positions. In this paper, we show that the consideration of the past values in the evolution is important for the performance of the algorithm, and we propose a method in this direction, WBBePSO. This method takes the previous and the current positions into account, dynamically adapting the current positions by a weighted mean of the past values (giving more importance to those positions closer in time). This methodology modifies the mean and the standard deviation of the normal distribution considered in the standard BBePSO.

Evolutionary algorithms are able to find high-quality areas using populations of individuals. For this reason, they are robust heuristics which can solve different problems. However, their main drawback is their poor ability when finding the precise optimum in the areas they converge. The application of local searches in different parts of the evolutionary process is a way to prevent this problem. In this work, we combine the best solutions obtained by all the evolutionary methods (PSO, BBePSO and the proposed WBBePSO) with a local search combining Bottom-Up and Top-Down algorithms. In this sense, the resulting hybrid methods are referred to as HPSO, HBBePSO, and HWBBePSO, respectively.

This paper is organised in the following sections: Section 2 describes the problem of time series segmentation, Section 3 presents all the algorithms adapted to time series segmentation, Section 4 describes the datasets, the performed experiments and the discussion of the results (including a statistical validation) and Section 5 concludes the paper.

2 Time series segmentation problem definition

The main objective of this paper is to reduce the number of points of a given time series $Y = \{y_i\}_{i=1}^{N}$ in a set of L segments by cutting the values of the time series using $L-1$ cut points ($t_1 < t_2 < \cdots < t_{L-1}$). The error approximation resulting from the linear interpolation between the cut points needs to be minimised with the aim of avoiding information loss. That is, the cut points $\mathbf{t}$ (arranged from the smallest to the largest) are extracted from all time indexes, obtaining the set of segments $\mathcal{S} = \{s_1, s_2, \ldots, s_L\}$, where $s_1 = \{y_1, \ldots, y_{t_1}\}$, $s_2 = \{y_{t_1}, \ldots, y_{t_2}\}, \ldots, s_L = \{y_{t_{L-1}}, \ldots, y_N\}$. As stated before, a linear interpolation each pair of consecutive cut points is considered. Note that the cut points belongs to two segments (the previous and the next one). The number of segments of the approximation is a value predefined by the user. In order to solve this problem, we apply swarm intelligence algorithms.

3 Algorithms and their adaptations

This section presents the details of PSO, BBePSO and WBBePSO, together with their specific adaptation for time series segmentation.

3.1 Particle Swarm Optimisation algorithm (PSO) and its barebones exploiting version (BBePSO)

The particle swarm optimisation (PSO) [14] is another evolutionary-type algorithm which simulates the behaviour of a set of particles in a swarm when they are looking for food (i.e. birds or fish). The population of individuals corresponds with a set of K particles moving in a dimensional space of length D. Each particle k is represented by a position array of real values ($\mathbf{x}_k$), which represents a solution of the problem, and the velocity of the particle $\mathbf{v}_k$, which represents the strength and the direction of the movement of the particle. The quality of a particle is calculated by a fitness function (f). PSO also stores the best position found by the particle ($\mathbf{p}_k$) and the best position found by the entire swarm ($\mathbf{g}$). The evolution is based on a good compromise between local and global best positions, also known as cognitive and social components, respectively. In each iteration t, the PSO algorithm performs the following updates:

- Velocity update: the velocity $\mathbf{v}_k$ is updated in iteration t ($\mathbf{v}_k^t$) following the next expression.

$$\mathbf{v}_k^t = w \cdot \mathbf{v}_k^{t-1} + \rho_1^t \cdot C_1 \cdot \left(\mathbf{p}_k^{t-1} - \mathbf{x}_k^{t-1}\right) + \rho_2^t \cdot C_2 \cdot \left(\mathbf{g}^{t-1} - \mathbf{x}_k^{t-1}\right), \quad (1)$$

  where w is the inertia weight (a parameter used for velocity reduction, i.e., particles roaming), ρ_1^t, ρ_2^t are uniform random values obtained at iteration t, $\rho_1,\ \rho_2 \sim U(0,1)$, and C_1, C_2 are the acceleration constants.
- Position update: the position of a particle at iteration t ($\mathbf{x}_k^t$) is then updated using the expression:

$$\mathbf{x}_k^t = \mathbf{x}_k^{t-1} + \mathbf{v}_k^t. \quad (2)$$

– Best local and global position update: finally, the best local position at iteration t is:

$$\mathbf{p}_k^t = \begin{cases} \mathbf{p}_k^{t-1} \text{ if } f(\mathbf{x}_k^t) \geq f(\mathbf{p}_k^{t-1}), \\ \mathbf{x}_k^t \text{ if } f(\mathbf{x}_k^t) < f(\mathbf{p}_k^{t-1}), \end{cases} \tag{3}$$

for $k = 1, \ldots, K$, and the global best position is updated as:

$$\mathbf{g}^t = \arg\min_{\mathbf{x} \in \{\mathbf{g}, \mathbf{p}_1^t, \ldots \mathbf{p}_K^t\}} \{f(\mathbf{x})\}. \tag{4}$$

Note that we consider minimisation problems (the lower value of $f(\mathbf{x}_k)$, the higher quality of $\mathbf{x}_k$), which is the case of the problem to solve (minimisation of approximation error).

An improved version of PSO is the exploiting barebones PSO (BBePSO) [11]. This algorithm updates the position of the particles in the swarm without considering velocities. BBePSO replaces Equations 1 and 2 by:

$$x_{k,j}^t = \begin{cases} N\left(\frac{p_{k,j}^{t-1} + g_j^{t-1}}{2}, |p_{k,j}^{t-1} - g_j^{t-1}|\right) \text{ if } U(0,1) < 0.5, \\ p_{k,j}^{t-1} \qquad \text{otherwise}, \end{cases} \tag{5}$$

where $N(\mu, \sigma)$ is a normal distribution with mean (μ) equal to the average value of the best global and local positions, and the standard deviation (σ) equal to the difference, in absolute terms, of their values. This expression represents a 0.5 probability that the j-th dimension of the particle k takes a random value from the previous normal distribution (exploration) or from the best personal position (exploitation). BBePSO outperforms other variants of PSO [11], and it is also better when the values of the velocities or the acceleration constants are difficult to estimate.

PSO and BBePSO for time series segmentation: the particle representation corresponds to a real array ($\mathbf{x_i}$). The closest integer of each value represents a cut point, for instance, if $\mathbf{x_i} = \{2.56, 6.08, 9.10, 11.75\}$, its corresponding set of cut points in the time series is $\mathbf{t} = \{3, 6, 9, 12\}$. In this way, the length of the chromosome will be the same that the number of cut points $L - 1$. The initial population of PSO and BBePSO is formed by K particles with integer values without repetition (the values need to be unique). The standard procedures are used for updating velocities (in PSO), the particle positions and the best personal and global positions. However, after position update, the new particle has to satisfy two constraints:

– The values of the positions must be sorted, that is, $(x_{k,1} < x_{k,2} < x_{k,L-1})$. For this reason, if $x_{k,j} > x_{k,j+1}$, or $x_{k,j} < x_{k,j-1}$, the cut points of the chromosome are sorted in ascending order.
– Furthermore, the cut points need to be higher than 1 and lower than N. If this constraint is not satisfied, the algorithm rescales the values of the particle with:
$\mathbf{x}_i^{t'} = \frac{\mathbf{x}_i^t - \min\{\mathbf{x}_i^t\}}{\max\{\mathbf{x}_i^t\} - \min\{\mathbf{x}_i^t\}} (\max\{\mathbf{x}_i^{t-1}\} - \min\{\mathbf{x}_i^{t-1}\}) + \min\{\mathbf{x}_i^{t-1}\}$.

Finally, the algorithm is run until a number of evaluations is reached.

3.2 Weighted BBePSO (WBBePSO)

Our proposal is a new dynamic version of BBePSO. BBePSO updates the values of the positions taking into account the best personal and global positions. In this sense, the previous values are forgotten during the evolution when a particle is updated. In PSO, the velocities and the inertia weight w can be considered as a memory of the previous values, but this algorithm is poorer than BBePSO in finding solutions, given that it lacks an exploiter component. Keeping in mind the necessity of this memory and that more recent positions should be given more importance, we define a new Weighted BBePSO (WBBePSO), where the position update is made as follows:

$$x_{k,j}^t = \begin{cases} N\left(\frac{\overline{p}_{k,j}^{t-1}+\overline{p}_{g,j}^{t-1}}{2}, |\overline{p}_{k,j}^{t-1} - \overline{g}_j^{t-1}|\right) & \text{if } U(0,1) < 0.5, \\ p_{k,j}^{t-1} & \text{otherwise,} \end{cases} \tag{6}$$

where the best local position is updated as:

$$\overline{p}_{k,j}^t = \frac{\sum_{m=1}^t m p_{k,j}^m}{\sum_{m=1}^t m}, \tag{7}$$

and the best global one as:

$$\overline{g}_j^t = \frac{\sum_{m=1}^t m g_j^m}{\sum_{m=1}^t m}. \tag{8}$$

It is important to mention that the higher the value of m, the more importance is given to the solution, so that more recent particles have more influence in the update process.

WBBePSO for time series segmentation: the adaptation of the algorithm to time series segmentation follows the same considerations than for PSO and BBePSO (a real encoding, rounding the values to time indices, and a procedure for ensuring the fulfilling of the constraints).

3.3 Common elements for all the algorithms

This section presents the elements which are common for PSO, BBePSO and the proposed WBBePSO, i.e. the fitness function and the local search procedure.

Fitness function As we mentioned before, the main objective is to reduce the number of points of the time series with the minimum information loss. For that, we minimise the approximation error, which is the difference between a real point of the time series and its corresponding approximation. The error of the i-th point in the k individual is:

$$e_i(\mathbf{x}_k) = (y_i - \hat{y}_i(\mathbf{x}_k)), \tag{9}$$

where y_i is the real value of the time series, and $\hat{y}_i$ is the approximation resulting of the interpolation coded in individual $\mathbf{x}_k$. The fitness function is defined as the root mean square error:

$$\text{RMSE}(\mathbf{x}_k) = f(\mathbf{x}_k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2(\mathbf{x}_k)}. \tag{10}$$

Local search The best solution obtained by all evolutionary algorithms in the last generation (PSO, BBePSO, or WBBePSO) is hybridised with a local search [2] based on the combination of Bottom-Up and Top-Down segmentation procedures [9], resulting in hybrid algorithms (HPSO, HBBePSO, HWBBePSO). On the one hand, Bottom-Up is an iterative algorithm which starts considering each point of the time series as a cut point. In each iteration, it removes the cut point (merging two consecutive segments) that results in the minimum increase of approximation error. On the other hand, Top-Down is the opposite algorithm, starting by considering only one segment. Then, in each iteration, the algorithm recursively adds the cut point (splitting a segment) which results in the maximum decrease of error. The local search is based on removing a percentage of cut points with the Bottom-Up algorithm and adding the same percentage of points with Top-Down. For that, the stopping criteria of these algorithms is the number of segments to be merged or cut.

4 Experimentation

The time series used, the experiments performed and the discussion of the results are shown in this section.

4.1 Time series

For the experiments, we use 13 time series collected from different fields. Table 1 summarises the following information of each time series: name, type, length, and source. Also, the time series are represented in Fig. 1.

4.2 Experimental setting

We evaluate the performance of HWBBePSO against the rest of hybrid methods described in Section 3, and we analyse the existence of significant differences using statistical tests. For all the algorithms, we consider the same parameter configuration than in [18] (which has been proved to be effective for many different optimisation problems): the population size is $K = 200$, the maximum number of evaluations is $20,000$, the inertia (w) is established to 0.72 and the acceleration constants (C_1, C_2) to 0.49. Finally, the percentages of reduction and hybridization are 2.5% and 40%, respectively. The percentage of reduction corresponds to the number of points of the approximation with respect to the original

Table 1. Time series used

Name	Type	Length	Source
Arrhytmia	Cardiology data	9000	PhysioBank ATM [15]
B41043	Wave Height TS	7303	NDBC [16] (Puerto Rico)
B41044	Wave Height TS	7303	NDBC [16] (Puerto Rico)
B46001	Wave Height TS	8767	NDBC [16] (Alaska)
B46075	Wave Height TS	7303	NDBC [16] (Alaska)
BBVA	Bank Market Rates	4174	(Spain)
DEUTSCHE	Bank Market Rates	4174	(Germany)
HandOutlines	Benchmark TS	8127	UCR Repository [17]
IBEX	Stock prices TS	5730	`https://es.finance.yahoo.com/`
Mallat	Benchmark TS	8192	UCR Repository [17]
SANPAOLO	Bank Market Rates	4174	(Italy)
Société Genéralé	Bank Market Rates	4174	(France)
StarLight	Benchmark TS	8192	UCR Repository [17]

size, while the percentage of hybridisation represents the number of cut points which are removed and added in the local search. All the algorithms are run 30 times with different seeds, due to their stochastic nature.

4.3 Discussion

The approximation errors in RMSE are shown in Table 2, together with associated average ranks ($R = 1$ for the best method in each dataset and $R = 3$ for the worse one). For all the algorithms, the mean and the standard deviation (SD) of the 30 runs are presented (Mean $\pm$ SD). As can be seen, HWBBePSO outperforms the rest of algorithms with the best results in all datasets except in the case of Société Générale, where it is the second best. The second best method seems to be HBBePSO with better results than HPSO in several datasets (B41043 or DEUTSCHE among others). However, this algorithm (HPSO) obtains lower errors in other datasets, such as BBVA or Mallat.

Analysing the standard deviation of the results, HWBBePSO presents the lowest values in almost databases (8 out of 13 datasets, and the second one in other two) showing its effectiveness and that the evolution does not depend on the initialisation. From this analysis, it can be observed that the algorithm HWBBePSO balances the use of previously visited positions, giving more importance to the most recent ones. In this way, it is able to converge to high-quality areas, avoiding a premature convergence, and, moreover, when combined with the local search, the resulting hybrid algorithm finds an optimum solution in these areas.

To determine the statistical significance of the rank differences observed for each swarm intelligence algorithm in the different time series, we have carried out a non-parametric Friedman test [19] with the ranking of RMSE of the best models as the test variable (since a previous evaluation of the RMSE values

Table 2. RMSE results and mean ranks ($\bar{R}_{RMSE}$) for all the algorithms

Algorithm	HPSO (Mean ± SD)	HBBePSO (Mean ± SD)	HWBBePSO (Mean ± SD)
Arrhytmia	*0.052 ± 0.002*	*0.052 ± 0.002*	**0.051 ± 0.001**
B41043	0.395 ± 0.006	*0.394 ± 0.006*	**0.389 ± 0.003**
B41044	0.392 ± 0.009	*0.391 ± 0.007*	**0.382 ± 0.004**
B46001	0.984 ± 0.008	*0.980 ± 0.007*	**0.975 ± 0.006**
B46075	1.046 ± 0.011	*1.040 ± 0.012*	**1.034 ± 0.009**
BBVA	*0.319 ± 0.008*	0.323 ± 0.009	**0.317 ± 0.008**
DEUTSCHE	1.926 ± 0.055	*1.915 ± 0.062*	**1.905 ± 0.076**
HandOutlines	**0.006 ± 0.000**	**0.006 ± 0.000**	**0.006 ± 0.000**
IBEX	*205.688 ± 3.894*	206.132 ± 3.954	**203.128 ± 4.085**
Mallat	*0.162 ± 0.007*	0.167 ± 0.007	**0.157 ± 0.009**
SANPAOLO	0.111 ± 0.003	*0.110 ± 0.002*	**0.109 ± 0.001**
SOGenéralé	2.154 ± 0.052	**2.127 ± 0.042**	*2.136 ± 0.031*
StarLightCurves	*0.024 ± 0.001*	*0.024 ± 0.001*	**0.023 ± 0.001**
$\bar{R}_{RMSE}$	2.62	2.23	1.15

The best method is shown in bold face and the second one in italics.

resulted in rejecting the hypothesis of normality and equality of variances). The test shows that the effect of the algorithm used for dimensionality reduction is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 3.40)$ and the F-distribution statistical value is $F^* = 16.16 \notin C_0$. Consequently, we reject the null-hypothesis stating that all the algorithms perform equally in mean ranking for RMSE.

Based on this rejection, the Holm post-hoc test is used to compare the three algorithms to each other. Holm test is a multiple comparison procedure that considers a control algorithm (CA), in this case HWBBePSO, and compares it with the remaining methods [20]. The test statistics for comparing the mean rank of the i-th and j-th algorithm using this procedure is:

$$z = \frac{\bar{R}_i - \bar{R}_j}{\sqrt{\frac{k(k+1)}{6N}}}, \tag{11}$$

where k is the number of algorithms and N the number of datasets. The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate level of confidence α. Holm's test adjusts the value for α in order to compensate for multiple comparison.

The results of the Holm test for $\alpha = 0.05$ can be seen in Table 3, using the corresponding p and α^*_{Holm} values. From the results of this test, it can be concluded that the HWBBePSO algorithm obtains a significantly higher ranking of RMSE when compared to the remaining two algorithms, which justifies the proposal.

Table 3. Holm tests comparing HWBBePSO (CA) with the rest of methods: p-values and α^*_{Holm} with initial $\alpha = 0.05$

CA: HWBBePSO	HPSO	HBBePSO
p-value	$1.90 \times 10^{-4}(*)$	$6.04 \times 10^{-3}(*)$
α^*_{Holm}	0.025	0.050

(*): statistically significant differences were found for $\alpha = 0.05$

Finally, to visually analyse the results, the approximations obtained by the algorithms HWBBePSO for all datasets are shown in Fig. 2. Comparing Figures 1 and 2, the algorithm results in faithful approximations of the real values.

5 Conclusions

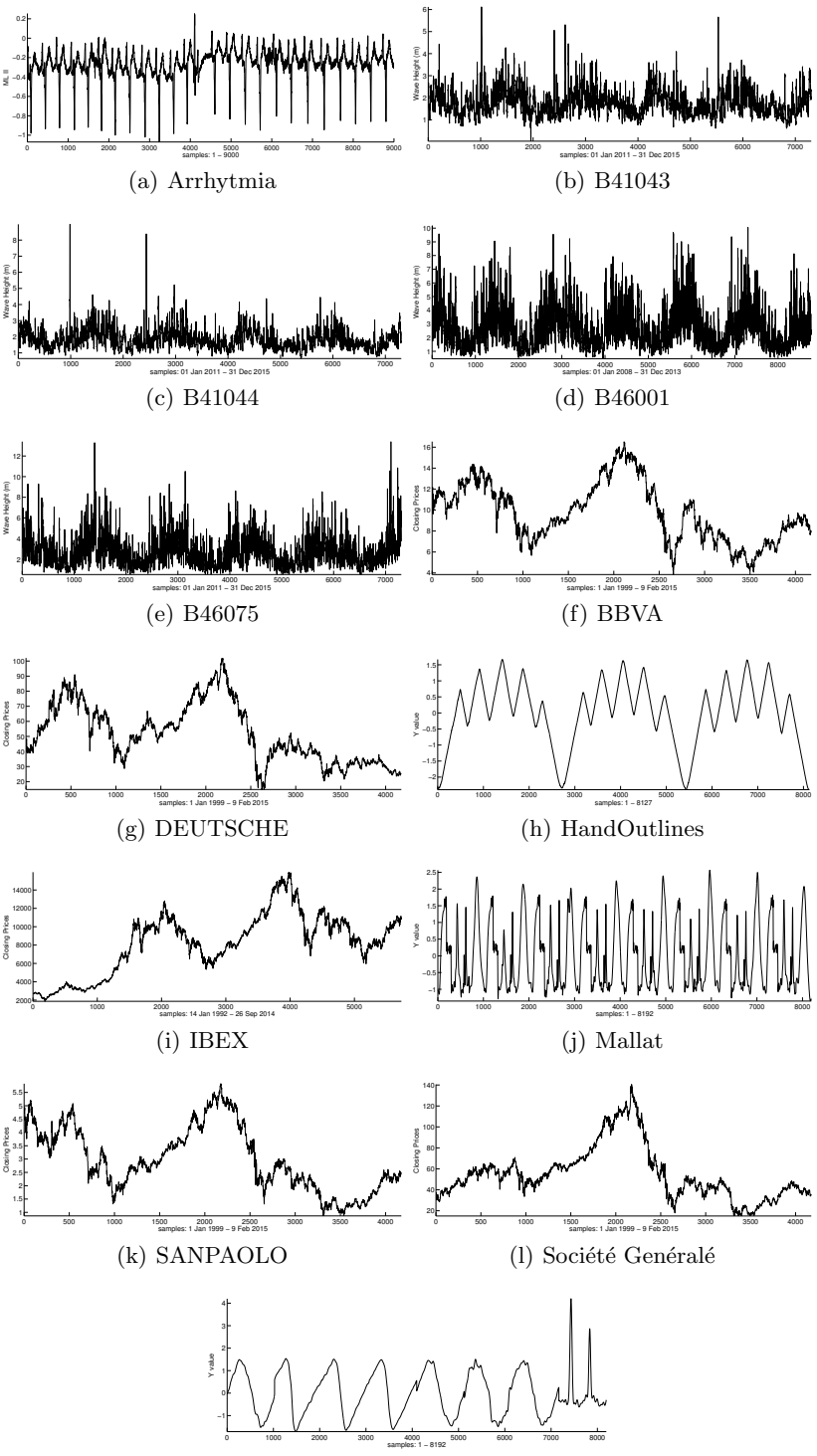
In this paper, we propose a new algorithm for time series segmentation with the objective of reducing the length of the time series with minimum information loss. The algorithm reduces the approximation error using interpolations of each segment. The paper includes a particle swarm optimisation algorithm (PSO), its exploiter barebones version (BBePSO), and a novel BBePSO version which takes into account the weighted value of the previously visited positions, giving more importance to recent ones (WBBePSO). The best solution obtained by all the algorithms are then improved with a local search procedure, resulting in hybrid versions (HPSO, HBBePSO, and HWBBePSO). The evaluation of the method is made in 13 time series from different fields.

The experiments show that the best results, those with lowest approximation error, are obtained by HWBBePSO, showing that considering all the positions visited by a particle during the evolution is good for this kind of problems. Moreover, the standard deviation of the new methodology is the lowest one in almost time series, i.e. the algorithm is less dependent on the initialisation. Finally, the approximated time series are represented, they being very similar to the original ones.

Future research includes considering the approximated time series in other posterior tasks, using other types of approximations (instead of PLA) and adding more time series to the validation.

Acknowledgement

This work has been subsidized by the projects TIN2017-85887-C2-1-P, TIN2014-54583-C2-1-R and TIN2015-70308-REDT of the Spanish Ministry of Economy and Competitiveness (MINECO), and FEDER funds (FEDER EU). The research of Antonio M. Durán-Rosal and David Guijo-Rubio have been subsidized by the FPU Predoctoral Program of the Spanish Ministry of Education, Culture and Sport (MECD), grant references FPU14/03039 and FPU16/02128.



(a) Arrhytmia (b) B41043

(c) B41044 (d) B46001

(e) B46075 (f) BBVA

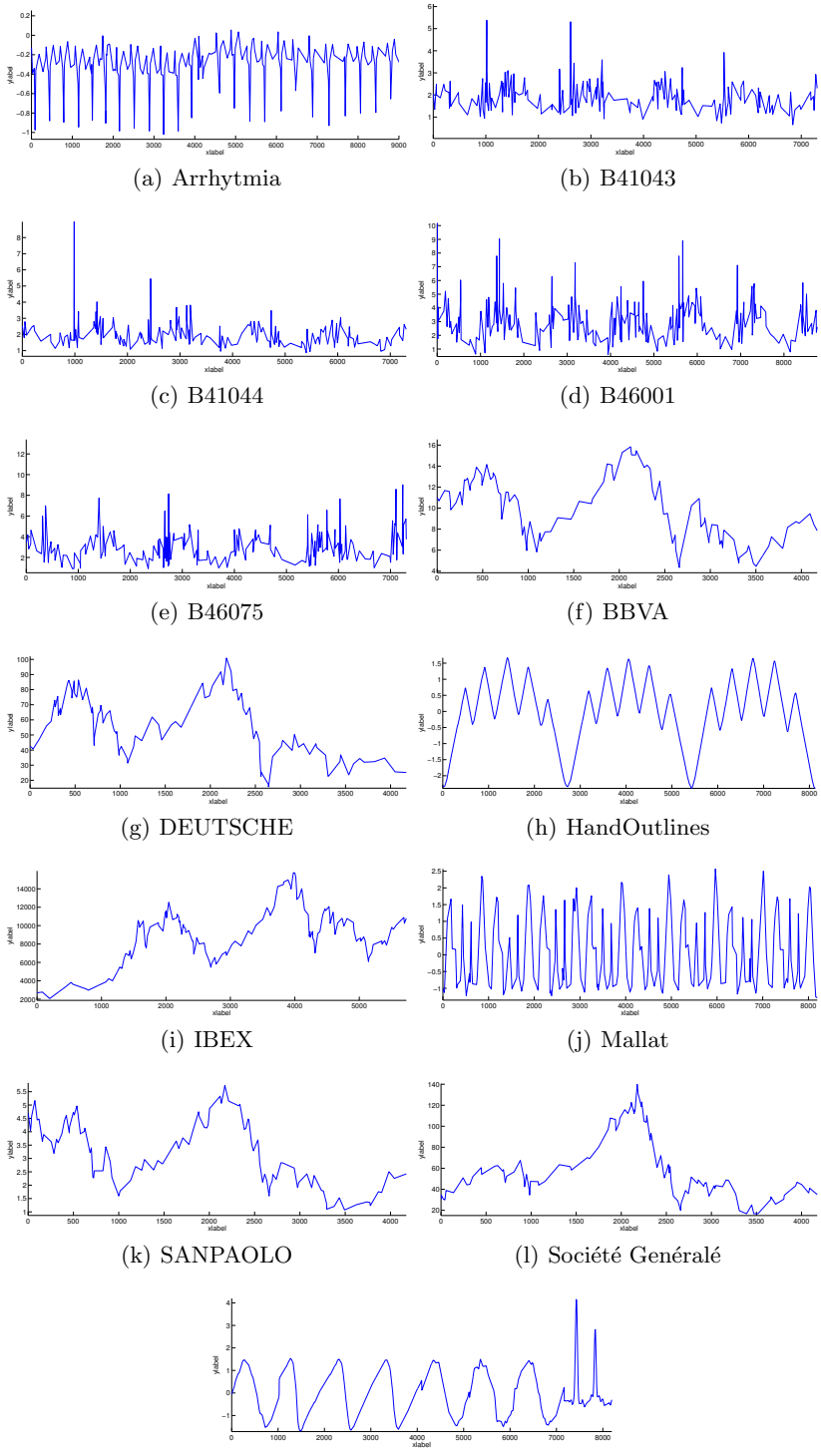
(g) DEUTSCHE (h) HandOutlines

(i) IBEX (j) Mallat

(k) SANPAOLO (l) Société Genéralé

(m) StarLight

Fig. 1. Time series considered for the experiments.



(a) Arrhytmia (b) B41043

(c) B41044 (d) B46001

(e) B46075 (f) BBVA

(g) DEUTSCHE (h) HandOutlines

(i) IBEX (j) Mallat

(k) SANPAOLO (l) Société Genéralé

(m) StarLight

Fig. 2. Approximation time series of HWBBePSO.

References

1. Esling, P., Agon, C.: Time-series data mining. ACM Computing Surveys (CSUR) **45** (2012) 12
2. Durán-Rosal, A.M., Gutiérrez-Peña, P.A., Martínez-Estudillo, F.J., Hervás-Martínez, C. In: Time Series Representation by a Novel Hybrid Segmentation Algorithm. Springer International Publishing, Cham (2016) 163–173
3. Ferreira, L.N., Zhao, L.: Time series clustering via community detection in networks. Information Sciences **326** (2016) 227–242
4. Zhao, J., Itti, L.: Classifying time series using local descriptors with hybrid sampling. IEEE Transactions on Knowledge and Data Engineering **28** (2016) 623–637
5. Chen, M.Y., Chen, B.T.: A hybrid fuzzy time series model based on granular computing for stock price forecasting. Information Sciences **294** (2015) 227–241
6. Pérez-Ortiz, M., Durán-Rosal, A., Gutiérrez, P., Sánchez-Monedero, J., Nikolaou, A., Fernández-Navarro, F., Hervás-Martínez, C.: On the use of evolutionary time series analysis for segmenting paleoclimate data. Neurocomputing. (2017)
7. Nikolaou, A., Gutiérrez, P.A., Durán, A., Dicaire, I., Fernández-Navarro, F., Hervás-Martínez, C.: Detection of early warning signals in paleoclimate data using a genetic time series segmentation algorithm. Climate Dynamics **44** (2015) 1919–1933
8. Gong, X., Si, Y.W., Fong, S., Biuk-Aghai, R.P.: Financial time series pattern matching with extended ucr suite and support vector machine. Expert Systems with Applications **55** (2016) 284–296
9. Keogh, E.J., Chu, S., Hart, D., Pazzani, M.: Segmenting Time Series: A Survey and Novel Approach. In: Data Mining In Time Series Databases. (2004) 1–22
10. Chakrabarti, K., Keogh, E., Mehrotra, S., Pazzani, M.: Locally adaptive dimensionality reduction for indexing large time series databases. ACM Transactions on Database Systems (TODS) **27** (2002) 188–228
11. Kennedy, J.: Bare bones particle swarms. In: Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE. (2003) 80–87
12. l Okulewicz, M., Mandziuk, J.: A particle swarm optimization hyper-heuristic for the dynamic vehicle routing problem. In: 7th BIOMA Conference. (2016) 215–227
13. Zhang, M., Xin, M., Yang, J.: Adaptive multi-cue based particle swarm optimization guided particle filter tracking in infrared videos. Neurocomputing **122** (2013) 163 – 171 Advances in cognitive and ubiquitous computing.
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on. Volume 4. (1995) 1942–1948
15. Moody, G., Mark, R.: The impact of the MIT-BIH arrhythmia database. Engineering in Medicine and Biology Magazine, IEEE **20** (2001) 45–50
16. http://www.ndbc.noaa.gov/: National buoy data center. National Oceanic and Atmospheric Administration of the USA (NOAA) (2015)
17. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The ucr time series classification archive (2015) `www.cs.ucr.edu/~eamonn/time_series_data/`.
18. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE transactions on Evolutionary Computation **6** (2002) 58–73
19. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics **11** (1940) 86–92
20. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research **7** (2006) 1–30

Two-stage Candidate Evaluation Strategy in Optimization of Truss Structures with Metaheuristics

M. Eryilmaz[1], H. Ozbasaran[2]

[1,2]Eskisehir Osmangazi University, Department of Civil Engineering, Eskisehir/TURKEY
e-mail: [1]meryilmaz@ogu.edu.tr, [2]ozbasaran@ogu.edu.tr
Tel: +90 (222) 239 37 50, Fax: +90 (222) 229 14 18

ABSTRACT

In today's world, human beings aim to utilize the maximum benefit from the limited amount of available resources. For example, in engineering design, engineers tend to design structures that meet all design requirements at the possible lowest cost. Optimization plays an important role in structural design, providing engineers with a variety of techniques to deal with such problems [1]. Size optimization has become a major field among researchers with the increasing processing power of the computers. The design variables are taken as the cross-sectional areas of the structural members and the constraints are the limitations imposed on stresses and displacements that occur in the structure under the applied loads. The objective function is generally considered as to minimize the overall or material cost of the structure. Searching for the optimum solutions of truss systems is one of the common benchmark problems in the field of structural engineering. In general terms, the optimum design of a truss structure is finding the optimum cross sectional areas for its members that results in a minimum weight or cost design of the structure. This is a complex and nonlinear design process, which causes the optimization problem to be highly nonlinear and thus difficult to solve. In addition to adversity of the problem, as the members of truss structures used in real life come from a finite number of standard cross-sections in the tables, the optimization problem is discrete in nature and requires metaheuristic algorithms that can handle discrete variable sets [2].

Initially, mathematical optimization techniques were used in structural optimization. However, designing an optimal structure with mathematical algorithms requires gradient information of the problem and the initial estimate of the solution vector to start the iterations. These problems have led the researchers to investigate new approaches based on different concepts, resulting in the emergence of stochastic search algorithms called "metaheuristics" [3]. Metaheuristic algorithms do not require gradient information. They have an ability to handle both discrete and continuous problems and decent incorporating global search features to yield reasonable solutions for mathematically complex problems [4]. The reviews of the widely-used metaheuristic algorithms in the field of structural optimization analyses are outlined in several extensive review articles [2,3,5–8].

Metaheuristics are usually applied to the problems that do not have a satisfactory problem-specific algorithm to solve them. They have a wide application field in industry and services, to solve the complex problems from finance to production management and engineering [8]. However, the slow convergence rate and the excessively time-consuming number of structural analyses are still considered to be fundamental deficiencies of these techniques in structural optimization applications [9]. To overcome these shortcomings, researchers proposed the improved versions of the existing algorithms such as enhanced particle swarm optimization [10] and enhanced biogeography-based optimization [11]. Yet, these

improved algorithms provide solutions for specific algorithms and problems. Another remedy proposed by the researchers is the hybridization of the optimization algorithms, which enables the use of the powerful and successful features of various algorithms in one hybrid algorithm. Christian and his colleagues reviewed hybrid metaheuristics in their comprehensive survey [12].

An alternative way to improve the computational efficiency of metaheuristics is to implement a strategy, which may be developed by inspiration from nature or some other selection scheme. In order to implement these strategies to the metaheuristic algorithms, the algorithm should have the same specific features with the proposed strategy. Azad et al. proposed upper bound strategy for optimization of steel frames with metaheuristic algorithms [13]. The upper bound strategy reduces the total number of structural analyses through avoiding unnecessary analyses during the course of optimization. Once identified, the non-improving designs are directly excluded from the structural analysis stage, thus the computational effort decreases significantly. This strategy could be implemented to the algorithms that employ a $\mu + \lambda$ selection scheme in their algorithmic models. In this selection scheme, μ parents generate λ offsprings and η (which is equal to μ) best individuals that are selected out of μ parents and λ offsprings according to their fitness values; thus, the number of individuals remains the same in each generation [13].

In size optimization of trusses, the "best" solution should be interpreted as "the lightest design that does not violate any of the constraints" in most of the applications. The "worst" design is "the design with the greatest penalty amongst the solutions that violates the constraints". Implementing the penalty function approach, the infeasible designs are moved to the ends of the candidates list and the solution with the greatest penalty generally has the most unfavorable objective function. Some of the metaheuristic algorithms order the candidates with respect to their objective values and/or use the best and worst solution obtained so far (global best and worst) or the ones in the candidates list (local best and worst) to build the next generation of candidates. The constraint functions should be evaluated and penalized objective values should be found to determine which solution is the worst. The other group of metaheuristics does not need to determine the best and the worst solution to apply the movement operator. Considering that the expensive operations of the truss optimization problems are checking the displacement and stress constraint violations by performing static analysis of the truss, any strategy that eliminates the redundant analyses improves the performance of the algorithm significantly.

This paper presents Two-stage Candidate Evaluation Strategy (TCES) for optimization of truss structures with metaheuristics to eliminate the unnecessary structural analyses during the optimization process and improve the computational efficiency for the algorithms that do not use the objective function values of the candidate solutions in their movement operators. The proposed technique considerably reduces the computational effort while maintaining the exploration/exploitation capability of the algorithms in solving optimization problems. The efficiency of TCES is validated by implementing it to the Crow Search Algorithm (CSA) [14], which is one of the recently published metaheuristic algorithms that does not use the objective function values of the candidate solutions, yet it can be integrated with any other metaheuristic algorithms of which movement operators do not take objective values of other individuals into consideration. The efficiency and reliability of the proposed strategy are demonstrated through the common benchmark problems of steel truss-sizing problems as 10-bar cantilever truss with

continuous and discrete variables [15] and numerical results are discussed in detail. Numerical results reveal that the proposed strategy significantly decreases the number of the structural analyses performed to converge to a near optimal result for each benchmark problem. The lightest designs in the existing literature are around 5062 and 5490 lbs for the continuous and discrete variable versions of the 10-bar cantilever truss sizing problem, respectively. It is shown that the TCES can provide up to 8.66 times saving in computational effort.

Keywords: *optimization, truss structures, bioinspired, metaheuristics, structural design*

References

[1] S. Talatahari, M. Kheirollahi, C. Farahmandpour, A.H. Gandomi, A multi-stage particle swarm for optimum design of truss structures, Neural Comput. Appl. 23 (2013) 1297–1309. doi:10.1007/s00521-012-1072-5.

[2] R. Alberdi, K. Khandelwal, Comparison of robustness of metaheuristic algorithms for steel frame optimization, Eng. Struct. 102 (2015) 40–60. doi:10.1016/j.engstruct.2015.08.012.

[3] M.P. Saka, O. Hasançebi, Z.W. Geem, Metaheuristics in structural optimization and discussions on harmony search algorithm, Swarm Evol. Comput. 28 (2015) 88–97. doi:10.1016/j.swevo.2016.01.005.

[4] O. Hasançebi, S.K. Azad, Adaptive dimensional search: A new metaheuristic algorithm for discrete truss sizing optimization, Comput. Struct. 154 (2015) 1–16. doi:10.1016/j.compstruc.2015.03.014.

[5] A.E. Charalampakis, Comparison of metaheuristic algorithms for size optimization of trusses, in: 11th HSTAM Int. Congr. Mech., Athens, 2016.

[6] A. Kaveh, A. Zolghadr, Comparison of nine meta-heuristic algorithms for optimal design of truss structures with frequency constraints, Adv. Eng. Softw. 76 (2014) 9–30. doi:10.1016/j.advengsoft.2014.05.012.

[7] O. Hasançebi, S. Çarbaş, E. Doğan, F. Erdal, M.P. Saka, Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures, Comput. Struct. 87 (2009) 284–302. doi:10.1016/j.compstruc.2009.01.002.

[8] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. (Ny). 237 (2013) 82–117. doi:10.1016/j.ins.2013.02.041.

[9] S.K. Azad, O. Hasançebi, Discrete sizing optimization of steel trusses under multiple displacement constraints and load cases using guided stochastic search technique, Struct. Multidiscip. Optim. 52 (2015) 383–404. doi:10.1007/s00158-015-1233-0.

[10] H. Cao, X. Qian, Z. Chen, H. Zhu, Enhanced particle swarm optimization for size and shape optimization of truss structures, Eng. Optim. 49 (2017) 1939–1956. doi:10.1080/0305215X.2016.1273912.

[11] S.H.S. Taheri, S. Jalili, Enhanced biogeography-based optimization: A new method for size and shape optimization of truss structures with natural frequency constraints, Lat. Am. J. Solids Struct. 13 (2016) 1406–1430. doi:10.1590/1679-78252208.

[12] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: A survey, Appl. Soft Comput. J. 11 (2011) 4135–4151. doi:10.1016/j.asoc.2011.02.032.

[13] S.K. Azad, O. Hasançebi, Upper bound strategy for metaheuristic based design optimization of steel frames, Adv. Eng. Softw. 57 (2013) 19–32. doi:10.1016/j.advengsoft.2012.11.016.

[14] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, Comput. Struct. 169 (2016) 1–12. doi:10.1016/j.compstruc.2016.03.001.

[15] C. V. Camp, Design of Space Trusses Using Big Bang–Big Crunch Optimization, J. Struct. Eng. 133 (2007) 999–1008. doi:10.1061/(ASCE)0733-9445(2007)133:7(999).

Ensemble and Fuzzy techniques applied to Imbalanced Traffic Congestion Datasets: a Comparative Study

Pedro Lopez-Garcia[1,2], Antonio D. Masegosa[1,2,3], Enrique Onieva[1,2], Eneko Osaba[1]

[1] DeustoTech-Fundacion Deusto, Deusto Foundation, 48007, Bilbao, Spain
[2] Faculty of Engineering, University of Deusto, 48007, Bilbao, Spain
[3] IKERBASQUE, Basque Foundation for Science, 48011, Bilbao, Spain.
{p.lopez, ad.masegosa, e.onieva, e.osaba}@deusto.es

Abstract. Class imbalance is among the most persistent complications which may confront the traditional supervised learning task in real-world applications. Among the different kind of classification problems that have been studied in the literature, the imbalanced ones, particularly those that represents real-world problems, have attracted the interest of many researchers in recent years. In order to face this problems, different approaches have been used or proposed in the literature, between then, soft computing and ensemble techniques. In this work, ensembles and fuzzy techniques have been applied to real-world traffic datasets in order to study their performance in imbalanced real-world scenarios. KEEL platform is used to carried out this study. The results show that different ensemble techniques obtain the best results in the proposed datasets.

Keywords: Intelligent Transportation Systems, Imbalanced Data, Ensemble techniques, Fuzzy techniques, Soft Computing techniques, Classification

1 Introduction

Class imbalance is among the most persistent complications which may confront the traditional supervised learning task in real-world applications [1]. The problem appears when the number of instances in one of the classes significantly outnumbers the number of instances in the other ones. This situation is a handicap when trying to identify the minority class, as the learning algorithms are not usually adapted to such characteristics. Without the loss of generality, it can be assumed that the class of interest is the minority class, while the other ones are the majority ones. Various applications demonstrate this characteristic of high class imbalance, such as bioinformatics, e-business, information security, and national security.

Among the different kind of classification problems that have been studied in the literature, the imbalanced ones, particularly those that represents real-world problems, have attracted the interest of many researchers in recent years

[2, 3]. In particular, in traffic environments, the apparition of a particularly complicated state of the road (i.e. traffic congestion) will represent a minority class for prediction algorithms, while its proper detection in advance is a topic of interest for administrations and users.

One of the most problematic issues in the development of actual cities is road traffic. This problem is actually one of the most important study focuses of the Intelligent Transportation Systems (ITS) field. In the last decades, intelligent techniques such as those mentioned before have been applied to solve this problem. In particular fuzzy systems are used in [4] to infer the future state of the road by combining several systems in a hierarchical way. In addition different metaheuristics have been used in order to optimize systems, such as Support Vector Machines [5] (SVM); Genetic Algorithms (GA) are used in [6], while Particle Swarm Optimization (PSO)is implemented in [7], among others.

Recently, ensemble learning is a popular and significant research in data mining and machine learning area. Ensemble classifiers have received considerable attention in applied statistics and machine learning for over a decade [8]. Several studies demonstrate that the practice of combining several models into a aggregated one leads to significant gains in performance over its constituent members [9].

The principal aim is to make a comparative study between the performance of ensembles and fuzzy recent approaches in traffic state prediction, which is a multi-classification problem with a high imbalance between classes. Data used in this work come from two sources. The first one comes from cameras in the city of Helmond (The Netherlands) collected by TASS International company [1] and took part of the developing of different models for traffic systems in Horizon 2020 TIMON project [2] (Enhanced real time services for optimized multimodal mobility relying on cooperative networks and open data). Another data source used for the development of this work is the data obtained in Lisbon (Portugal) A5 highway, and used in the European Project ICSI (Intelligent Cooperative Sensing for Improved Traffic Efficiency).

The rest of the paper is structured as follow. Section 2 contains the state of the art of the two kind of techniques applied in this work: ensembles and metaheuristics. Section 3 is dedicated to the descriptions of the different methods used for this comparative study. In Section 4 information about the datasets used and its comparative is shown. Finally, in Section 5 the conclusions obtained for this study are collected.

2 Background

In this section, a brief study of the state of the art is presented in order to show the contributions of the community to the imbalance data problem using ensembles (Section 2.1), in specially boosting and bagging algorithms, and metaheuristics (Section 2.2).

[1] https://www.tassinternational.com/

[2] https://www.timon-project.eu/

2.1 Ensembles

Ensemble learning is defined as the use of multiple learning algorithms to obtain better predictive performance that could be obtained from any of these algorithms alone [10]. Over the last decade, this kind of approach has been used in different themes such as optimization [11], medicine [12], or ITS [13]. Focusing in imbalance classification problems, these algorithms can be found in many articles. For example, in [14], Lim et al. propose a evolutionary cluster-based oversampling ensemble framework. This method is based on contemporary ideas of identifying oversampling regions using clusters. The evolutionary part of the ensemble is used to optimize the parameters of the data generation method and to reduce the overall computational cost. The proposal is applied to a set of 40 imbalance datasets.

Among the different ensemble techniques, two of them can be frequently found in the literature applied to several themes: bagging and boosting techniques [15]. While in bagging several models are created using different subsets of the training set [16], in boosting, a set of weak learning algorithms create a single strong learner and produce only one model [17]. Both kind of methods have been used in imbalance classification.

Authors in [9] analyze different corrective and total corrective boosting algorithms in order to present its own boosting algorithm adding a strong classifier to the linear constraints of LPBoost. Besides, in [18], an Adaboost algorithm to learn fuzzy-rule-based classifiers is proposed. Adaboost approach is applied to approximate and descriptive fuzzy-rule bases, and the performance of the proposed method is compared with other classification schemes applied on a set of benchmark classification tasks.

Other example can be found in[19]. This article presents a research about the Roughly Balanced Bagging and its basic properties that can influence its classification performance. Variables such as the number of component classifiers, their diversity, and ability to deal with difficult types of the minority examples are studied. The experiments are carried out using synthetic and real life data.

The number of articles related with this theme is wide extended in the literature, which means that it is an active issue. In this section, some interesting examples have been exposed, but, in order to give more information and related articles about the problem we are dealing with, interested reader are referred to [20], [21], and [22] for different surveys about this issue.

2.2 Soft Computing techniques applied to imbalance datasets

Soft Computing techniques have been widely used since its presentation in 90's by Zadeh [23]. Machine Learning, Fuzzy Logic, and Evolutionary Computation methods are inside the vast group of Soft Computing techniques. Techniques such as GAs, SVM, Fuzzy Rule Based Systems, PSO and so on, have been developed and applied to different themes along the years, showing their good performance and the huge range of possibilities that they offer.

Regarding Fuzzy Logic techniques, fuzzy logic methods have been used in imbalance cases of study along the years. For example, in [24], a fuzzy technique is developed to predict heart diseases. The technique is divided in three phases: first, a fuzzy c-means clustering algorithm is used. Then, rules are generated from the rough set theory, and those rules are used for prediction with the fuzzy classifier.

Another case can be found in [25], where linguistic Fuzzy Rule Based Systems have been applied to imbalance datasets to deal with the overlapping problems between the concepts to be learned. This problem is more severe in imbalance datasets due to the most of the techniques try to correctly classify the majority class and, in cases of imbalance distribution of the data, it is the minority class where the most important data can be found. Datasets used are extracted from KEEL dataset repository.

Finally, authors of this study are aware of the huge amount of related papers that can be found in the literature. In this work, we have mentioned some of the most interesting research papers, in order to give an idea of the activity that is being carried out in the community. For further information, we recommend the reading of any of the review papers that can be found in the literature, such as [26], or [27]. In this work, fuzzy methods will be used to study their performance in a real imbalance scenario.

3 Techniques used for the comparative study

As mentioned in previous sections, one of the aims of this work is to study the performance of ensembles and fuzzy meta-heuristic techniques when they are applied to imbalanced problems. A total of 10 techniques are chosen, divided in two principal groups: six ensemble techniques, and four fuzzy ones. Due to the limited space, only the name of the techniques as well as a brief description of them are listed below:

- Ensemble techniques
  1. AdaBoost (I) [28] is an adaptation of general Adaboost for imbalance datasets.
  2. MSSMOTE Bagging [29] oversamples minority class instances using MSMOTE preprocessing algorithm. In this method both classes contribute to each bag with N instances.
  3. MSSMOTE Boosting [30] introduces synthetic instances in each iteration of AdaBoost technique, using the MSMOTE data preprocessing algorithm.
  4. RUSBoost [31] removes instances from the majority class by random undersampling the data-set in each iteration.
  5. SMOTE Bagging [32] oversamples minority class instances using SMOTE preprocessing algorithm.
  6. SMOTE Boosting [33] introduces synthetic instances in each iteration of AdaBoost technique, using the SMOTE data preprocessing algorithm.

All ensemble techniques used in this work have C4.5 algorithm as base classifier.

– Fuzzy Classification techniques
  1. AdaBoost (C) [34] is a boosting algorithm, which repeatedly invokes a learning algorithm to successively generate a committee of simple, low-quality classifiers.
  2. LogitBoost [35] is a backfitting algorithm, which repeatedly invokes a learning algorithm to successively generate a committee of simple, low-quality classifiers.
  3. FARCHD-C [36] mines fuzzy association rules limiting the order of the associations in order to obtain a reduced set of candidate rules with less attributes in the antecedent.
  4. C4.5 [37] is a decision tree generating algorithm that it induces classification rules in the form of decision trees from a set of given examples. C4.5 is based on ID3 algorithm.

It is important to remark that the different between $AdaBoost(C)$ and $AdaBoost(I)$ is the base classifier. While the first one counts with fuzzy classifiers, the second one uses a C4.5 algorithm as base classifier.

4 Experimentation

This section compiles the experimentation carried out in this work. Datasets used in this work as well as the information related to them are exposed in Section 4.1 while the results, and statistic methods applied are summarized in Section 4.2.

4.1 Datasets and preprocessing

Datasets used in this work contains real data from traffic cameras in the city of Helmond (The Netherlands). This data is provided by TASS international [1] and used in the Horizon 2020 project TIMON project[2] (Enhanced real time services for optimized multimodal mobility relying on cooperative networks and open data). Congestion in the road is used as class variable. In the raw data, this variable can take four different values: Normal, Increasing, Dense and Congestion. In order to simplify and make the problem equal to the techniques mentioned in the previous section, the classes have been reduced by two: Normal (majority class) and Congestion value (minority class), which includes Increasing, Dense and Congestion instances. Each dataset counts with a total of 22 variables, which includes not only information about the speed, the number of vehicles or the occupancy of the road, but the weather when data was taken. Data used in this work are collected during two months by four cameras, and divided in four different horizons of time (15, 30, 45 and 60 minutes respectively), which makes a total of 16 datasets.

[1] https://www.tassinternational.com/
[2] https://www.timon-project.eu/

Besides, data collected from Lisbon highway A5 used in EU project ICSI [1] have been also used. This highway is a 25 km long motorway in Portugal that connects Lisbon to Cascais. Data used in this work was collected from seven sensors displayed in the road and transformed into datasets. As well as in Helmond datasets, congestion in the road is taken as class variable. In this case, this class contains a value of congestion that appear in the next hour at a certain point and can take as values LOW, if the number of vehicles are below the percentile 15; MED (Medium), if the it is between percentiles 15 and 30; and $HIGH$ otherwise. Following the same logic applied to previous datasets, LOW and MED instances have been labeled as Normal (mayority class) while $HIGH$ instances have been changed to Congestion label (minority class). Data was collected during a month. The three first weeks are used as training data while the last week of the month is used to validate the solutions. These datasets are called BRISA datasets along the rest of the work.

Information about Imbalance Ratio (IR) and number of instances in each dataset are shown in Table 1

	Name of Dataset	N. Instances	IR
TASS datasets	C1	5333	8.1
	C2	5338	8.2
	C28	5348	8.16
	C47	5449	7
Brisa datasets	CL_{600}	721	2.04
	CL_{1980}	1441	2.26
	CL_{3600}	721	5.43
	CL_{4000}	1441	2.57
	CL_{6800}	721	2.13
	CL_{8050}	1441	2.25
	CL_{9400}	721	2.53

Table 1. Information about the datasets used in this work

4.2 Results

KEEL software [38] has been used to carry out the experiments. In the case we are dealing with, the module for imbalanced techniques are used. The experimentations have been executed in a Intel Xeon E5 2.30 GHz with a RAM memory of 32 GB. Related with the configuration of the techniques used in the experimentation, the default configuration given by KEEL has been retained. The Area Under the Curve (AUC) has been used as error metric. To show TASS dataset results, datasets are divided by id of the camera and horizon of time. Those results are shown in Table 2. Bold values represent the two best results obtained in each dataset. .

[1] http://www.ict-icsi.eu

Techniques	C1				C2				C28				C47			
	15	30	45	60	15	30	45	60	15	30	45	60	15	30	45	60
C4.5	**.968**	.958	.956	**.963**	**.970**	**.954**	**.955**	**.955**	**.963**	.949	.958	**.962**	**.958**	**.964**	.940	.953
FARCHD	.805	.727	.642	.623	.788	.729	.612	.575	.808	.723	.667	.500	.829	.732	.642	.622
AdaBoost(C)	.549	.563	.508	.500	.564	.614	.507	.501	.560	.540	.510	.500	.566	.541	.512	.501
LogitBoost	.672	.668	.585	.537	.703	.678	.594	.548	.659	.657	.567	.531	.685	.662	.604	.560
AdaBoost(I)	.939	.950	.957	.941	.951	.940	.950	.953	.952	.952	.938	.941	.950	.945	.930	.952
MSMOTEBagging	.872	.942	.943	.941	.886	.939	.932	.929	.903	.945	.947	.936	.912	.937	.926	.927
MSMOTEBoost	.916	.948	.935	.932	.936	.939	.925	.918	.939	.942	.940	.934	.935	.952	.934	.930
RUSBoost	**.976**	**.973**	**.971**	**.968**	**.973**	**.972**	**.967**	**.965**	**.972**	**.977**	**.968**	**.971**	**.971**	**.973**	**.968**	**.969**
SMOTEBagging	.916	.953	.936	.938	.923	.941	.934	.923	.937	.947	.943	.932	.893	.945	.926	.937
SMOTEBoost	.956	**.963**	**.960**	.954	.946	**.954**	.954	.946	.946	**.961**	**.959**	.955	.952	.960	**.958**	**.955**

Table 2. AUC values obtained for each technique in each dataset and horizon of time for TASS datasets

As it can be seen, three techniques stand out from the rest: RUSBoost, SMOTEBoost, and C4.5. In case of RUSBoost, it obtains one of the two best results in every dataset used, being the first one in each one of them. For SMOTEBoost, it gets one of the two best AUC values in 7 out of 16 datasets. Finally, for C4.5, it achieves a value between the best two in 10 out of 16 datasets, especially in C2 dataset. About the rest of the techniques, in general, ensemble techniques obtain better results than fuzzy ones. Focusing in the fuzzy techniques, though FARCHD and C4.5 achieves good performance in this problem without changing anything in its execution, AdaBoost(C) and LogitBoost do not obtain a considerable performance. In fact, AdaBoost(C) obtain the lowest AUC values in every dataset in comparison with the rest of techniques. If both AdaBoost techniques presented in this experimentation are compared, ensemble version of AdaBoost (AdaBoost(I)) outperforms the fuzzy one. On the other hand, taking into account ensemble techniques, RUSBoost outperforms the rest of them, followed by SMOTEBoost. However, all the techniques obtain a good performance in every dataset and horizon of time, which always achieve an AUC value higher than 0.9. About the horizon of time, the increasing of this value does not seem to affect to the performance of the techniques significantly. Only AdaBoost(C) and LogitBoost notice the change of this value. The rest of the techniques obtains almost the same performance when the horizon of time is 15 minutes than when it takes the value 60 minutes. Some of them (SMOTEBagging, C1 dataset) even improve its performance between these two horizons.

Table 2 contains the results obtained by each one of the techniques for each BRISA dataset. As in the previous results, the two best values are highlighted in bold.

The results show that MSMOTEBagging is the best technique so far in these datasets, obtaining 4 out of 7 best values, following by RUSBoost and SMOTEBagging, which both obtain 3 out of 7 best results. For the rest of the techniques, about fuzzy techniques used, only AdaBoost (C) and LogitBoost obtain bold values. Although their performance is not far from those obtained by the best techniques, they do not reach the high AUC value obtained by the rest of the techniques. Adaboost (C) and LogitBoost obtain one bold value, in dataset

	CL_{600}	CL_{1980}	CL_{3600}	CL_{4000}	CL_{6800}	CL_{8050}	CL_{9400}
C4.5	.893	.919	.898	.945	.872	.940	.875
FARCHD	.830	.955	.906	.928	.893	.951	.954
AdaBoost (C)	.882	.938	.808	.938	.864	.948	**.979**
LogitBoost	.853	.951	.891	.945	.884	.945	**.975**
AdaBoost(I)	.886	.954	.859	.941	.852	.957	.892
MSMOTE-Bagging	**.924**	**.961**	.928	.941	**.909**	**.962**	.867
MSMOTE-Boost	.884	.957	.899	.954	.881	**.965**	.871
RUSBoost	.902	.955	**.919**	**.955**	**.909**	.951	.896
SMOTEBagging	.914	**.958**	**.935**	**.958**	.901	.954	.875
SMOTEBoost	**.928**	.934	.915	.941	.897	.940	.921

Table 3. AUC values obtained for each technique in each Lisbon dataset

CL_{9400}, being the two best techniques in the mentioned dataset. Comparing the results obtained in the previous datasets, in this case, bagging techniques overpass boosting techniques, being RUSBoost the only one that can be compared with the results obtained by them.

In order to assess if the differences in performance among the techniques studied here are significantly different we employed non-parametric tests following the guidelines given by Garcia et al. in [39]. The procedure carried out is described next. We first apply Friedman's non-parametric test for multiple comparison at a significance level $\alpha \leq 0.05$ to assess if we can reject the null hypothesis of similar performance among all algorithms. If so, then we evaluate if the performance of the best algorithm according to Friedman's averaged ranking versus the other classifiers is significantly better. To this end, we apply Holm's [40] and Finner's [41] post-hoc tests at a significance level $\alpha \leq 0.05$ using the best method as control algorithm. Following this procedure, we analyse the performance of the algorithms globally over the two datasets.

We do the exercise of evaluating the performance of the methods over all datasets. According to Friedman's tests there exists significant differences among algorithms. The averaged ranking displayed in Table 4 confirm that RUSBoost is the most robust classifier followed by SMOTEBoost. On the contrary, the three fuzzy algorithms are clearly the ones that show a worse performance, whereas the result of the rest of algorithms is very similar. Using RUSBoost as control algorithm for the Holm's and Finner's post-hoc tests, we observe in Table 5 that, taking into account all datasets, it obtains significantly better AUC values that the other studied methods, excepting SMOTEBoost, although even in this case the significance level is quite near to the threshold, being equal to 0.07.

5 Conclusions

In this work, ensemble and fuzzy rules techniques have been applied to imbalance real traffic datasets in order to classify correctly the state of the road in a real scenario. In this case, data collected from cameras in the city of Helmond (The

Algorithm	Ranking
AdaBoost (I)	4.9783
C4.5	4.2391
FARCHD	7.5652
AdaBoost (C)	9.1739
LogitBoost	8.1087
MSMOTEBagging	5.2609
MSMOTEBoost	5.3913
RUSBoost	1.8043
SMOTEBagging	5.0652
SMOTEBoost	3.413

Table 4. Average Rankings of the algorithms provided by Friedman's non-parametric test for multiple comparisons over all datasets

Algorithm	Adjusted p-value Holm	Adjusted p-value Finner
AdaBoost (C)	0	0
LogitBoost	0	0
FARCHD	0	0
MSMOTEBoost	0.000353	0.000132
MSMOTEBagging	0.000541	0.000195
SMOTEBagging	0.001039	0.00039
AdaBoost (I)	0.001134	0.000486
C4.5	0.012778	0.007185
SMOTEBoost	0.07157	0.07157

Table 5. Adjusted p-value returned by Holm's and Finner's post-hoc tests for all datasets

Netherlands), and from A5 Highway in Lisbon are used. Data from cameras was collected by TASS international and used in H2020 TIMON project. In case of A5 highway, this data was used in ICSI project. The aim of this article is to compare the performance of ensemble and fuzzy techniques in imbalance real scenarios.

As results, in Helmond datasets, ensemble techniques outperform those fuzzy techniques used in the experimentation, with two techniques between the best ones. Three techniques stand out the rest: RUSBoost, SMOTEBoost, and C4.5. Among all, RUSBoost obtained at least one of the two best values in every dataset used. For SMOTEBoost and C4.5, they obtained 7 out of 16 and 10 out of 16 best values respectively. Regarding Lisbon datasets, ensemble techniques again, specially Bagging techniques and RUSBoost, obtain better performance than fuzzy techniques. All these results are checked using different statistical tests.

As future works, other techniques for both groups can be used. Besides, the experimentation could be applied to more datasets and other horizons of time. Regarding this, one future work to take into account is to adapt ensemble techniques to work with multiclass classification. This will increase the difficulty of the problem as well as the IR of each dataset, making the data a good real benchmark to use in comparatives like the presented in this paper.

Acknowledgements

This work has been supported by TIMON project (Enhanced real time services for optimized multimodal mobility relying on cooperative networks and open data) which received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 636220. E. Osaba performed his contribution when he was working on the University of Deusto.

Bibliography

[1] López, V., Fernández, A., Moreno-Torres, J.G., Herrera, F.: Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. Expert Systems with Applications **39** (2012) 6585 – 6608

[2] Sardari, S., Eftekhari, M.: A fuzzy decision tree approach for imbalanced data classification. (2016) 292–297

[3] Savetratanakaree, K., Sookhanaphibarn, K., Intakosum, S., Thawonmas, R.: Borderline over-sampling in feature space for learning algorithms in imbalanced data environments. IAENG International Journal of Computer Science **43** (2016) 363–373

[4] Lopez-Garcia, P., Onieva, E., Osaba, E., Masegosa, A.D., Perallos, A.: A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy. IEEE Transactions on Intelligent Transportation Systems **17** (2016) 557–569

[5] Guo, L., Ge, P.S., Zhang, M.H., Li, L.H., Zhao, Y.B.: Pedestrian detection for intelligent transportation systems combining adaboost algorithm and support vector machine. Expert Systems with Applications **39** (2012) 4274–4286

[6] Cervantes, J., Li, X., Yu, W.: Imbalanced data classification via support vector machines and genetic algorithms. Connection Science **26** (2014) 335–348

[7] Xu, Z., Watada, J., Wu, M., Ibrahim, Z., Khalid, M.: Solving the imbalanced data classification problem with the particle swarm optimization based support vector machine. IEEJ Transactions on Electronics, Information and Systems **134** (2014) 788–795

[8] Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine learning **36** (1999) 105–139

[9] Fang, Y., Fu, Y., Sun, C., Zhou, J.: Improved boosting algorithm using combined weak classifiers. Journal of Computational Information Systems **7** (2011) 1455–1462

[10] Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review **33** (2010) 1–39

[11] Nama, S., Saha, A.: An ensemble symbiosis organisms search algorithm and its application to real world problems. Decision Science Letters **7** (2018) 103–118

[12] Zhao, Z., Liu, Y., Li, J., Wang, J., Wang, X.: A study of fuzzy clustering ensemble algorithm focusing on medical data analysis. Lecture Notes in Electrical Engineering **422** (2018) 383–396

[13] Pescaru, D., Curiac, D.I.: Ensemble based traffic light control for city zones using a reduced number of sensors. Transportation Research Part C: Emerging Technologies **46** (2014) 261–273

[14] Lim, P., Goh, C., Tan, K.: Evolutionary cluster-based synthetic oversampling ensemble (eco-ensemble) for imbalance learning. IEEE Transactions on Cybernetics **47** (2017) 2850–2861

[15] Kotsiantis, S.B.: Bagging and boosting variants for handling classifications problems: a survey. The Knowledge Engineering Review **29** (2014) 78–100
[16] Breiman, L.: Bagging predictors. Machine learning **24** (1996) 123–140
[17] Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. **96** (1996) 148–156
[18] Del Jesus, M., Hoffmann, F., Navascués, L., Sánchez, L.: Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Transactions on Fuzzy Systems **12** (2004) 296–308
[19] Lango, M., Stefanowski, J.: Multi-class and feature selection extensions of roughly balanced bagging for imbalanced data. Journal of Intelligent Information Systems (2017) 1–31
[20] Jurek, A., Bi, Y., Wu, S., Nugent, C.: A survey of commonly used ensemble-based classification techniques. Knowledge Engineering Review (2013)
[21] Mokeddem, D., Belbachir, H.: A survey of distributed classification based ensemble data mining methods. Journal of Applied Sciences (2009)
[22] Wang, S., Yao, X.: Multiclass imbalance problems: Analysis and potential solutions. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics (2012)
[23] Zadeh, L.A.: Fuzzy logic, neural networks, and soft computing. Communications of the ACM **37** (1994) 77–85
[24] Antonelli, M., Ducange, P., Marcelloni, F.: An experimental study on evolutionary fuzzy classifiers designed for managing imbalanced datasets. Neurocomputing **146** (2014) 125–136
[25] Harandi, F., Derhami, V.: A reinforcement learning algorithm for adjusting antecedent parameters and weights of fuzzy rules in a fuzzy classifier. Journal of Intelligent and Fuzzy Systems **30** (2016) 2339–2347
[26] Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al.: Handling imbalanced datasets: A review. GESTS International Transactions on Computer Science and Engineering **30** (2006) 25–36
[27] Ramyachitra, D., Manikandan, P.: Imbalanced dataset classification and solutions: a review. International Journal of Computing and Business Research (IJCBR) **5** (2014)
[28] Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. (1995) 23–37
[29] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42** (2012) 463–484
[30] Hu, S., Liang, Y., Ma, L., He, Y.: Msmote: improving classification performance when training data is imbalanced. In: Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on. Volume 2., IEEE (2009) 13–17
[31] Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **40** (2010) 185–197

[32] Wang, S., Yao, X.: Diversity analysis on imbalanced data sets by using ensemble models. In: Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on, IEEE (2009) 324–331
[33] Chawla, N., Lazarevic, A., Hall, L., Bowyer, K.: Smoteboost: Improving prediction of the minority class in boosting. Knowledge Discovery in Databases: PKDD 2003 (2003) 107–119
[34] Del Jesus, M.J., Hoffmann, F., Navascués, L.J., Sánchez, L.: Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Transactions on Fuzzy Systems **12** (2004) 296–308
[35] Otero, J., Sánchez, L.: Induction of descriptive fuzzy classifiers with the logitboost algorithm. Soft Computing-A Fusion of Foundations, Methodologies and Applications **10** (2006) 825–835
[36] Alcala-Fdez, J., Alcala, R., Herrera, F.: A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. IEEE Transactions on Fuzzy Systems **19** (2011) 857–872
[37] Quinlan, J.R.: C4. 5: Programming for machine learning. Morgan Kauffmann **38** (1993)
[38] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic & Soft Computing **17** (2011)
[39] García, S., Herrera, F., Shawe-taylor, J.: An extension on —statistical comparisons of classifiers over multiple data sets‖ for all pairwise comparisons. Journal of Machine Learning Research (2008) 2677–2694
[40] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics (1979) 65–70
[41] Finner, H.: On a monotonicity problem in step-down multiple test procedures. Journal of the American Statistical Association **88** (1993) 920–923

Scrum Task Allocation Based on Particle Swarm Optimization

Lucija Brezočnik*[0000−0002−3622−428X], Iztok Fister Jr.[0000−0002−6418−1272], and Vili Podgorelec[0000−0001−6955−7868]

Information Systems Laboratory, Institute of Informatics,
Faculty of Electrical Engineering and Computer Science, University of Maribor,
Koroška cesta 46, SI-2000 Maribor, Slovenia
*Corresponding author: lucija.brezocnik@um.si

Abstract. In this paper, we present a novel algorithm called STAPSO, which comprises Scrum task allocation and the Particle Swarm Optimization algorithm. The proposed algorithm aims to address one of the most significant problems in the agile software development, i.e., iteration planning. The actuality of the topic is not questionable, since nowadays, agile software development plays a vital role in most of the organizations around the world. Despite many agile software development methodologies, we include the proposed algorithm in Scrum Sprint planning, as it is the most widely used methodology. The proposed algorithm was also tested on a real-world dataset, and the experiment shows promising results.

Keywords: Agile Software Development, Particle Swarm Optimization, Scrum, Software Engineering, Task Allocation

1 Introduction

The idea of the iterative and agile development is all but new [1]. Ongoing changing priorities, desire to accelerate product delivery, the increase of productivity, improvement of project visibility, and enhancing software quality [2] are the top five reasons for adopting agile. Furthermore, in the report from Gartner Inc. [3], which is the world's leading research and advisory company, it is evident that the traditional project and development methods, e.g., waterfall, are evermore unsuitable [4, 5]. Consequently, we can state that agile software development is, nowadays, not a competitive advantage anymore, but rather the need for the organizations to survive on the market.

Regardless of the chosen agile method, e.g., Scrum, Kanban, Scrumban, XP (extreme programming), and Lean, monitoring of its performance must be carried out. Success in agile projects is most often measured by velocity in 67%, followed by the iteration burndown (51%), release burndown (38%), planned vs. actual stories per iteration (37%), and Burn-up chart (34%) [2]. However, a prerequisite for a successful monitoring of the progress is undoubtedly precise

iteration planning. The latter is not only the number one employed agile technique in the organizations [2], but also one of the hardest tasks, as is evident from many scientific papers [6, 7] and interviews conducted with different CIOs (Chief Information Officers). Also, each task defined in a given iteration must be estimated precisely. The estimation can be conducted with various techniques [8, 2], e.g., number sizing (1, 2, ..., 10), Fibonacci sequence (1, 2, 3, 5, 8, ...), and T-shirt sizes (XS, S, M, L, XL, XXL or XXXL). However, we must not forget about dependencies between tasks which result in the implementation order.

Thus, from an apparently simple problem arises a considerable optimization problem that is dealt with daily in organizations all around the world. When dealing with numerous dependencies and tasks, solving a problem by hand becomes very hard. On the contrary, we propose a systematical solution that is based on nature-inspired algorithms. Nature-inspired algorithms are a modern tool for solving hard continuous and discrete problems. They draw inspiration for solving such problems from nature. Until recently, more than 100 nature-inspired algorithms have been proposed in the literature [9], where Particle Swarm Optimization (PSO) [10] is one of the oldest and well-established nature-inspired algorithms. Many studies have proved theoretically and practically that PSO is a very simple, as well as efficient algorithm [11, 12] appropriate even for real-world applications [13].

In this paper, we show the modifications of the basic PSO algorithm that is applied to the problem of Scrum task allocation. The new algorithm, called STAPSO, is developed, implemented, and tested on a real dataset.

We believe that this is the first work that deals with the problem of Scrum task allocation in the optimization domain. Altogether, the purpose of this paper is to:

- represent Scrum task allocation as an optimization problem,
- propose the Particle Swarm Optimization algorithm for solving Scrum task allocation, or simply STAPSO, and
- test the proposed algorithm on a real dataset.

The structure of this paper is as follows: Section 2 outlines the fundamentals of Scrum, while Section 3 describes the fundamentals of the PSO algorithm, together with STAPSO algorithm. Section 4 presents the design of the experiment, along with the results in Section 5. The paper concludes with a summary of the performed work and future challenges.

2 Scrum

Scrum is the most used agile methodology, with 58% share of the market [2] and is by definition "a framework for developing, delivering, and sustaining complex products" [14, 15]. It consists of three primary roles, i.e. the Scrum Master, the Product Owner, and the Development Team. In the organizations, the Scrum Master is responsible for Scrum promotion and offers support regarding Scrum theory, values, and practices. Product Owner is a focal role since it is connected

with the development team and the stakeholders. Two of his/her primary goals are to maximize the value of the product, and definition of the user stories from the product backlog. The remaining role, i.e., the Development Team, is liable for product increment delivery at the end of each Sprint. The Development Team is cross-functional and self-organizing, meaning that the people in it have all the skills required to deliver the product successfully.

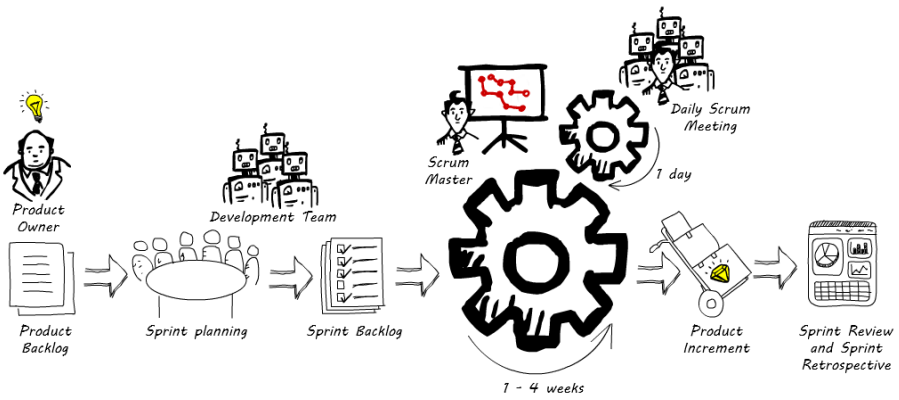


Fig. 1. The Scrum framework.

In Scrum, the process starts with the Product Owners' definition of the product backlog, which is a prioritized list of user stories (see Fig. 1). Afterwards, Sprint Planning starts. At this meeting, the team decides which user stories from the Product Backlog will be carried out in the upcoming Sprint (because the Product Backlog is prioritized, they pull user stories from the top of the list). The newly created document is called a Sprint Backlog and contains an in-depth description of the chosen user stories. After that, everything is ready for the beginning of the Sprint, that usually lasts between one and four weeks. Each day of the Sprint starts with a brief daily Scrum (short stand-up meeting) at which the Development Team exchanges opinions regarding the previous day and highlights possible problems. At the end of each Sprint, Sprint Review and Sprint Retrospective are carried out by the Development Team and the Product Owner, with the intention to find the potential for improvement.

For the calculation of the optimal line, it is necessary to determine the duration of the Sprint first (n_days). For example, if we decide on the two week long Sprints, the Development Team actually has ten working days, assuming Saturday and Sunday are free days. After that, based on tasks from the Sprint Backlog, the total estimated effort (t_effort) is obtained as their sum. Optimum per day (opt_d) can now be calculated by Eq. 1.

$$opt\_d = \frac{t\_effort}{n\_days} \tag{1}$$

Ideal or optimal line (*Oline*) is derived from linear function $f(x) = ax + b$ and is presented by Eq. 2, where x denotes the specific day of the Sprint.

$$Oline = -\frac{t\_effort}{n\_days} * x + t\_effort \tag{2}$$

Current line (*Cline*) is calculated by Eq. 3, where *Edone* denotes the summarized effort of the tasks per given day.

$$Cline = Oline(x) - (Oline(x-1) - Edone) \tag{3}$$

3 Particle Swarm Optimization for Scrum task allocation

In this Section, the proposed algorithm called STAPSO is described in detail. Since the algorithm is based on the PSO, its explanation is divided into two Subsections. Subsection 3.1 depicts the fundamentals of the PSO, and Subsection 3.2 presents the proposed algorithm in detail.

3.1 Fundamentals of PSO

The PSO algorithm [10] preserves a population of solutions, where each solution is represented as a real-valued vector $\mathbf{x} = (x_{i,1}, \ldots, q_{i,D})^T$ for $i = 1, \ldots, Np$ and $j = 1, \ldots, D$, and the parameter Np denotes the population size, and the parameter D dimension of the problem. This algorithm explores the new solutions by moving the particles throughout the search space in the direction of the current best solution. In addition to the current population $\mathbf{x}_i^{(t)}$ for $i = 1, \ldots, Np$, also the local best solutions $\mathbf{p}_i^{(t)}$ for $i = 1, \ldots, Np$ are preserved, denoting the best i-th solution found. Finally, the best solution in the population $\mathbf{g}^{(t)}$ is determined in each generation. The new particle position is generated according to Eq. (4):

$$\begin{aligned} \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^{(t)} + C_1 U(0,1)(\mathbf{p}_i^{(t)} - \mathbf{x}_i^{(t)}) + C_2 U(0,1)(\mathbf{g}^{(t)} - \mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \end{aligned} \tag{4}$$

where $U(0,1)$ denotes a random value in interval $[0,1]$, and C_1 and C_2 are learning factors. Algorithm 1 depicts the original PSO algorithm.

Interestingly, many surveys have recently revealed that the PSO algorithm was used in numerous real-world applications [13, 16]. However, the presence of the PSO algorithm in the software engineering research area is still in the minority.

In the next Subsection, the proposed STAPSO algorithm is presented for the Scrum task allocation problem.

Algorithm 1 Pseudocode of the basic PSO algorithm

Input: PSO population of particles $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, *MAX_FEs*.
Output: The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.

```
init_particles;
eval = 0;
while termination_condition_not_meet do
  for i = 1 to Np do
    f_i = evaluate_the_new_solution(x_i);
    eval = eval + 1;
    if f_i ≤ pBest_i then
      p_i = x_i; pBest_i = f_i; // save the local best solution
    end if
    if f_i ≤ f_min then
      x_best = x_i; f_min = f_i; // save the global best solution
    end if
    x_i = generate_new_solution(x_i);
  end for
end while
```

3.2 STAPSO algorithm

The following Section depicts the process of a Scrum task allocation problem using the STAPSO algorithm. For this problem, the following three modifications were applied to the basic PSO algorithm:

- representation of individuals,
- design of fitness function, and
- constraint handling.

Representation of individuals Candidate solutions in the basic PSO algorithm are represented as real-valued vectors $\mathbf{x}$, whilst a Scrum task allocation problem demands an integer vector $\mathbf{y}$ symbolizing the effort of a particular task. For that reason, mapping between representation of solutions in real-valued search space to the solution in a problem space is needed. In a STAPSO, this mapping is conducted in a similar manner as it was done for the problem of sport training sessions' planning [17]. A candidate solution in the proposed STAPSO algorithm is also represented, using the real-valued vector $\mathbf{x}_i = \{x_i0, \ldots, x_in\}^T$ for $i = 1 \ldots n$ with elements $x_{ij} \in [0, 1]$. In order to obtain effort values for fitness function calculation, firstly the permutation of task effort $\pi_i = \{\pi_{i1}, \ldots, \pi_{in}\}$ is mapped from the vector $\mathbf{x}_i$ such that the following relation is valid:

$$x_{i\pi_{i0}} < x_{i\pi_{i1}} < \ldots < x_{i\pi_{in}}. \tag{5}$$

Vector $y_i = \{y_{i0}, \ldots, y_{in}\}^T$ is determined from task description, Table 1. Table 2 presents an example of mapping the candidate solution $\mathbf{x}_i$ via permutation of task effort π_i to the final task allocation.

Table 1. Task description table (example)

Task_ID	Effort
0	3
1	2
2	4
3	3
4	5

Table 2. Candidate solution mapping

	Dimension j				
Elements i	0	1	2	3	4
Candidate solution $\mathbf{x}_i$	0.70	0.42	0.21	0.94	0.52
Permutation π_i	3	1	0	4	2
Task allocation $\mathbf{y}_i$	3	2	3	5	4

Fitness function Fitness function is calculated according to Eq. 6 as follows:

$$f(x) = |\sum_{j=0}^{n\_days} (calculated\_effort\_per\_day_j)| \tag{6}$$

where n_days denotes the number of days, and $calculated\_effort\_per\_day$ is calculated effort for every day according to the constraint:

$$\begin{aligned} &\forall d \in \{1, 2, \ldots, n\_days\}, \forall t \in \{1, 2, \ldots, n\_tasks(d)\}, \\ &\sum_{i=1}^{t} effort(i) \leq opt\_d \end{aligned} \tag{7}$$

where the variables d and t denote the current day of the Sprint, and the number of tasks per day, respectively. Final effort per day is then calculated as the sum of the tasks' efforts, that should not exceed the value of the opt_d (Eq. 1).

Constraint handling As discussed in previous Sections, there is sometimes a particular order (dependency) of some tasks. In other words, it means that one task must be completed before the beginning of another task. Most candidate solutions that are obtained according to mapping in Table 2 are unfeasible, i.e., they violate the dependency conditions. In our case, unfeasible solutions are penalized. Algorithm 2 presents our solution for handling constraints, where the function $is\_violated()$ checks if the dependency condition is violated. If the dependency condition is violated, the algorithm assigns a very high penalty [18] value to this particle. Despite many constraint handling methods, our penalization method behaves very well on the current problem. For that reason, we have not tested the behavior of any other constraint handling methods yet [19].

Algorithm 2 Constraint handling in STAPSO

```
violations = 0;
fitness = f(x);{calculated by Eq. 6}
for i = 1 to Num_Rules do
    if is_violated() then
        violations = violations + 1;
    end if
end for
if violations > 0 then
    fitness = violations * 1000;
end if
```

4 Experiment

The experimental approach was used in order to show the power of the STAPSO algorithm. Thus, Subsection 4.1 comprises parameter settings of the algorithm and the computer environment, and Subsection 4.2 presents test data, along with the constraints on Scrum tasks that were used in this study.

4.1 Setup

Experiments were conducted on an Intel XEON Z240 computer. STAPSO is implemented in the Python programming language without using any special software libraries. The algorithm ran on a Linux Ubuntu 16.04 operating system. After the extensive parameter tuning, the following parameter settings were used based on their best performance:

- population size Np: 75,
- dimension of the problem D: 60,
- number of function evaluations per run $MAX\_FEs = 30000$,
- total runs: 25,
- cognitive component $C_1 = 2.0$,
- social component $C_2 = 2.0$,
- velocity: [-4, 4],
- number of days: 10,
- number of Sprint: 1.

4.2 Test data and constraints

Table 4 presents test data that were used in this study. Test data for such experiments is very hard to get due to the company policy of confidential data. Thus, the source of test data is an internal project that was conducted within our laboratory. In Table 4, $Task\_ID$ denotes the identification number of a particular task, while *Effort* symbolizes the effort of this task. In this study, the following constraints were considered:

$$\Psi = \{(T7, T3), (T6, T22), (T4, T58), (T33, T31)\}.$$

Thereby, Ψ denotes the implementation order of the tasks, i.e., task $T7$ must be implemented before task $T3$, task $T6$ must be implemented before task $T22$, etc. In the context of the algorithm, this means that all of the possible solutions must obey all of the given constraints and provide legitimate task allocation also considering Eq. 2 and Eq. 3.

5 Results

In the total of 25 runs, an optimal solution was found 20 times (i.e. success rate: 80%), meaning that no tasks were left behind for the next Sprint. In three cases (12%), the algorithm did not allocate one task with the estimated effort of 1, and in two cases (8%), the algorithm did not allocate one task estimated with the effort of 2. We want to highlight that all constraints presented in Subsection 4.2 were satisfied in all runs. On average, an optimal solution was found after 5533 function evaluations.

Table 3 comprises an in-depth description of one optimal solution. The latter presents the sequence of tasks' implementation for one Sprint, which is described with the Task IDs (column 2) and belonging tasks' effort (column 3). Per each day, the number of tasks and remaining effort is recorded, respectively.

Table 3. Example of task allocation from Table 4 (optimal solution)

Day	**Tasks allocated**	**Tasks' effort**	**Number of tasks**	**Effort remaining**
1	4, 12, 15, 17, 21, 32, 42	5, 3, 1, 1, 1, 2, 2	7	0
2	43, 27, 49, 48, 58, 33	3, 3, 3, 1, 1, 4	6	0
3	51, 7, 50, 5	1, 5, 4, 5	4	0
4	24, 26, 45, 35, 57, 54, 25	2, 1, 2, 2, 4, 2, 2	7	0
5	18, 10, 29, 16	2, 5, 3, 5	4	0
6	6, 22, 8, 53, 31	5, 4, 3, 1, 2	5	0
7	28, 44, 19, 0, 30, 3	4, 2, 1, 2, 4, 2	6	0
8	1, 14, 20, 37, 40, 52, 23, 38	2, 2, 2, 1, 1, 3, 3, 1	8	0
9	56, 34, 41, 11, 2, 9, 13	2, 3, 1, 2, 1, 3, 3	7	0
10	36, 39, 46, 47, 55, 59	3, 2, 4, 2, 2, 2	6	0
Σ	**60**	**150**	**60**	**0**

Fig. 2 and Fig. 3 present the same proposed solution of the STAPSO algorithm, where two tasks with the estimated effort of 1 were not allocated. A non-optimal solution was chosen deliberately for easier explanation of the results and deviations. Fig. 2 presents the solution in the form of the burndown chart, and Fig. 3 shows allocated tasks per day of the Sprint.

In Scrum, a burndown chart is one of the most frequently used graphs to present the current state of the work of the project [2, 20]. On the abscissa axis (Fig. 2), days of the Sprint are displayed, and on the ordinate axis, the remaining

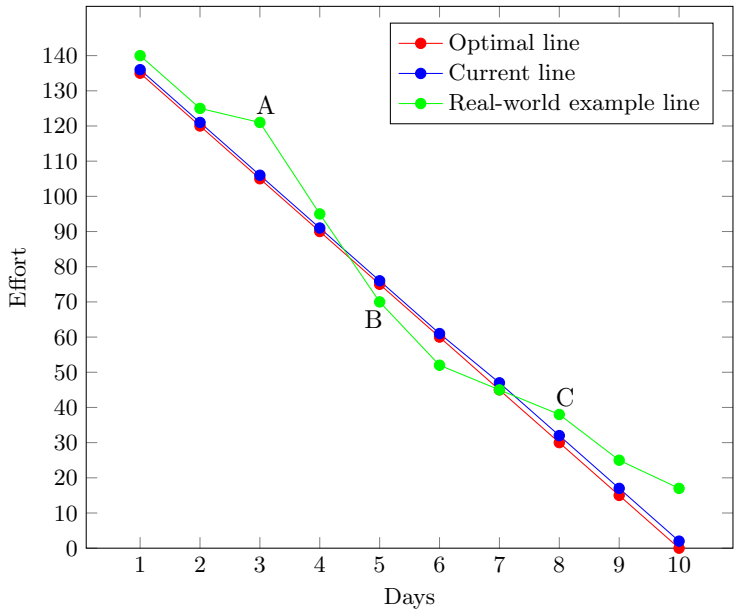


Fig. 2. Burndown chart of non-optimal solution (chosen deliberately for easier explanation of the results)

effort of the Sprint. The preparation of such a graph is carried out in several stages. Firstly, the optimal line is drawn. The optimal line is calculated with Eq. 2 and shows an ideal or optimal course of the implementation of the tasks. In Fig. 2, this line is colored with red. As stated before, all tasks are estimated with effort (see Table 4) and with their fulfillment, we can monitor remaining effort in a Sprint. If we look at the first day of the Sprint in Fig. 2, we can see that ideal effort completed per day is 15 (calculated with Eq. 1). Thus, the Development Team should, on their first day, complete tasks with the sum of the effort of at least 15. As we can see from Fig. 3, algorithm STAPSO for the first day allocated 5 tasks with the estimated effort sum of 14, meaning that, after the first day, the Development Team is one effort behind the optimal line (see blue line). In a real-world scenario, we can witness lines that are similar to the green line. From the latter, it is evident that the Development Team was behind the optimal line for the first four days, and on day 3 (point A) they fulfilled tasks with the effort sum of only 4. However, after the third day, the fulfillment of tasks went very quickly, so in two days they caught up and were in front of the optimal line on day five (point B). Point C shows that the progress has stopped on day 8 (they were behind the optimal line again), and they stayed behind it until the end of the Sprint.

In Fig. 3 the days of the Sprint show the allocated tasks given by the STAPSO algorithm. As we have said in the description of Fig. 2, the optimal effort sum

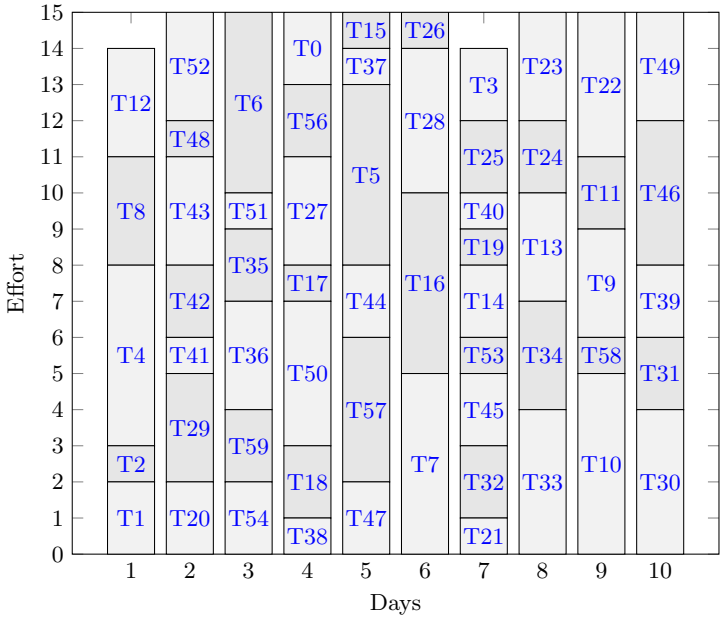


Fig. 3. Task allocation of non-optimal solution (chosen deliberately for easier explanation of the results)

per day is 15 (maximum value of the ordinate axis). This sum is also the value that the algorithm is trying to achieve per day. If we look at the results, on the first day the STAPSO algorithm allocated five tasks, i.e., $T1$, $T2$, $T4$, $T8$, and $T12$ (see Table 4), with the sum of effort of 14. On the second day, a sum of effort of 15 is fulfilled with the tasks $T20$, $T29$, $T41$, $T42$, $T43$, $T48$, and $T52$, etc. This kind of graph is beneficial for the Development Team and the Product Owner, since they have allocated tasks from the beginning of the Sprint.

6 Conclusion

A novel algorithm STAPSO was implemented and tested successfully on a real dataset. It offers a solution to the global problem of task allocation in the agile software development. The STAPSO algorithm can be applied to all of the known estimation techniques, e.g. number sizing, Fibonacci sequence, and T-shirt planning. Furthermore, it can be included in companies regardless of their size and maturity degree.

In the design of the algorithm, there is still significant room for improvement. In the future, we intend to study the impact of various constraint handling methods and variants of PSO on the behavior of the STAPSO algorithm. Hybridization of STAPSO with any other well-established algorithms, e.g., Differential Evolution is also a sparkling way for future research.

Since we have not found any similar algorithms for Scrum task allocation that are based on nature-inspired algorithms yet, we believe that this study could be a stepping stone for more links between the vibrant agile software development research area and optimization.

Acknowledgment

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

References

1. Takeuchi, H., Nonaka, I.: The New New Product Development Game. Harvard Business Review **64** (1986) 137–146
2. VersionOne: VersionOne 11th Annual State of Agile Report. (2017)
3. Kyte, A., Norton, D., Wilson, N.: Ten Things the CIO Needs to Know About Agile Development. Technical report, Gartner, Inc. (2014)
4. Gandomani, T.J., Nafchi, M.Z.: Agile transition and adoption human-related challenges and issues: A grounded theory approach. Computers in Human Behavior **62** (2016) 257–266
5. Chen, R.R., Ravichandar, R., Proctor, D.: Managing the transition to the new agile business and product development model: Lessons from cisco systems. Business Horizons **59**(6) (2016) 635–644
6. Heikkil, V.T., Paasivaara, M., Rautiainen, K., Lassenius, C., Toivola, T., Jrvinen, J.: Operational release planning in large-scale scrum with multiple stakeholders a longitudinal case study at f-secure corporation. Information and Software Technology **57** (2015) 116–140
7. Barney, S., Ke Aurum, A., Wohlin, C.: A product management challenge: Creating software product value through requirements selection. Journal of Systems Architecture **54** (2008)
8. Usman, M., Mendes, E., Weidt, F., Britto, R.: Effort estimation in agile software development: A systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering. PROMISE '14, NY, USA, ACM (2014) 82–91
9. Fister Jr., I., Yang, X.S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. Elektrotehniški vestnik **80**(3) (2013) 116–122
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. Volume 4., IEEE (1995) 1942–1948
11. Shi, Y., et al.: Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 Congress on evolutionary computation. Volume 1., IEEE (2001) 81–86
12. Yang, X.S.: Nature-inspired metaheuristic algorithms. Luniver press (2010)
13. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical Problems in Engineering (2015)
14. Sutherland, J.V., Sutherland, J.J.: Scrum: the art of doing twice the work in half the time. 1st edn. Currency (2014)

15. Schwaber, K., Sutherland, J.: The Scrum Guide. (2017)
16. Pluhacek, M., Senkerik, R., Viktorin, A., Kadavy, T., Zelinka, I.: A review of real-world applications of particle swarm optimization algorithm. In: International Conference on Advanced Engineering Theory and Applications, Springer (2017) 115–122
17. Fister, I., Rauter, S., Yang, X.S., Ljubič, K., Fister Jr., I.: Planning the sports training sessions with the bat algorithm. Neurocomputing **149** (2015) 993–1002
18. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer methods in applied mechanics and engineering **191**(11) (2002) 1245–1287
19. Mezura-Montes, E., Coello, C.A.C.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm and Evolutionary Computation **1**(4) (2011) 173–194
20. Cooper, R.G., Sommer, A.F.: Agile-stage-gate: New idea-to-launch method for manufactured new products is faster, more responsive. Industrial Marketing Management **59** (2016) 167–180

Test data

Table 4. Test data

Task_ID	Effort	Task_ID	Effort
T0	2	T30	4
T1	2	T31	2
T2	1	T32	2
T3	2	T33	4
T4	5	T34	3
T5	5	T35	2
T6	5	T36	3
T7	5	T37	1
T8	3	T38	1
T9	3	T39	2
T10	5	T40	1
T11	2	T41	1
T12	3	T42	2
T13	3	T43	3
T14	2	T44	2
T15	1	T45	2
T16	5	T46	4
T17	1	T47	2
T18	2	T48	1
T19	1	T49	3
T20	2	T50	4
T21	1	T51	1
T22	4	T52	3
T23	3	T53	1
T24	2	T54	2
T25	2	T55	2
T26	1	T56	2
T27	3	T57	4
T28	4	T58	1
T29	3	T59	2

Surrogate model based optimization of constrained mixed variable problems

Julien Pelamatti[1,3], Loïc Brevault[1], Mathieu Balesdent[1], El-Ghazali Talbi[2], and Yannick Guerin[3]

[1] ONERA - The French Aerospace Lab, 91120 Palaiseau, France
[2] INRIA Lille - Nord Europe, 59650 Villeneuve d'Ascq, France
[3] CNES - Direction des Lanceurs Ariane, 75612 Paris, France

Abstract. Surrogate model based optimization is an increasingly popular tool for engineering design as it enables to optimize the performance of complex systems at a limited computational cost. The working principle of this approach consists in creating a surrogate model of the fitness and constraint functions by relying on a limited amount of data, which is then refined by adding data samples in the area of interest of the search space determined according to a given criterion. Several techniques exist for the surrogate model based optimization of continuous functions, however, only few methods have been developed for mixed continuous/discrete problems. In this paper, two different adaptations of the Efficient Global Optimization algorithm for mixed continuous/discrete problems are presented and their performance are tested on different constrained analytical test-cases and aerospace engineering design problems.

Keywords: Mixed continuous/discrete variables, Surrogate model based optimization, Constrained optimization

1 Introduction

Within the framework of system design, an optimization problem can involve continuous and discrete decision variables. A common example of such a design problem is the preliminary design of launch vehicles, which involves discrete architectural and technological choices (*e.g.*, type of propulsion, number of stages) as well as continuous variables (*e.g.,* sizing parameters). This type of design problem is usually subject to a number of constraints (*e.g.,* structural integrity, target altitude and speed) which must be taken into account in order to obtain a feasible design. Due to the possibly large computation time required to evaluate the performance of the system, it is necessary to rely on optimization techniques that require as few function evaluations as possible to converge towards the optimum, such as the Efficient Global Optimization (EGO) algorithm [2], which relies on Gaussian process surrogate models [4]. The optimization is performed through a refinement of the meta-models according to an infill criterion which takes into account the predicted value and position of the problem optimum as well as the uncertainty associated to the prediction. The original version of EGO

was developed exclusively for functions depending on continuous variables and applying it to a mixed variables problem would therefore require to separately model each discrete category of the considered problem [3]. In this paper, two adaptations of EGO for the optimization of problems depending simultaneously on continuous and discrete variables are presented. Following this introduction, the used Gaussian process surrogate modeling techniques for mixed variables functions are described. Subsequently, the adaptation of EGO to the presence of constraints and discrete variables is discussed. The performance of this novel optimization algorithm is then tested on a number of constrained analytical test-cases and aerospace engineering design problems. Finally, the conclusions that may be drawn from the obtained results are presented.

2 Mixed continuous/discrete EGO

EGO is an optimization algorithm based on Gaussian process surrogate models which map the probability distribution of possible regression functions as a function of the co-variance between the data samples [4]. In this paper, the co-variance is defined as a function of a distance based spatial correlation between data samples. The first adaptation of Gaussian processes to mixed continuous/discrete functions that is considered was proposed (with a slightly different implementation) by M. Halstrup [1] and is based on an alternative definition of distance between samples in the continuous/discrete search space known as the Gower distance. In the Gower distance, the coordinates of two samples that are being considered are compared dimension-wise. For the continuous dimensions, the distance is proportional to the Manhattan distance, while for the discrete variables the distance is a weighted binary value, depending on the similarity between the variable values. This allows to rely on common distance based kernels in order to determine the correlation between samples in the mixed search space. In this paper, the the p-exponential correlation [4] is used. The second adaptation of Gaussian processes to mixed continuous/discrete functions that is presented in this paper is based on the idea of defining the correlation function as a product of two separate terms, each one representing the influence of either the continuous or the discrete variables of two considered data samples. Such a correlation function is proposed by Q. Zhou et al. [6], as the product between a distance based correlation function computed on the continuous variables and a parameter characterizing the correlation between the discrete coordinates of the two data samples. In order to ensure that the correlation matrix is valid and invertible, the hyper-parameters characterizing the discrete term of the correlation are computed as the elements of a matrix obtained through a so-called hypersphere decomposition. Having adapted the definition of the correlation function between samples in the search space to the presence of discrete variables, the EGO algorithm can subsequently be applied by including the discrete variables the considered function depends on in the set of parameters to be optimized in order to determine the optimal data sample infill criterion, thus allowing to optimize mixed continuous/discrete functions.

In order to take into account the constraints the problem is subject to, in this paper the considered function and its constraints are modeled separately. Subsequently, data samples are added to refine both surrogate models. In order to achieve this, the infill criterion is re-defined as the product between the EGO infill criterion and the probability that the considered sample complies with the given constraints, as described by M. Schonlau *et al.* [5]. This infill criterion is adapted to the mixed continuous/discrete case by including the discrete variables the considered function and the constraints depend on within the set of optimization parameters. In this paper, this novel optimization algorithm is tested on a number standard analytical test-cases as well as on launch vehicle design related engineering problems such as the minimization of a thrust frame mass and the minimization of a launch vehicle lift-off mass. The obtained results are then compared with standard mixed-variable optimization techniques such as the Genetic Algorithm (GA).

3 Conclusions

From the results obtained on analytical test-cases and aerospace design problems, it is shown that the adaptation of the EGO algorithm for mixed variable problems tends to converge to the considered problem optimum with considerably fewer function evaluations when compared to standard algorithms. Depending on the problem, the reduction of required function evaluations ranges from one to two orders of magnitude. This novel optimization algorithm results therefore promising for the design of complex systems, were the computational cost is a limiting element of the performance evaluation process.

References

1. Momchil Halstrup. *Black-box optimization of mixed discrete-continuous optimization problems*. PhD thesis, TU Dortmund, jan 2016.
2. Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998.
3. Julien Pelamatti, Loïc Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Yannick Guerin. Overview and comparison of Gaussian process-based surrogate models for mixed continuous and discrete variables, application on aerospace design problems (In Review). *High-performance simulation based optimization, Springer series on Computational Intelligence*, 2018.
4. Carl Edward. Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.
5. Matthias Schonlau, William J Welch, and Donald R Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, pages 11–25, 1998.
6. Qiang Zhou, Peter Z. G. Qian, and Shiyu Zhou. A Simple Approach to Emulation for Computer Models With Qualitative and Quantitative Factors. *Technometrics*, 53(3):266–273, aug 2011.

Single and multiobjective evolutionary algorithms for clustering biomedical information with unknown number of clusters

María Eugenia Curi[1], Lucía Carozzi[1], Renzo Massobrio[1], Sergio Nesmachnow[1], Grégoire Danoy[2], Marek Ostaszewski[3], and Pascal Bouvry[2]

[1] Universidad de la República, Uruguay
{maria.curi,lucia.carozzi,renzom,sergion}@fing.edu.uy
[2] FSTC/CSC-ILIAS, University of Luxembourg, Luxembourg
{gregoire.danoy,pascal.bouvry}@uni.lu
[3] LCSB, University of Luxembourg, Luxembourg
marek.ostaszewski@uni.lu

Abstract. This article presents single and multiobjective evolutionary approaches for solving the clustering problem with unknown number of clusters. Simple and ad-hoc operators are proposed, aiming to keep the evolutionary search as simple as possible in order to scale up for solving large instances. The experimental evaluation is performed considering a set of real problem instances, including a real-life problem of analyzing biomedical information in the Parkinson's disease map project. The main results demonstrate that the proposed evolutionary approaches are able to compute accurate trade-off solutions and efficiently handle the problem instance involving biomedical information.

Keywords: clustering, biomedical information, multiobjective

1 Introduction

The clustering problem aims at grouping a set of elements in such a way that elements in the same group (*cluster*) are more similar to each other than to the elements in other clusters [1]. Similarity between elements is evaluated according to a predefined similarity metric to be maximized. Clustering is one of the most important unsupervised learning problems, which models many other problems dealing with finding a structure in a given set of data.

In particular, biomedical research demands dealing with a large number of concepts linked by complex relationships, which are often represented using large graphs. In order to process and understand these knowledge bases, researchers need reliable tools for visualizing and exploring large amounts of data conveniently. In order to get a deep understanding of such knowledge bases, concepts with similar characteristics need to be accurately grouped together.

Clustering is an NP-hard optimization problem [2] that has been thoroughly studied in the last 30 years [3]. Heuristics and metaheuristics [4] have been

applied to solve the clustering problem efficiently. Among them, evolutionary algorithms (EAs) have proven to be accurate and powerful methods [5, 6].

This article addresses two formulations of the clustering problem, a first one in which the number of clusters is known in advance, and a multiobjective variant which simultaneously maximizes the similarity between elements in the same cluster and minimizes the number of clusters. Three EAs are presented, two for the single objective and one for the multiobjective clustering problem. The proposed EAs are compared against several methods from the related literature. The evaluation focuses on a large problem instance from the Parkinson's disease map project [7], a research initiative that proposes building a knowledge repository to describe molecular mechanisms related to that condition [8]. The repository compiles literature-based information about Parkinson's disease and organizes the main concepts and contents in an easy to explore and freely accessible map, including experimental data, drug targets and other concepts.

The article is organized as follows. Section 2 presents the single and multiobjective clustering problem formulation and reviews related works on heuristics and metaheuristics applied to the clustering problem. The proposed EAs are described in Section 3 and the experimental evaluation is reported in Section 4. Finally, Section 5 presents the conclusions and the main lines for future work.

2 Clustering problem and related work

This section defines the clustering problem in both its single and multiobjective variants and reviews related works.

2.1 Problem formulation

Let us consider the following elements:

- The set $E = \{e_1, e_2, ..., e_n\}$ of elements to be grouped.
- The function $s : E \times E \rightarrow [0, 1]$; $s(e_i, e_j)$ is the similarity between e_i and e_j. The following conditions hold: $\forall e_i, e_j, s(e_i, e_j) = s(e_j, e_i)$ and $s(e_i, e_i) = 1$.
- An integer $k > 0$, which indicates the number of clusters to consider for grouping elements (*only for the single-objective version of the problem*).

The clustering problem consists in assigning the elements in E to a set of groups (*clusters*) $G = \{G_1, ..., G_k\}$; $G_i = \{c_i\} \cup \{e_m / s(e_m, c_i) \leq s(e_m, c_j) \forall e_m \in E, c_j, c_i \in C, i \neq j\}$; $C \subseteq E$, $|C| = k$ is the set of centers of the groups. The following properties hold: a) *cluster index in* $[1, k]$) $\forall (i, j), i \neq j : 1 \leq i, j \leq k$, and b) *clusters are disjoint sets* $G_i \cap G_j = \emptyset$.

The goal of the single objective version of the problem is to maximize the *total similarity* metric (TS) defined in Equation 1.

$$\max \quad TS = \sum_{e_i \in E} \max_{c_i \in C} \; s(e_i, c_i) \tag{1}$$

In the multiobjective version of the problem, the goal is to simultaneously maximize the value of TS and minimize the number of clusters k.

2.2 Related work

Many articles have presented heuristic and metaheuristic methods applied to the clustering problem. Early works considered the single objective version of the problem, based on minimizing the distance or maximizing similarity.

Das et al. [9] reviewed the application of metaheuristics for clustering problems. Trajectory-based metaheuristics offer limited problem solving capabilities, mainly due to scalability issues when solving large problem instances. Deng and Bard [10] applied GRASP for the Capacitated Clustering Problem, which proposes grouping elements in clusters, where each cluster has capacity constraints (minimum and maximum number of elements). GRASP was able to find optimal solutions for the problem instances with 30 and 40 nodes, and outperformed solutions found using CPLEX when using an execution time limit of one hour.

Early proposed EAs did not follow an explicit multiobjective approach. Sheng and Liu [6] compared k-medoids, local search, and Hybrid K-medoid Algorithm (HKA) over two datasets (517 elements/10 groups, and 2945 elements/30 groups). HKA obtained the best results on the largest problem instance and slightly better results for the small test problem. The EA by Cowgill et al. [11] optimized clustering metrics defined in terms of external cluster isolation and internal cluster homogeneity, improving over hierarchical clustering algorithms considering an internal criterion. Bandyopadhyay and Maulik [12] proposed an EA for clustering with a number of clusters not defined a priori, to analyze several clustering metrics.

Multiobjective EAs (MOEAs) for clustering have been presented in the book by Maulik et al. [13], most of them focused on optimizing two similarity metrics, thus studying different features of the data to analyze. The multiobjective approach by Ripon et al. [14] considered intracluster variation and intercluster distance, without assuming the number of clusters. The experimental analysis over problems with up to 3000 elements, nine classes, and two features, showed improved solutions over a custom NSGA-II.Handl and Knowles [15] proposed multiobjective clustering with automatic k determination (MOCK), considering objective functions based on compactness (deviation) and connectedness of clusters. These are conflicting objectives because the overall deviation improves when using more clusters, but the connectivity decreases. MOCK showed good behavior and scalability when compared with single-objective clustering algorithms. Korkmaz et al. [16] presented a Pareto-based MOEA to find a set of non-dominated clusters considering intracluster variation and the number of clusters. The experimental evaluation was performed over two small standard datasets (150 and 75 elements, with only two attributes), but no numerical results or multiobjective optimization analysis is reported.

Most of the previous works have proposed ad-hoc EAs to address the clustering problem and few of them have solved multiobjective variants. This article contributes with simple EAs and an explicit MOEA designed to scale properly for solving large problem instances, and we focus on a real-life instance considering biomedical information in the context of the Parkinson disease map project.

3 Evolutionary algorithms for clustering

This section describes in detail the single and multiobjective EAs proposed to tackle the clustering problem.

3.1 Single objective EAs

Fitness function. The fitness function computes the sum of similarities between each element and its most similar center, as presented in Section 2.1.

Solution encoding. Two solution encodings are proposed and evaluated. Binary encoding: a solution is represented as a binary vector of length n (the number of elements to be grouped). Each position in the vector represents whether the corresponding element is a group center (1) or not (0). Integer encoding: each solution is a vector of k integers in $[1, N]$, representing the set of cluster centers. Numbers only appear once, as the k clusters must have different centers.

Crossover operators. Two crossover operators were implemented for the binary encoding: Single Point Crossover (SPX) randomly selects a crossover position and exchanges the genes after the crossover point between both parents. Two-Point Crossover (2PX) randomly selects two crossover positions and exchanges the genes located between these two points.

For the integer encoding, three crossover operators were implemented: SPX, Generalized Cut and Splice (GenC&S), and Hybrid Crossover (SPX-GenC&S). GenC&S is a variant of Cut and Splice (C&S) [17] for the clustering problem, to preserve useful features of the information in both parents (Algorithm 1). GenC&S selects a random cutting point cp on one parent and a random integer $s \in [0, k]$. Two lists are created, sorted by similarity with the element on position cp in parent1: LP1 (elements on parent1) and LP2 (elements in parent2). The first s elements in LP1 are copied to offspring1 and the $k - s$ remaining elements are copied from LP2, if their similarity to elements already copied to offspring1 is smaller than the input parameter ε. If less than k centers are copied to offspring1, the solution is completed with randomly selected centers. SPX-GenC&S uses a single random number p instead of cp and s. Elements before p in parent1 are copied to offspring1 (like in SPX), and the $k - p$ remaining elements in offspring1 are copied from parent2, if their similarity to elements already copied to offspring1 is smaller than ε (like in GenC&S). If less than k centers are copied to offspring1, the solution is completed with randomly selected centers.

Mutation operators. Five mutation operators were implemented. For binary encoding, Bit Flip Mutation changes encoded values by the opposite binary value; Add Mutation changes data points to centers; and Delete Mutation changes centers to data points. For integer encoding, One Gene Mutation changes elements to another that is not included in the solution (randomly selected according to a uniform distribution in the set E) and Adapted One Gene Mutation changes an element in the encoding to the most similar element, found by applying the following search: all elements in the solution are processed, and the similarity to

Algorithm 1 GenC&S crossover for the clustering problem (integer encoding)

Input: parent1, parent2, ε; **Output:** offspring1
cp = rand(0,k)
s = rand(0,k)
cp_element = parent1[cp]
offspring1.add(cp_element)
LP1 = sortAscending(parent1,cp_element)
LP2 = sortAscending(parent2,cp_element)
for $i = 0$ to $s - 1$ **do** ▷ Copy the first s elements from LP1 to offspring1
  offspring1.add(LP1[i])
end for
for $j = 0$ to $k - s$ **do** ▷ Copy the first $N - s$ elements from LP2 to offspring1
  if similarity(LP2[j],offspring1)$< \varepsilon$ **then** ▷ not too close
    offspring1.add(LP2[j]) ▷ already in offspring1
  end if
end for
while offspring1.length() $< k$ **do** ▷ Complete with random elements
  new_center = rand(0,N)
  offspring1.add(new_center)
end while

the element being mutated is evaluated. The best similarity value (γ) is stored and the new center is selected to have a similarity less than γ.

Corrective function. Some evolutionary operators do not guarantee to preserve the number of centers in a solution. A simple corrective function is applied both for binary and integer encodings. For binary encoding, if the number of 1s in the solution is not k, random centers are added or deleted until the solution becomes feasible. For integer encoding, if the same element appears more than once in the vector, each repeated element is replaced with another chosen randomly (uniform distribution) among elements that are not already centers.

Population initialization. The individuals in the population are randomly generated following a uniform distribution in $\{0, 1\}$ (binary encoding) and a uniform distribution in the set of centers C (integer encoding). The initialization procedure generates feasible solutions by applying the corrective function to each individual in the initial population.

3.2 Multiobjective EA

A variant of NSGA-II [18] was implemented to solve the multiobjective variant of the clustering problem. Following an incremental approach, the encoding and evolutionary operators that achieved the best results in the comparative analysis of the single objective EA for the problem were used in the proposed NSGA-II: binary encoding, SPX, and Delete Mutation.

In the multiobjective problem, the solution with all genes set to 0 is not feasible, since it does not represent any grouping at all. To avoid this situation, the corrective function randomly adds a center to the solution. The initial pop-

ulation is randomly generated following a uniform distribution in $[0, 1]$ and the corrective function is applied to the generated individuals.

4 Experimental evaluation

This section describes the evaluation of the proposed EAs for clustering.

4.1 Problem instances

A total number of 13 problem instances were used to evaluate the proposed EAs. These instances correspond to clustering problems arising in different fields of study, including two instances that model the Parkinson's disease map:

- Instance #1 consists of hydrometric data from 46 basins in Uruguay [19].
- Instances #2 to #8 and #10 to #12 are from the Knowledge Extraction based on Evolutionary Learning dataset [20], a data repository for classification problems. These instances have between 80 and 846 elements each.
- Instances #9 and #13 contain data from the Parkinson's disease map, which visually represents all major molecular pathways involved in the Parkinson disease pathogenesis. Instance #9 has 801 elements. Instance #13 has 3056 elements and it is used to test the performance of the multiobjective approach on a large problem instance containing biomedical information.

4.2 Experimental configuration and methodology

Development and execution platform. The proposed EAs were developed using ECJ [21], an open source framework for evolutionary computation in Java. Experiments were performed on an Intel Core i5 @ 2.7GHz and 8 GB of RAM.

Results evaluation The results computed by the proposed EAs are compared against clustering algorithms from the literature in terms of the objective function (total similarity) and in terms of the relative hypervolume (RHV) metric for the multiobjective variant of the clustering problem. RHV is the ratio between the volumes (in the objective functions space) covered by the computed Pareto front and the volume covered by the true Pareto front. The ideal value for RHV is 1. The true Pareto front—unknown for the problem instances studied—is approximated by the set of non-dominated solutions found in each execution.

The algorithms used in the comparison are:

- *k-medoids* [22], a classic partitional method related to k-means. Clusters are built to minimize the distance between points and the center of the corresponding cluster, according to a given distance metric.
- *Linkage*, an agglomerative hierarchical clustering technique based on building clusters by combining elements of previously defined clusters. A distance function evaluates a relevant similarity metric for the problem and different linkage implementations use different distance functions. The Matlab implementation of single linkage (nearest neighbor), which uses the smallest distance between objects in the two cluster, in the results comparison.

– *Local Search* [6], combining k-medoids and an exhaustive search performed for each cluster. Starting from a randomly selected set of centers, the set of p nearest neighbors is found for each center. A local search is performed over these sets to find a new center that minimizes the distance with all elements. The search ends when no center is changed in two consecutive iterations.
– *Greedy*, which builds clusters iteratively, taking a locally optimal decision in each step. Starting from a randomly selected center, in each step searches for the element with the lowest similarity with the solution already built. This element is included in the solution as a new center. All clusters are recomputed and the procedure is applied until building k clusters.
– *Hybrid EA*, combining an EA and the local search by Sheng and Liu [6] (Algorithm 2). The hybrid EA uses binary encoding, random initialization, tournament selection, Mix Subset Recombination, and Bit Flip Mutation.

Algorithm 2 Generic schema of the hybrid EA for the clustering problem

```
Initialize k centers randomly
while not stopping_criterion do
    [parent1, parent2] = TournamentSelection(P)
    if rand(0,1) > p_C then
        [offspring1, offspring2] = Mix Subset Recombination(parent1, parent2)
    end if
    [offspring1, offspring2] = Bit Flip Mutation(p_M)
    if rand(0,1) > p_LS then
        [offspring1, offspring2] = Local Search()
    end if
end while
return best solution found
```

Statistical analysis. Thirty independent executions of each algorithm were performed over each problem instance to have statistical confidence. For each problem instance, the best and the average fitness value (for the single objective problem) and the average multiobjective metrics (for the multiobjective problem) are reported. The Kolmogorov-Smirnov test is applied to each set of results to assess if the values follow a normal distribution. After that, the non-parametric Kruskal-Wallis test is applied to compare the results distributions obtained by different algorithms. A confidence level of 95% is used for both statistical tests.

4.3 Single objective clustering problem

Parameter settings. The parameter values of each algorithm were configured based on preliminary experiments and suggestions from related works:

– *Single objective EAs*: population size (pop)= 100, crossover probability (p_C) = 0.75, mutation probability (p_M) = 0.01, tournament size = 2, and stopping criterion of 10000 generations.
– *k-medoids*: the algorithm stops when the cluster centers remain unchanged in consecutive iterations.

– *Local search*: size of the search neighborhood = 3 and the stopping criterion is the same as for k-medoids, as recommended by Sheng and Liu [6].
– *Hybrid EA*: $pop = 30$, $p_C = 0.95$, $p_M = 0.02$, $p_{LS} = 0.2$, neighborhood size = 3, tournament size = 2, and stopping criterion of 10000 generations.

Comparison of evolutionary operators. For the binary encoding, two crossovers and three mutations were proposed, generating six possible combinations: SPX and Bit Flip Mutation (*SPX-bit*), SPX and Add Mutation (*SPX-add*), SPX and Delete Mutation (*SPX-del*), 2PX and Bit Flip Mutation (*2PX-bit*), 2PX and Add Mutation (*2PX-add*), and 2PX and Delete Mutation (*2PX-del*). Experimental results showed that *SPX-del* performed better on small problem instances, outperforming the other combinations of evolutionary operators. On medium sized instances #5 and #6, *SPX-bit* computed the best results, while on large instances *2PX-del* achieved the best results. Therefore, the rest of the experimental analysis of the single objective EA using binary encoding focused on these three combinations of evolutionary operators.

For the integer encoding, three crossover operators and two mutations were presented, generating six possible combinations: SPX and One Gene Mutation (*SPX-One*), SPX and Adapted One Gene Mutation (*SPX-Adapt*), SPX-GenC&S Crossover and One Gene Mutation (*SPXGCS-One*), SPX-GenC&S Crossover and Adapted One Gene Mutation (*SPXGCS-Adapt*), GenC&S Crossover and One Gene Mutation (*GCS-One*), and GenC&S Crossover and Adapted One Gene Mutation (*GCS-Adapt*). Results showed that *SPX-One* computed the best results in 7 instances and *GCS-One* in 5 instances, both outperforming the other combinations. Therefore, the rest of the experimental analysis of the single objective EA using integer encoding focused on these two combinations.

Comparison of solution encodings. Table 1 reports the average similarity results computed on 30 independent executions of the proposed EA using binary and integer encoding and the evolutionary operators that achieved the best results in the previous analysis.

Table 1: Average similarity using different encodings and evolutionary operators.

#I	integer encoding		binary encoding		
	SPX-One	*GCS-One*	*SPX-bit*	*SPX-del*	*2PX-del*
#1	**18.66**	**18.66**	**18.66**	**18.66**	**18.66**
#2	**1.96**	**1.96**	**1.96**	**1.96**	**1.96**
#3	12.42	12.44	12.27	**12.46**	**12.46**
#4	16.43	16.41	15.93	**16.50**	**16.50**
#5	78.35	78.16	**78.61**	78.51	78.42
#6	116.18	116.39	**116.45**	115.69	115.34
#7	54.71	54.68	54.80	**54.98**	**54.98**
#8	63.27	63.30	61.10	63.42	**63.43**
#9	673.57	656.56	633.91	**675.20**	**675.20**
#10	37.77	36.49	35.88	**38.22**	**38.22**
#11	235.33	229.58	221.17	**236.11**	**236.11**
#12	32.89	32.08	31.20	**33.23**	**33.23**

Results indicate that the binary encoded EA using *SPX-del* and *2PX-del* significantly outperformed the results computed using integer encoding and *SPX-bit*. There is no significant difference when using *SPX-del* and *2PX-del*, and for simplicity, the rest of the experimental evaluation was performed using *SPX-del*.

Comparison against other algorithms. The proposed EA with binary encoding, SPX, and delete mutation was compared against the baseline algorithms. Table 2 reports the average similarity computed over 30 independent executions of each algorithm for the 12 problem instances (the best results are marked in bold). The Kolmogorov-Smirnov test was performed on the results' distributions. In most cases, the test allowed rejecting–with 95% confidence–the null hypothesis that the results follow a normal distribution. Therefore, the Kruskal-Wallis test was used to compare the results' distributions computed by each EA (the p-value is reported in the last column). Kruskal-Wallis allows rejecting the null hypothesis that the results computed by all algorithms follow the same distribution.

Table 2: Comparison of average similarity against other algorithms.

instance	*greedy*	*linkage*	*k-medoids*	*local search*	*hybrid*	*EA*	*SPX-del*	*p-value K-W*
#1	7.28	17.01	17.03	15.49	**18.66**		**18.66**	$< 10^{-15}$
#2	1.12	1.65	1.95	1.70	**1.96**		**1.96**	$< 10^{-15}$
#3	5.77	10.18	12.14	10.50	12.45		**12.46**	$< 10^{-15}$
#4	7.41	14.04	16.00	13.23	16.22		**16.50**	$< 10^{-15}$
#5	47.69	76.08	76.47	69.11	**78.62**		78.51	$< 10^{-15}$
#6	83.61	109.68	116.30	108.86	**116.45**		115.69	$< 10^{-15}$
#7	29.31	50.77	54.98	41.68	**54.98**		**54.98**	$< 10^{-15}$
#8	31.81	62.25	62.51	52.99	63.24		**63.42**	$< 10^{-15}$
#9	499.54	523.19	667.94	615.64	661.48		**675.20**	$< 10^{-15}$
#10	22.90	30.61	37.09	32.94	36.73		**38.22**	$< 10^{-15}$
#11	170.65	198.75	236.10	205.96	229.56		**236.11**	$< 10^{-15}$
#12	22.80	27.02	32.85	28.56	33.10		**33.23**	$< 10^{-15}$

The proposed EA outperformed all other algorithms, computing the best average results in 10 instances. Improvements were up to 9.5% over k-medoids and 156.2% over greedy. The proposed EA also improved over Linkage in up to 29.1% and over the local search on of 31.9%. Finally, the improvements against the hybrid algorithm are smaller. In the best case (instance #10) the proposed EA outperformed the hybrid EA in up to 4.0% (2.3% on average).

4.4 Multiobjective clustering problem

Parameters setting. The parameters of the proposed MOEA were defined based on preliminary experiments: $pop = 100$, $p_C = 0.75$, $p_M = 0.01$, tournament of size 2, and a stopping criteria of 1000 generations.

Numerical results. The best EA for the single objective clustering problem (i.e., using SPX and delete mutation) and k-medoids were used to compare the NSGA-II results. 30 independent executions of each algorithm were executed, changing the number of clusters for the single objective algorithms.

Figures 1 and 2 show sample Pareto fronts computed by the proposed MOEA and the best solutions computed by k-medoids and in 30 independent executions of the single objective EA using different numbers of clusters. These are representative results for the set of problem instances solved.

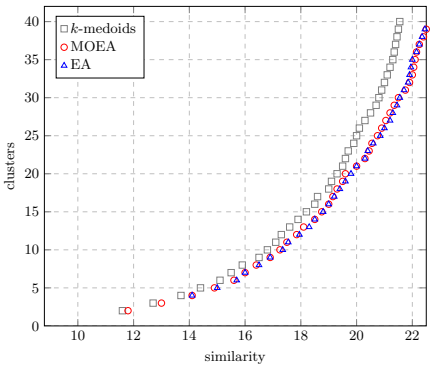


Fig. 1: Pareto fronts for instance #4

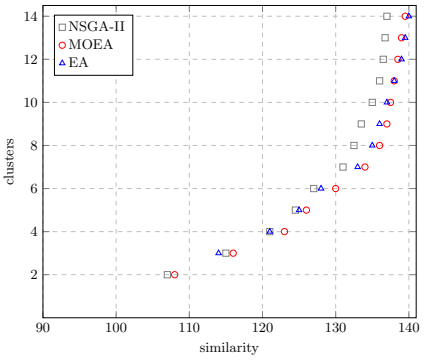


Fig. 2: Pareto fronts for instance #6

Results showed that for small number of clusters there is no significant difference in the solutions computed by EA and MOEA. Both evolutionary approaches improve over k-medoids. As the number of groups increases, the MOEA is able to found solutions with better similarity values than the single objective EA, and both significantly improves over the k-medoids results. In addition, the MOEA is able to obtain a Pareto front of solutions with different trade-off values in a single execution, while several executions (each one for a different number of clusters) are needed for the single objective EA and k-medoids. Therefore, the MOEA is useful for a decision-maker to be able to visualize several groupings with different trade-offs between the problem objectives and select the one that better captures the problem features. This is especially relevant in the case of biomedical information, where the number of clusters is particularly difficult to define a priori for a given dataset.

The RHV results over 30 independent executions, reported in Table 3, indicated that the proposed MOEA is robust and computes accurate Pareto fronts for the problem instances studied. The average RHV was 0.99, the maximum difference from the ideal RHV was 0.02 (instances #6 and #12), and the optimum value of 1.00 was achieved for three problem instances.

Table 3: RHV values obtained by the proposed algorithms.

MOEA		EA		k-medoids	
average	*best*	*average*	*best*	*average*	*best*
0.99	1.00	0.96	1.00	0.83	0.92

Regarding the problem instances from the Parkinson's disease map, the proposed EAs allowed to compute accurate configurations that provide different trade-offs between the problem objectives. Using the evolutionary approaches,

several new possible clusterings have been found. These clusters provide novel promising information, different to the current manually built solutions (see the project website at http://wwwen.uni.lu/lcsb/research/parkinson_s_disease_map).

Overall, considering the complete set of problem instances, EA and MOEA were able to improve over k-medoids 15.8% and 14.1% in average (respectively), and up to 31.4% and 27.0% in the best case. The best improvements were obtained in the problem instances with larger number of elements, clearly demonstrating the good scalability behavior of the proposed evolutionary approaches. The best improvement of EA over MOEA was 8.7% and the best improvement of MOEA over EA was 4.4%.

5 Conclusions and future work

This article presented evolutionary algorithms applied to the clustering problem in its single and multiobjective variants, with unknown number of clusters. This is a very important problem in many research areas that involve dealing with large volumes of information to be categorized and grouped.

The proposed evolutionary algorithms were conceived to apply simple and ad-hoc operators, trying to keep the search as straightforward as possible in order to scale up for solving large instances of the clustering problem.

The experimental evaluation was performed considering a set of real problem instances, including one problem consisting of biomedical information in the context of the Parkinson disease map project. The main results from the experimental analysis indicate that the proposed evolutionary algorithms are able to compute accurate solutions for the problem instances studied. The evolutionary approaches outperform several algorithms of the related literature. In the single objective clustering problem, the proposed evolutionary algorithm is able to compute the best average result in 10 out of 12 problem instances. For the multiobjective clustering problem, the proposed evolutionary algorithm is able to compute accurate Pareto fronts, which offer decision-makers solutions with different trade-offs between the problem objectives.

The evolutionary approach is especially helpful for organizing biomedical information in the case of the Parkinson's disease map project. The proposed EAs are able to find accurate organizations for the data, which provide different trade-offs between the problem objectives and allow capturing different features of the information. The computed solutions provide new promising clustering patterns to be examined along the existing ones, manually built by experts.

The main lines of future work include extending the experimental analysis considering datasets from different fields of study. Additionally, a parallel model for EAs should be considered to both reduce execution times and handle bigger datasets. Finally, the possibility of combining the proposed evolutionary algorithms with visualization tools should be studied, in order to help researchers analyze the information in a more intuitive way.

References

1. Kaufman, L., Rousseeuw, P.: Finding groups in data: an introduction to cluster analysis. Wiley, New York (1990)
2. Welch, W.: Algorithmic complexity: threeNP- hard problems in computational statistics. Journal of Statistical Computation and Simulation **15**(1) (1982) 17–25
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer New York (2009)
4. Nesmachnow, S.: An overview of metaheuristics: accurate and efficient methods for optimisation. International Journal of Metaheuristics **3**(4) (2014) 320–347
5. Hruschka, E., Campello, R., Freitas, A., de Carvalho, A.: A survey of evolutionary algorithms for clustering. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **39**(2) (2009) 133–155
6. Sheng, W., Liu, X.: A hybrid algorithm for k-medoid clustering of large data sets. In: IEEE Congress on Evolutionary Computation. (2004) 77–82
7. University of Luxembourg: Parkinson's disease map project http://wwwen.uni.lu/lcsb/research/parkinson_s_disease_map [November, 2017].
8. K. Fujita et al.: Integrating pathways of Parkinson's disease in a molecular interaction map. Molecular Neurobiology **49**(1) (2014) 88–102
9. Das, S., Abraham, A., Konar, A.: Metaheuristic Clustering. Volume 178 of Studies in Computational Intelligence. Springer (2009)
10. Deng, Y., Bard, J.: A reactive GRASP with path relinking for capacitated clustering. Journal of Heuristics **17**(2) (2011) 119–152
11. Cowgill, M., Harvey, R., Watson, L.: A genetic algorithm approach to cluster analysis. Computers & Mathematics with Applications **37**(7) (1999) 99–108
12. Bandyopadhyay, S., Maulik, U.: Nonparametric genetic clustering: comparison of validity indices. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) **31**(1) (2001) 120–125
13. Maulik, U., Bandyopadhyay, S., Mukhopadhyay, A.: Multiobjective Genetic Algorithms for Clustering. Springer Nature (2011)
14. Ripon, K., Tsang, C.H., Kwong, S., Ip, M.K.: Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In: 18^{th} International Conference on Pattern Recognition. (2006) 3609–3616
15. Handl, J., Knowles, J.: An evolutionary approach to multiobjective clustering. IEEE Transactions on Evolutionary Computation **11**(1) (2007) 56–76
16. Korkmaz, E., Du, J., Alhajj, R., Barker, K.: Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. Intelligent Data Analysis **10**(2) (2006) 163–182
17. Deaven, D., Ho, K.: Molecular geometry optimization with a genetic algorithm. Physical Review Letters **75** (1995) 288–291
18. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons (2001)
19. Ministerio de Vivienda Ordenamiento Territorial y Medio Ambiente (Uruguay): Red de estaciones hidrométricas http://www.mvotma.gub.uy [November, 2017].
20. J. Alcalá et al.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing **17**(2-3) (2010) 255–287
21. Sean Luke et al.: ECJ 23: a Java-based Evolutionary Computation Research System [Online], https://cs.gmu.edu/ eclab/projects/ecj, accessed March 2017.
22. Kaufman, L., Rousseeuw, P.: Clustering by means of medoids. In: Statistical Data Analysis Based on the L1-Norm and Related Methods. (1987)

Ensemble of Kriging with Multiple Kernel Functions for Engineering Design Optimization

Pramudita Satria Palar[1] and Koji Shimoyama[1]

Institute of Fluid Science, Tohoku University, Sendai, Miyagi Prefecture 980-8577, Japan
pramudita.satria.palar.a5@tohoku.ac.jp

Abstract. We introduce the ensemble of Kriging with multiple kernel functions guided by cross-validation error for creating a robust and accurate surrogate model to handle engineering design problems. By using the ensemble of Kriging models, the resulting ensemble model preserves the uncertainty structure of Kriging, thus, can be further exploited for Bayesian optimization. The objective of this paper is to develop a Kriging methodology that eliminates the needs for manual kernel selection which might not be optimal for a specific application. Kriging models with three kernel functions, that is, Gaussian, Matérn-3/2, and Matérn-5/2 are combined through a global and a local ensemble technique where their approximation quality are investigated on a set of aerodynamic problems. Results show that the ensemble approaches are more robust in terms of accuracy and able to perform similarly to the best performing individual kernel function or avoiding misspecification of kernel.

1 Introduction

The computationally expensive nature of many real-world engineering optimizations hinders the crude of use of evolutionary algorithms (EA) and other metaheuristics for obtaining highly optimized designs. To this end, surrogate models are now commonly deployed to act as a replacement for black-box functions in order to accelerate the optimization process. There are basically two frameworks to apply surrogate models, that is, to utilize them either as a global or local surrogate model. Global surrogate models are particularly useful when the number of design variables is low to moderate under the constraint of a limited computational budget. On the other hand, local surrogate models are typically used under the condition of high-dimensionality and moderate computational budget, such as to assist the local search for memetic algorithm [1][2]. For a comprehensive review of this topic, readers are referred to Jin [3] and Viana et al. [4].

Kriging is one of the most widely used types of surrogate model for approximating engineering functions. One powerful aspect of Kriging models is that they provide a measure of estimation error that could be used to guide Bayesian optimization or error-based refinement in order to improve the approximation

quality [5]. The most widely used kernel function for constructing Kriging models in the context of engineering design is the Gaussian kernel function. On the other hand, Stein recommends that Matérn kernel function should be used instead of Gaussian since the smoothness of Gaussian function is unrealistic for many real-world processes [6]. It is worth noting that the best kernel function is highly problem dependent; therefore, it is of utmost importance to correctly deploy a proper kernel for optimum approximation accuracy.

One approach to combine or take the best from multiple surrogate models is to perform an ensemble of surrogate models [7][8] [9] [10] [11]. Traditionally, various surrogate models such as radial basis function (RBF), Kriging, and support vector regression are combined together, which results in the inapplicability of Bayesian optimization (e.g., efficient global optimization [5]). Bayesian optimization can be performed with the ensemble model if each of the constituent models possesses an uncertainty structure. In this paper, we propose to combine multiple Kriging models with multiple kernel functions. The advantage of the ensemble of Kriging models with various kernel functions is that the uncertainty structure is still conserved. We tested the proposed framework on a set of aerodynamic problems using various ensemble methods. In this paper, we limit the research scope to only analyzing the approximation accuracy of the ensemble Kriging models and compared them to those with single kernel function.

Note that the mixture of Kriging with kernel function is not totally a new idea; in fact, this idea was first proposed by Ginsbourger et al. [12]. Ginsbourger et al. approach uses the combination of Gaussian and exponential kernel function and mix them globally with Akaike weights; while in this paper, we utilize the cross-validation (CV) error to mix the Kriging model with Gaussian and advanced Matérn kernel function using both the global and local ensemble. Moreover, Ginsbourger et al.'s method was only tested on Branin function while we directly used engineering functions in our study.

2 Ensemble of Kriging

2.1 Kriging model

We are interested in approximating a black box function $y = f(\boldsymbol{x})$ with a Kriging surrogate model, where $\boldsymbol{x} = \{x_1, x_2, \ldots, x_m\}$ and m is the dimensionality of the decision variables. The Kriging approximation is modeled as a realization of a stationary Gaussian process $Y(\boldsymbol{x})$ reads as

$$Y(\boldsymbol{x}) = \sum_{i=0}^{P-1} \alpha_i \Psi_i(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{\Psi}(\boldsymbol{x}) = \{\Psi_0(\boldsymbol{x}), \ldots, \Psi_{P-1}(\boldsymbol{x})\}$ is a collection of regression polynomial functions, $\boldsymbol{\alpha} = \{\alpha_0(\boldsymbol{x}), \ldots, \alpha_{P-1}(\boldsymbol{x})\}$ is the vector of regression coefficients, and Z is a stochastic process. In this paper, we use the ordinary Kriging which assumes that the trend is a constant Ψ_0.

One building block of Kriging is the covariance function which represents the similarity between two input points in the design space. There are several choices to model this similarity using different types of kernel function. In this paper, we opt for the Gaussian and Matérn class function due to their robustness and popularity in various applications. We do not opt for the exponential function since based on our experiment in aerodynamic functions, it did not yield a satisfactory accuracy. These kernel functions are explained in detail below.

Gaussian The Gaussian kernel function is defined as

$$R(h,\theta) = \exp\Big(-\sum_{k=1}^{m} \big(\frac{h}{\theta}\big)^2 \Big), \tag{2}$$

where $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ are the vector of hyperparameters that needs to be estimated and $h = \big|\boldsymbol{x} - \boldsymbol{x}'\big|$.

Matérn class The general form of Matérn kernel function is expressed as

$$R(h,\theta,\nu) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \Big(2\sqrt{\nu}\frac{|h|}{\theta}\Big) \mathcal{K}_\nu \Big(2\sqrt{\nu}\frac{|h|}{\theta}\Big), \tag{3}$$

where $\nu \geq 1/2$ is the shape parameter, Γ is the Gamma function, and $\mathcal{K}_\nu$ is the modified Bessel function of the second kind.

For $\nu = 3/2$, the formulation of Matérn kernel function is defined as

$$R(h,\theta,\nu = 3/2) = \Big(1 + \frac{\sqrt{3}|h|}{\theta}\Big) \exp\Big(-\frac{\sqrt{3}|h|}{\theta}\Big), \tag{4}$$

while for $\nu = 5/2$ is defined as

$$R(h,\theta,\nu = 5/2) = \Big(1 + \frac{\sqrt{5}|h|}{\theta} + \frac{5h^2}{3\theta^2}\Big) \exp\Big(-\frac{\sqrt{5}|h|}{\theta}\Big). \tag{5}$$

The Matérn$-3/2$ and Matérn$-5/2$ are two forms that are widely used to model real-world processes. We, therefore, used these two forms of Matérn kernel function in our study and compare it with the standard Gaussian.

A set of n observations points $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ and the responses $\boldsymbol{y} = \{y^{(1)}, \ldots, y^{(n)}\} = \{f(\boldsymbol{x}^{(1)}), \ldots, f(\boldsymbol{x})^{(n)}\}$ are collected first in order to create a Kriging surface. As opposed to the majority of the types of surrogate model, Kriging allows the computation of both the prediction $\hat{y}(\boldsymbol{x})$ and the mean-squared error $\hat{s}^2(\boldsymbol{x})$.

The ordinary Kriging prediction for an arbitrary input variable reads as

$$\hat{y}(\boldsymbol{x}) = \mu_{KR} + \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}), \tag{6}$$

with the mean-squared error of the Kriging prediction $\hat{s}(\boldsymbol{x})$ reads as:

$$\hat{s}^2(\boldsymbol{x}) = \sigma^2\big(1 - (\boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x})) + \big(1 - \boldsymbol{1}^T \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x})\big)^2 \big(\boldsymbol{1}^T \boldsymbol{R}^{-1} \boldsymbol{1}\big)^{-1}\big). \tag{7}$$

Here, $\boldsymbol{R}$ is the $n \times n$ matrix with the (i, j) entry is $\text{corr}[Z(\boldsymbol{x}^{(i)}), Z(\boldsymbol{x}^{(j)})]$, $\boldsymbol{r}(\boldsymbol{x})$ is the correlation vector between $\boldsymbol{x}$ and $\boldsymbol{\mathcal{X}}$ whose $(i, 1)$ entry is $\text{corr}[Z(\boldsymbol{x}^{(i)}), Z(\boldsymbol{x})]$, and $\mathbf{1}$ is the vector of ones with length n. As we can see from this formulation, the choice of kernel function enters the formulation through $\boldsymbol{R}$ and $\boldsymbol{R}(\boldsymbol{x})$. In this paper, we opt for the standard technique of maximizing the likelihood function to determine the hyperparameters. We do not go too much into details of the Kriging method; readers are referred to other literatures such as [5][13].

Since the determination of a proper kernel function is not trivial, we advocate the use of the ensemble of Kriging models with various kernel functions instead of choosing just one specific kernel. We hypothesized that this would improve the robustness and accuracy of Kriging models while still preserving the uncertainty structure through the law of total expectation and total variance [12].

2.2 Ensemble of Kriging models

Assuming that we possess K different surrogate models, the general form of the ensemble of surrogate models reads as

$$\hat{f}_{ens}(\boldsymbol{x}) = \sum_{i=1}^{K} w_i(\boldsymbol{x}) \hat{f}_i(\boldsymbol{x}) \tag{8}$$

where $\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_K(\boldsymbol{x})$ are K surrogate models to be combined into one model and $\boldsymbol{w}(\boldsymbol{x}) = \{w_1(\boldsymbol{x}, \ldots, w_i(\boldsymbol{x})\}$ are the weights that define the contribution of each surrogate model to the ensemble function.

To perform the ensemble of surrogates, we need the information of the CV error, i.e., $\boldsymbol{e}$ for each Kriging model. We firstly define $\boldsymbol{e}_i = \{e_i^{(1)}, \ldots, e_i^{(n)}\}$, where $e^{(j)} = y(\boldsymbol{x}^{(j)}) - \hat{y}^{(-j)}(\boldsymbol{x}^{(j)})$ is the CV error for sample j with the sample j is removed from the experimental design, as the CV errors for surrogate i. For Kriging models, the CV error can be obtained analytically without the need to construct Kriging n times [14]. The simplest approach is to directly select the Kriging model with the lowest CV error, where in this paper we opt for the root-mean-squared error (RMSE) to compute the CV error for each surrogate. However, as argued by Viana et al. [8], the ensemble of surrogate models is the better approach since it uses all information from each constituent surrogate model instead of directly choosing the best one in terms of the CV error.

There are two techniques to ensemble the function, that is, the global and local ensemble approach which are explained below.

Global ensemble The global ensemble approach employs a constant weight for each surrogate model in the range of the design space. In this paper, we opt for Acar and Rohani's approach [9] to construct the global ensemble. Here, the constant weight $\boldsymbol{w}$ is found by solving the following minimization problem

$$\min_{\boldsymbol{w}} \text{MSE}_{\text{ens}} = \mathbb{E}\big(e^2_{\text{WAS}}(\boldsymbol{x})\text{d}\boldsymbol{x}\big) = \mathbf{w}^T \mathbf{C} \mathbf{w}, \tag{9}$$

where MSE_{ens} is the mean squared error of the global ensemble, $\mathbf{C}$ is the matrix of CV error at sample points for all surrogate models, and e^2_{WAS} is the mean squared error of the weighted average surrogate at a specific point. Following Viana et al.'s suggestion [8], we only used the diagonal matrix of $\mathbf{C}$ to compute $\mathbf{w} = \{w_1, w_2, \ldots, w_K\}$ using Lagrange multipliers.

Local ensemble One downside of the global ensemble approach, in spite of its simplicity, is its inability to cope with the locality of the response surface. There are situations where Kriging with one type of kernel function is accurate in a certain region while another kernel is more suitable in other regions of the design space. To this end, the local ensemble is probably more suitable since it allows a non-constant weight function to be used. In this paper, we opt for Liu et al.'s approach [11] which originally proposed the method for creating the local ensemble of radial basis function models; readers are also referred to this paper for a more detail explanation about the method. Using Liu et al.'s approach, the weight for surrogate j at a certain design point is calculated by

$$w_j(\boldsymbol{x}) = \begin{cases} \text{if } \boldsymbol{x} \neq \boldsymbol{x}_i & : \sum_{i=1}^{n} \frac{d_i^{-B_i\Theta}}{\sum d_i^{-B_i\Theta}}, \\ \text{if } \boldsymbol{x} = \boldsymbol{x}_i & : W_{ij} \end{cases} \tag{10}$$

where Θ is the attenuation coefficient that is automatically selected using CV error, d_i is the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$, B_i is the normalized global accuracy of the constituent model that yields the the lowest error at $\boldsymbol{x}_i$, and $\boldsymbol{W}$ is the observed weight matrix.

According to the law of total expectation, the prediction and variance from the mixture of multiple Kriging models can be computed through the law of total expectation and total variance [12], respectively. In this paper, we use the UQLab open source software to construct the Kriging model [15].

3 Applications to Aerodynamic Problems

We consider two engineering test cases in order to demonstrate the efficacy of the ensemble methods. The two problems considered are the subsonic and transonic airfoil (i.e., the cross-section of an aircraft wing) design, with two subcases for the subsonic airfoil problem. Here, the output of interest for all cases is the drag coefficient (C_d), computed by a computational fluid dynamics method, which measures the efficiency of aerodynamic bodies. For each test case, we compared the Kriging model with Gaussian, Matérn-3/2, and Matérn-5/2 kernel functions, the scheme that yields the lowest CV error (i.e., model selection), local ensemble, and global ensemble. The Kriging quality is measured by the squared correlation coefficient, i.e., R^2. We also use the average performance score (APS) [16] to compare various Kriging methods. APS indicates the number of other methods that strictly dominate the method being investigated; thus, low APS value denotes a good performing method.

3.1 Subsonic airfoil problem

Design optimization of an airfoil in subsonic flow regime for low-speed is highly useful for such applications as unmanned aerial vehicles or low-speed training aircraft. For the subsonic airfoil problem, we used the PARSEC airfoil parameterization technique [17] (see Fig. 1 and Table 1) and we set the Reynolds and Mach number to 3×10^6 and 0.3, respectively. Here, the first and second subcase considers a fixed angle of attack of 2° and fix $C_l = 0.5$, respectively. The lower and upper bound (i.e., l_b and u_b, respectively) for the subsonic airfoil problem are shown in Table 1. Since this case is cheap, we could evaluate a large number of samples for the training and validation set. We used training sample points with $n = 40$ and $n = 80$ generated by Latin hypercube sampling with 1000 validation samples.

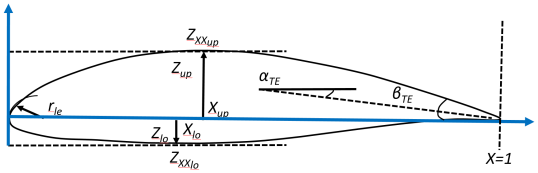


Fig. 1: Illustration of PARSEC airfoil parameterization.

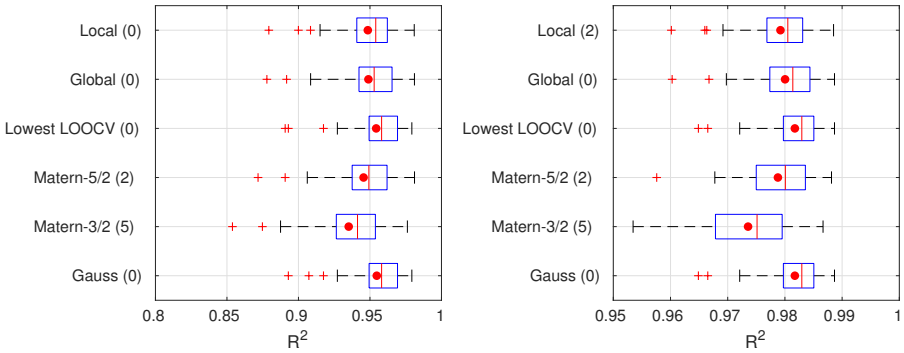
Table 1: The upper and lower bounds for the subsonic airfoil problem.

Variable	Definition	l_b	u_b
r_{le}	leading edge radius	0.0108	0.0162
X_{up}	upper crest position in horizontal coordinates	0.3288	0.4932
Z_{up}	upper crest position in vertical coordinates	0.0830	0.1245
$Z_{XX_{up}}$	upper crest curvature	-0.8700	-0.5800
X_{lo}	lower crest position in horizontal coordinates	0.3254	0.4881
Z_{lo}	lower crest position in vertical coordinates	-0.0690	-0.0460
$Z_{XX_{lo}}$	lower crest curvature	0.3086	0.4629
α_{te}	trailing edge direction	-0.2286	-0.1524
β_{te}	trailing edge wedge angle	0.1120	0.1680

Results for subcase 1. The R^2 results for the first subsonic case are shown in Fig. 2. Comparison of Kriging models with single kernel function shows that Gaussian is the most suitable kernel function for this particular problem, followed by Matérn-5/2 and Matérn-3/2. The global ensemble and model selection are

the most robust multiple kernel approaches that can match the approximation quality of the Kriging with the Gaussian kernel. On the other hand, the local ensemble is outperformed by two methods on a high number of sample points (i.e., $n = 80$).

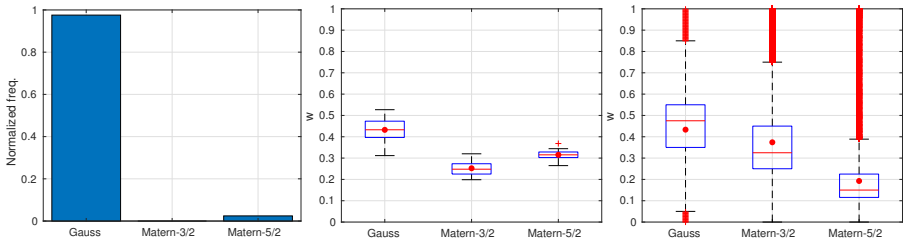
The weight distribution obtained from all 30 independent runs for $n = 40$ and $n = 80$ are shown in Figs. 3 and 4, respectively. First, the model selection has a very strong tendency to select Gaussian kernel over the others for both n; this indicates that the approximation quality of Kriging with the Gaussian kernel is far superior over the Matérn kernels for this problem. For the global ensemble scheme, there is a fairer distribution of kernel, where the kernel with the highest portion is Gaussian followed by Matérn-5/2 and Matérn-3/2. We observe a difference between the proportion of kernels for the global and local ensemble, that is, the latter tends to favor Matérn-3/2 over Matérn-5/2. This means that the Kriging model with Matérn-3/2 kernel is able to produce a locally accurate approximation near the design points over the Matérn-3/2 kernel; this trend is stronger with higher sample size. However, such scheme is not as optimal as the global ensemble for this particular problem.



(a) $n = 40$. (b) $n = 80$.

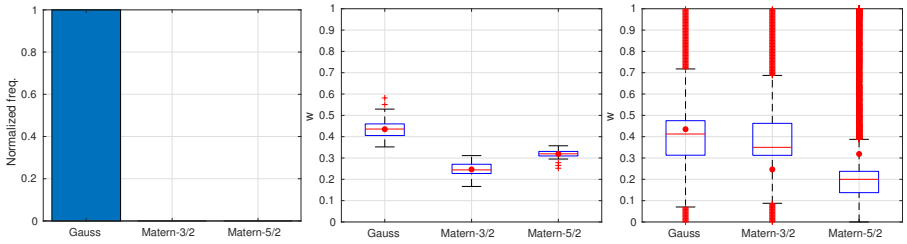
Fig. 2: R^2 results for the first subcase of subsonic airfoil problem. The number inside the bracket shows the corresponding APS value.

Result for subcase 2. The results for subsonic case 2 (see Fig. 5) reveal that the global ensemble is able to compete with the two best-performing methods for this problem, that is, Kriging with Matérn-3/2 and Matérn-5/2 kernel function. In contrast to the first subsonic case, the performance of Kriging with Gaussian kernel is not really satisfying as indicated by its high APS for both sample sizes. The model selection approach does not perform so well in low sample size, primarily due to the effect of low sample size on the CV accuracy. Results also show that the local ensemble is more favorable compared to the model



(a) Lowest LOOCV. (b) Global ensemble. (c) Local ensemble.

Fig. 3: Distribution of the weights for the first subsonic case with $n = 40$.



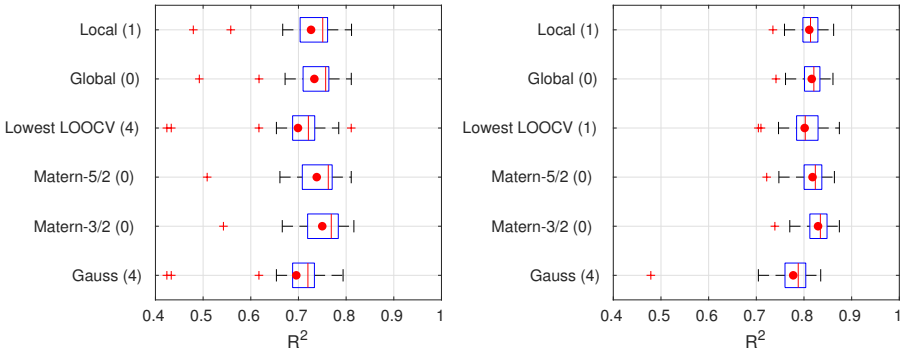
(a) Lowest LOOCV. (b) Global ensemble. (c) Local ensemble.

Fig. 4: Distribution of the weights for the first subsonic case with $n = 80$.

selection, indicating that mixing several Kriging models with multiple kernel function is more advantageous than model selection for this problem. However, the fact that the local ensemble is outperformed by the global ensemble means that the latter is even more favorable; also, the global ensemble is very easy to be constructed. The corresponding weights are shown in Figs. 6 and 7. It is interesting to see that although the ensemble approaches give more weights to the Gaussian kernel, combination with other kernel functions yields a suppressing effect to the inadequacy of Gaussian on this particular problem.

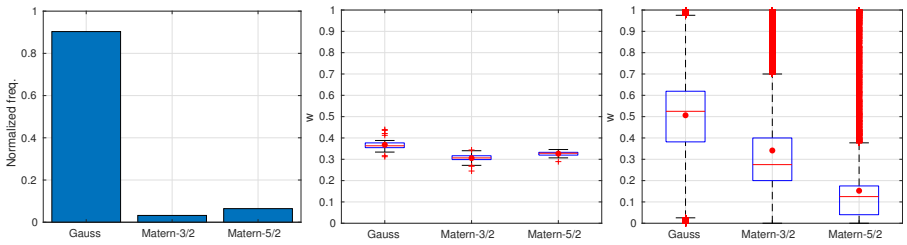
3.2 Inviscid transonic airfoil problem

The second sub-case is the design of inviscid (i.e., no friction) transonic airfoil problem in Mach number of 0.73, which is the flying regime of a modern commercial aircraft, and angle of attack of 2°. The airfoil shape for the transonic problem is parameterized by the Class Shape Transformation (CST) [18] method with a total of 16 variables (i.e., 8 variables for each upper and lower surface). We used the RAE 2822 airfoil shape as the datum and then varied the CST shape parameters by $\pm 20\%$. We set n to $n = 50$ and $n = 100$ by taking a subset of random samples from the available 400 samples and then use the other subset as validation samples.



(a) $n = 40$. (b) $n = 80$.

Fig. 5: R^2 results for the second subsonic airfoil case. The number inside the bracket shows the performance score.

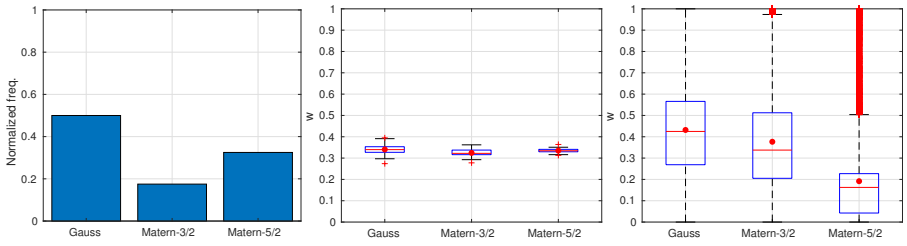


(a) Lowest LOOCV. (b) Global ensemble. (c) Local ensemble.

Fig. 6: Distribution of the weights for the second subsonic case with $n = 80$.

The R^2 results are depicted in Fig. 8 while the distributions of the generated weight are shown in Figs. 9 and 10. We observe that the Gaussian kernel strictly outperforms Matérn kernels, especially Matérn-3/2. Due to this significant performance difference, the performance of the model selection scheme successfully mimics that of the Kriging model with the Gaussian kernel. On the other hand, the local and the global ensemble approach are outperformed by the model selection and Kriging with the Gaussian kernel; however, it is worth noting that the ensemble schemes successfully avoid the relatively poor performance of Kriging with Matérn-3/2 kernel and are also better than that of Matérn-5/2. In this regard, the ensemble schemes act as a safeguard that prevents a misspecification of kernel that yields a relatively poor performance.

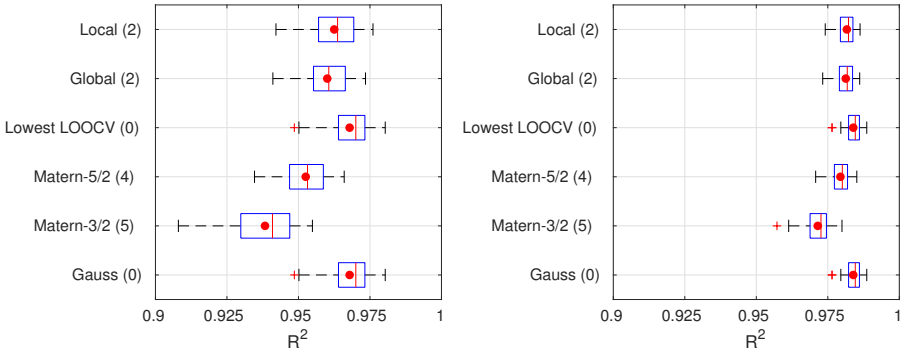
The trend of the weighting for the kernels shows a similar trend to that of the subsonic airfoil case. In the inviscid transonic airfoil case, the weight of Matérn-5/2 in the global ensemble scheme is more dominant than that of Matérn-3/2, while it is the opposite case for the local ensemble. Especially on the first subsonic and transonic airfoil case, Kriging with Matérn-5/2 kernel is more



(a) Lowest LOOCV. (b) Global ensemble. (c) Local ensemble.

Fig. 7: Distribution of the weights for the second subsonic case with $n = 80$.

globally accurate than that of Matérn-3/2; which explains why the local ensemble scheme is less accurate than the global one since the former tends to give more reward to the Matérn-3/2 kernel. However, when Bayesian optimization is to be performed, there is a chance that the local ensemble scheme would be better than the global ensemble due to that a local accuracy is more important for optimization; empirical experiments are needed to test this hypothesis.

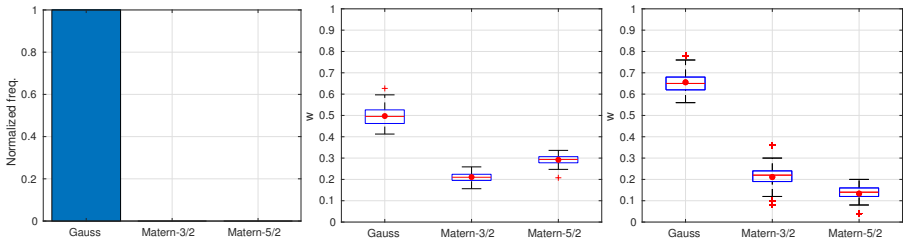


(a) $n = 40$. (b) $n = 80$.

Fig. 8: R^2 results for the inviscid transonic airfoil case. The number inside the bracket shows the performance score.

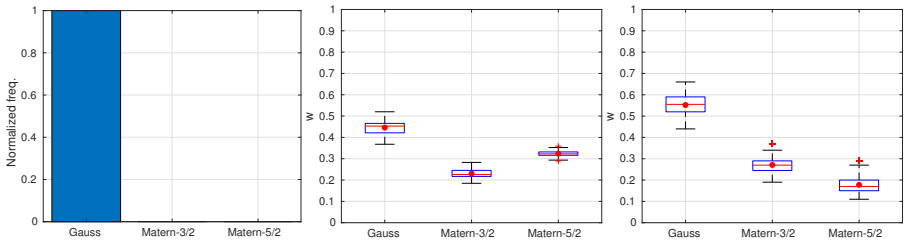
4 Conclusions and future works

In this paper, we studied the efficacy of the ensemble of Kriging with multiple kernel functions for approximating black-box engineering functions. Our research is motivated by the need to create a robustly accurate surrogate models without eliminating the advantage of uncertainty structure that can be used for



(a) Lowest LOOCV. (b) Global ensemble. (c) Local ensemble.

Fig. 9: Distribution of the weights for the inviscid transonic airfoil case with $n = 50$.



(a) Lowest LOOCV. (b) Global ensemble. (c) Local ensemble.

Fig. 10: Distribution of the weights for the inviscid transonic airfoil case with $n = 100$.

Bayesian sequential optimization strategy. To this end, we extend the previous work in the ensemble of Kriging with multiple kernel functions by introducing advanced kernel functions (i.e., Matérn class) besides the widely used Gaussian. Further, we implement the global and the local ensemble technique to mix multiple Kriging models. Since our primary objective is for engineering design, we directly tested the approach on aerodynamic problems as representatives for general engineering problems. It is shown that for the airfoil problems, the local and ensemble approaches are robust in terms of the approximation quality, in a sense that they could mimic the performance of the best performing kernel or at least avoiding misspecification of the kernel. Comparing the two ensemble approaches, the global ensemble is better in mixing multiple Kriging models than the local ensemble; further, this also comes with a simpler method for computing the weights. The model selection approach (i.e., select the model with the lowest LOOCV error), although might outperform ensemble approaches when a single kernel function is strictly better than the others, is prone to the misguidance of the CV error in selecting the best model as shown in the results from the second subsonic airfoil case.

For future works, benchmarking of the ensemble of Kriging models and those with single-kernel function should be performed within the Bayesian optimiza-

tion context. Furthermore, the capability of the aforementioned approach should also be investigated for applications besides optimization (e.g., uncertainty quantification and global sensitivity analysis).

References

1. Le, M.N., Ong, Y.S., Menzel, S., Jin, Y., Sendhoff, B.: Evolution by adapting surrogates. Evolutionary computation **21**(2) (2013) 313–340
2. Palar, P.S., Tsuchiya, T., Parks, G.T.: A comparative study of local search within a surrogate-assisted multi-objective memetic algorithm framework for expensive problems. Applied Soft Computing **43** (2016) 1–19
3. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2) (2011) 61–70
4. Viana, F.A., Simpson, T.W., Balabanov, V., Toropov, V.: Special section on multidisciplinary design optimization: Metamodeling in multidisciplinary design optimization: How far have we really come? AIAA Journal (2014)
5. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**(4) (1998) 455–492
6. Stein, M.L.: Interpolation of spatial data: some theory for kriging. Springer Science & Business Media (2012)
7. Goel, T., Haftka, R.T., Shyy, W., Queipo, N.V.: Ensemble of surrogates. Structural and Multidisciplinary Optimization **33**(3) (2007) 199–216
8. Viana, F.A., Haftka, R.T., Steffen, V.: Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. Structural and Multidisciplinary Optimization **39**(4) (2009) 439–457
9. Acar, E., Rais-Rohani, M.: Ensemble of metamodels with optimized weight factors. Structural and Multidisciplinary Optimization **37**(3) (2009) 279–294
10. Song, X., Sun, G., Li, G., Gao, W., Li, Q.: Crashworthiness optimization of foam-filled tapered thin-walled structure using multiple surrogate models. Structural and Multidisciplinary Optimization **47**(2) (2013) 221–231
11. Liu, H., Xu, S., Wang, X., Meng, J., Yang, S.: Optimal weighted pointwise ensemble of radial basis functions with different basis functions. AIAA Journal (2016)
12. Ginsbourger, D., Helbert, C., Carraro, L.: Discrete mixtures of kernels for kriging-based optimization. Quality and Reliability Engineering International **24**(6) (2008) 681–691
13. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical science (1989) 409–423
14. Dubrule, O.: Cross validation of kriging in a unique neighborhood. Mathematical Geology **15**(6) (1983) 687–699
15. Marelli, S., Sudret, B.: Uqlab: A framework for uncertainty quantification in matlab. In: Vulnerability, Uncertainty, and Risk: Quantification, Mitigation, and Management. (2014) 2554–2563
16. Bader, J., Zitzler, E.: Hype: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary computation **19**(1) (2011) 45–76
17. Sobieczky, H.: Parametric airfoils and wings. In: Recent Development of Aerodynamic Design Methodologies. Springer (1999) 71–87
18. Kulfan, B.M.: Universal parametric geometry representation method. Journal of Aircraft **45**(1) (2008) 142–158

Particle Swarm Optimization with Levenberg Marquardt Algorithm for Artificial Neural Network Training

Betul Aygun[1] and Ahmet Cosar[2]

[1] Middle East Technical University, Informatics Institute, Ankara, Turkey
[2] Middle East Technical University, Computer Engineering, Ankara, Turkey

Abstract. Artificial neural network training is an important technique in terms of the similarity of human brains. Evolutionary algorithms, swarm intelligence algorithms, single solution based algorithms are algorithms used for ANN training. They have some advantages and disadvantages. The aim of this paper is to develop an algorithm that combines the particle swarm optimization (PSO) with the Levenberg-Marquardt algorithm. By using PSO technique, optimum particle is get and by using this particle, proposed algorithm train the neural network by using the back propagation algorithm. Each backpropagation algorithms are different in terms of storage and computational requirements. Since reducing train time and increasing model accuracy is the most important metric while evaluating ANN training algorithms, Levenberg-Marquardt algorithm is selected because of the shorting duration and more satisfactory performance.

Keywords: Artificial Neural Network, backpropogation algorithm, particle swarm optimization

1 Introduction

Artificial Neural Networks (ANN) offer several advantages like less statistical training, many training algorithms [1][2]. Backpropagation algorithms are most commonly used algorithms for training ANN. However, they have some disadvantages like that their successes are based on learning parameters. These ones are probably stuck in local optima solution especially for complex functions [2, 3]. In addition to these, backpropagation algorithms suffer from extensive calculation and so cause to lower convergence speed [4]. Implementing PSO technique is easier and it needs few parameters than the other algorithms like genetic algorithm (GA) [5, 6]. Furthermore, convergence speed of GA may become too slow when coding chromosomes with more genes to enhance the accuracy of the algorithm when a problem is complex and needs many parameters [3, 7]. However, PSO has also some disadvantages; easily falling into a local optimum in contrast to backpropagation which finds global optima easily [3,8,9]. While training neural networks, it is considered that LM algorithm has the highest speed among others [10, 11, 12]. In conclusion, the aim of the proposed algorithm in this paper is using both PSO and BP algorithms which have strong searching ability globally and locally respectively [3]. Four data sets; Iris, Cancer,

Diabetes and Thyroid are used to show the results. The main contribution of this work is that new algorithm is designed to overcome deficiencies of the existing algorithms. The remainder of the paper is organized as follows: In the following section, proposed algorithm is described. Section 3 presents us the results of our experiment. Section 4 presents us the conclusion of our study and discusses about future work.

2 Proposed Algorithms

In this part, proposed algorithm which optimizes the weights of ANN with the combination of PSO algorithm is explained. Proposed algorithm is a combination of PSO and LM algorithms. For the LM backpropagation algorithm, fully connected layered feed forward networks are used. Apart from input units, all units have a bias. In the subsections, PSO with time varying acceleration coefficients, opposition based learning components, Levenberg Marquardt Algorithm and proposed algorithm that combines them will be detailed.

PSO with Time-varying Acceleration Coefficients

Particles for PSO algorithm are defined as the combination of weights and biases of the neural network;$p = \{w_1, w_2, w_3, w_4, b_1, w_5, w_6, b_2, b_3, \}$where w`s are weights for ANN and b`s are bias defined for hidden and output layer.

Due to the fact that the choice of the parameters of the PSO algorithm has enormous effects on the performance and accuracy, selecting well PSO parameters is another major search area. If cognitive component has higher value, particles travelling with no preset route i.e. result in excessive wandering of particles [13]. If social component has higher value, particles converge prematurely [13]. To overcome this problem, time varying parameters are used. 0.5 is for c1(m) and c2(1) and 2.5 for c1(1) and c2(m) are selected as the most effective values where m is the maximum number of iterations [13, 14].

Opposition based Learning Components

Opposition based learning components concept is based on evaluation of opposite value of candidate solution to provide another chance for finding global optimum solution [15].

Proposed Algorithm Details

Table 1. Algorithm Details of Proposed Algorithm

	Algorithm: Neural Network with PSO Algorithm
1:	Initialize number of iteration, inertia and social and cognitive constants for PSO algorithm
2:	Initialize PSO population randomly according to input architecture, init_pop
3:	FOR each particle in population
4:	Compute opposite values of each particle
5:	Evaluate the fitness value of both particle and opposite value of particle
6:	IF fitness value of particle is better than fitness value of opposite value, add particle to population, init_pop
7:	ELSE add opposite particle to population, init_pop
8:	END IF
9:	END FOR
10:	RUN PSO algorithm with the init_pop
11:	FOR each iteration
12:	Update social and cognitive constants
13:	END FOR

14:	START ANN training with the initial weights and biases decided by PSO algorithm
15:	END

3 Results

To evaluate the accuracy of the proposed algorithm, the following equation is used. For the algorithm execution, MATLAB NNET toolbox is used. In this study, dataset is divided into two sets which are for training and testing.

$$Accuracy(ACC) = \frac{TC_1 + TC_2 + TC_3}{TC_1 + FC_1 + TC_2 + FC_2 + TC_3 + FC_3}$$

Results of Cancer: In figure 1, 0 is for not being cancer and 1 is for being cancer. Only 2 data is not predicted correctly.
Results of Diabetes: Although 28 people are diabetic, they predicted as not diabetic. Accuracy Rate is 0.7969.
Results of Iris: Accuracy is 0.9733. Only 2 data is predicted wrong in test data set. All test data except 2 of them are correctly predicted.
Results of Thyroid: Accuracy is 0.9641. 72 data is predicted wrong in test data. However, since the data size is very high, accuracy rate is also enough high.
Comparison between different neural network training algorithms

We compare the three neural network training algorithms (Levenberg marguard, gradient descent and gradient descent with momentum) in terms of accuracy metric. Training neural network with LM yields better results than the other methods. Also, when we look at the iteration numbers of neural network, LM algorithm iterates less than the other two algorithms which decreases the execution time.

Table 6. Comparison with other training methods

Data Set	PSO with Levenberg-Marquardt	PSO with Gradient Descent	PSO with Gradient Descent with Momentum
Cancer	1000 iterations - 0.9771	57 iterations - 0.7650	193 iterations - 0.7650
Diabetes	41 iterations - 0.7969	48 iterations - 0.6797	144 iterations - 0.6797
Iris	33 iterations - 0.9733	1000 iterations - 0.8933	1000 iterations - 0.9200
Thyroid	89 iterations - 0.9641	1000 iterations - 0.9252	1000 iterations - 0.9252

4 Conclusion

To sum up, we have propose a new modified method that weights of nodes of neural network is decided by using PSO and network is trained by using Levenberg Marquardt technique. We realized that the proposed method yields better results than the other popular training methods for artificial neural networks. For the future work, we can apply our method to different data sets and measure the accuracy of the proposed method with extensive statistical techniques. Also, neural network is designed in a simplest way. It is also modeled complexly to increase the accuracy of the results and efficiency of the algorithm.

References

1. Tu JV. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. J Clin Epidemiol, 1996, 49, 1225–31.
2. M. Yaghini, MM. Khoshraftar, M. Fallahi. A hybrid algorithm for artificial neural network training. Eng Appl Artif Intell (article in press,. Available online 23 Mar 2012.
3. H. Shayeghi, H.A. Shayanfar and G. Azimi. A Hybrid Particle Swarm Optimization Back Propagation Algorithm For Short Term Load Forecasting. In Proc. IJTPE, 2010, 2,12-22.
4. S. Mohagheghi, Y. Del Valle, G. Venayagamoorthy, and R. Harley. A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with STATCOM. In Proc. IEEE Swarm Intell. Symp., Jun. 2005, 381–384.
5. Y. Karpat, T. Özel. Swarm-intelligent neural network system (SINNS) based multi-objective optimization of hard turning. Trans NAMRI/SME, 2006, 34,179–186.
6. M. Clerc and J. Kennedy. The Particle Swarm Explosion, Stability, and Convergence in a Multidimentional Complex Space. IEEE Trans. On Evolutionary Computation, 2002, 6, 58- 73.
7. H. Shayeghi, A. Jalili and H.A. Shayanfar. Multi-Stage Fuzzy Load Frequency Control Using PSO. Energy Conversion and Management, 2008, 49, 2570-2580.
8. J.R. Zhang, J. Zhang, T.M. Lok and M.R. Lyu. A Hybrid Particle Swarm Optimization Back Propagation Algorithm for Feed-forward Neural Network Training. Applied Mathematics and Computation, 2007, 185, 1026-1037.
9. Y. Shi. Particle Swarm Optimization", Electronic Data Systems. Inc. IEEE Neural Networks Society, 2004, 8-13.
10. B. M. Wilamowski , Y. Chen, and A. Malinowski. Efficient algorithm for training neural networks with one hidden layer. In Proc. IJCNN, 1999, 3, 725-728.
11. K.Kipli, M. S. Muhammad, Sh. Masniah Wan Masra, N. Zamhari, K. Lias, D. Azra, A. Mat, Performance of Levenberg- Marquardt Backpropagation for Full Reference Hybrid Image Quality Metrics. IMECS, vol.1, Hong Kong, 2012.
12. H.K. Cigizoglu, O. Kisi. Flow prediction by three back propagation techniques using k-fold partitioning of neural network training data. Nordic Hydrology, 36 (1) (2005), pp. 49–
13. J. Kennedy, R. Eberhart. Particle swarm optimization. In: Proceedings of the IEEE conference on neural networks (ICNN'95), vol. IV. Perth, Australia; 1995. pp.42–48.
14. K.T. Chaturvedia, M. Panditb, L. Srivastavab. Particle swarm optimization with time varying acceleration coefficients for non- convex economic power dispatch. International Journal of Electrical Power & Energy Systems 2009, vol.31, pp. 249-257
15. H. Wang, Z.J. Wu, S. Rahnamayan, Y. Liu, M. Ventresca. Enhancing particle swarm optimization using generalized opposition-based learning. Inform. Sci., 181 (20) (2011), pp. 4699–4714

Indicator-based versus Aspect-based Selection in Multi- and Many-objective Biochemical Optimization

Markus Borschbach[1] and Susanne Rosenthal[1]

[1] University (FHDW)
Hauptstr. 2 51465 Bergisch Gladbach - Germany

Abstract. The identification of qualified peptides as ligands for diagnostic and therapeutical interventions requires the compliance of multi- and many-objective biochemical optimization compositions. Various MOEAs have been designed and achieve competitive performance on multi-objective problems, but encounter considerable challenges in many-objective problems. Pareto-based MOEAs slow down in convergence as the classification of the solutions' quality according to the Pareto dominance principle becomes increasingly undifferentiated with the rise of objectives. This research compares a MOEA evolved for drug design with an indicator-based selection to an enhanced version of this algorithm with a general aspect-based selection strategy. This strategy selects individuals for the succeeding generation by meta-modeling of the optimization problem. The performance of the indicator-based and aspect-based selection are analyzed on generic 3- to 6-dimensional physiochemical optimization problems and the impact of the reference point in the meta-modeled selection is investigated.

1 Introduction

Peptides have several attractive features as they are highly selective, of low toxicity and immune reactions, have a high-affinity and are very effective in binding to targets. Due to these facts, peptides are highly suitable as applicants for diagnostic and therapeutic agents. Peptides as drug components have to fulfill several additional properties simultaneously like a good cell permeability, high stability of the molecule and insoluble in aqueous solutions.[1]

The synthesis and laboratory analysis of peptides is a time and cost consuming process. A MOEA for molecular optimization, referred to as COSEA-MO, has been recently reported and benchmarked in [2] identifying a selected number of highly qualified molecules within a very low number of generations in the case of 3- and 4-dimensional physiochemical optimization problems. COSEA-MO is evolved to improve an *in vitro* drug design process as a computer-assisted system to identify a selected number of improved molecules providing a wide range of genetic diversity within a very low iteration number for an efficient laboratory examination. Dynamic deterministic variation operators are used in COSEA-MO and a mating pool of the old population and the offspring is generated after variation. A combination of fitness-proportionate and indicator-based selection determines the individuals of the succeeding generation. The Pareto dominance principle as a part of the selection strategy potentially induces problems in the case of Many-objective Optimization Problems (MaOPs).

Multi-objective Optimization Problems (MOPs) with more than three objectives are usually referred to as MaOPs. They pose a challenge for MOEAs in the both targets, convergence and diversity. Pareto-based MOEAs have difficulties to solve MaOPs due to their inability to classify the quality of solutions by the Pareto dominance principle. Furthermore, the definition of diversity is less straightforward to reformulate in MaOPs.[3]

A more sophisticated selection strategy was recently introduced in [4] to enhance COSEA-MO for many-objective molecular optimization problems. This selection strategy applies the Pareto dominance principle not directly to the optimization problem, but to a two-dimensional indicator problem covering two generic aspects of molecular optimization problems: firstly, an indicator for the quality of the peptides; secondly, an indicator for the genetic dissimilarity of a peptide with regard to the current population. The performance of COSEA-MO with the indicator-based and the aspect-based selection strategy, further termed nCOSEA-MO, are analyzed in the case of generic 3- to 6-dimensional physiochemical optimization problems as well as the impact of the reference

point in the aspected-based selection strategy. The results are discussed regarding the quality of the identified peptides, the distribution of the solutions over the search space and the specifications in the search behavior.

2 Related work

MOEAs nowadays are categorized as Pareto-based, decomposition-based and indicator-based methods and have a high potential to achieve excellent performance in optimization problems with two or three objectives. Otherwise, MOEA have substantial difficulties to solve MaOPs [11]. These difficulties are to be found in the selection operators, the computational cost and visualization of the solutions. Pareto-based MOEA like NSGA-II [12] experience a low efficiency in terms of convergence as the selection criteria of NSGA-II is primary Pareto-based. The consequence is a significant increase of the non-dominated solutions as the Pareto principle has difficulties in distinguishing the individuals of a population. As the term convergence is neglected, diversity is predominant, see e.g. [25]. For decomposition-based methods like MOEA/D [13], assigning of weight vector values or a reference point in high dimensions is challenging. Indicator-based problems like HypeE [14] produce highly increasing computational complexity caused by the hypervolume indicator. Improvement of these algorithms for many-objective optimizations have been published addressing the challenge of convergence and diversity by methods of objective reduction, incorporation and preferences, modified dominance definitions and the introduction of additional selection criteria:

Dimensionality reduction methods have been published dealing with redundant objectives: In [15], a technique of selecting a subset of conflicting objectives using a correlation-based ordering of objectives is presented. In [16], objective reduction is formulated as a multi-objective search problem. Three formulations are introduced of this problem: two formulations base on preservation of dominance structure and one formulation utilizes the correlation between the objectives. NSGA-II is applied to generate Pareto front subsets that offer decision support to the user.

Preference-based many-objective evolutionary algorithms are developed providing a decision-maker search for user's preferred solutions. In [17], a brushing method is proposed to focus on a subset of Pareto optimal solutions on user's preference. In [18], a preference-inspired coevolutionary algorithm is proposed applying the concept of a set of decision-makers preferences together with a population of candidate solutions.

Alternative Pareto dominance principles have been proposed modifying the definition of Pareto dominance. Alternative rules such as ϵ-dominance [19], L-dominance [20], fuzzy- [21] and grid-dominance [22] have been published.

An established and improved MOEA for many-objective optimization is NSGA-III [23]. The primary Pareto-based selection of NSGA-II is improved by using the non-dominated sorting for the first aspect and a more complex niching operator based on a set of predefined reference directions, termed weight vectors, to address diversity. It is a challenge to design the weight vectors in real-world applications. Furthermore, MOEA/DD and Two-Arch2 achieved excellent performance in MaOPs [24]. MOEA/DD uses the Pareto dominance principle and decomposition, Two-Arch2 is also based on Pareto dominance and an indicator.

3 Designed MOEA for molecular optimization

The presented COmponent-Specific Evolutionary Algorithm for Molecule Optimization (COSEA-MO) [2] is designed to complement an *in vitro* drug design process with a computer-assisting system aimed at the specific requirements of such combined *in vitro* and *in silico* process: Firstly, several molecular properties are not predictable by numerical approximation models or descriptor value sets and have to be determined in an *in vitro* process. As a consequence, the evolutionary process has to provide a selected number of high-qualified peptides within a very low number of generations and objective evaluations. Secondly, the proposed optimized peptides have to be highly diverse in its primary genetic structure and therefore, the algorithm has to propose the whole range from very similar to very diverse peptide sequences in each of the low iteration. Thirdly, the algorithm has to be independent of problem-specific parameters as these are either usually unknown or expert rule of thumbs in real-world application problems.

The algorithms briefly described in the following makes use of a combination of deterministic dynamic variation operators and a selection strategy for the determination of the individuals for the succeeding generation. The traditional selection concept is tournament-based and a combination of fitness-proportionate and indicator-based selection. The procedure of the proposed algorithm is similar to NSGA-II. The initial population of COSEA-MO is generated by N random individuals. Individuals are selected randomly from the actual population for variation. Parent and offspring sets are combined to a set of size $2N$. The succeeding generation of size N is generated by optionally applying either the proposed indicator-based selection strategy (COSEA-MO), or the aspect-based selection procedure (nCOSEA-MO).

The individuals in COSEA-MO represent peptides of length 20 consisting of the 20 canonical amino acids. The individuals are encoded as character strings. As a consequence, the search space has a complexity of 20^{20}. This encoding present all feasible and only feasible solutions, which have an equal probability to be presented. Tools for the determination of physiochemical peptide properties often make use of this character encoding. Therefore, this encoding does not require a conversion of the data format based on a character set representing an amino acid chain.

The mutation and recombination operator are motivated by a suitable balance of global and local search. Especially high mutation rates in early generations support the explorative search behavior and lower mutation rates in later generations support the exploitive search. Deterministic dynamic variation operators are suitable operators to achieve this purpose. The characteristic of deterministic dynamic operators is the adaptation of mutation and recombination rates by a predefined functional reduction with the iteration progress. This is implemented by deterministic functions depending on the current iteration number, the total number of iterations and the molecule length.
The recombination operator varies the number of recombination points over the generations via a linearly decreasing function:

$$x_R(t) = \frac{l}{2} - \frac{l/2}{T} \cdot t, \tag{1}$$

which depends on the length of the individual l, the total number of the generations T and the index of the current generation t. Three parents are used for recombinations.
An adapted version of the deterministic dynamic operator of Bäck and Schütz [**?**] determines the mutation probabilities via the following function with $a = 5$

$$p_{BS} = (a + \frac{l-2}{T-1}t)^{-1}, \tag{2}$$

The mutation rates of the traditional operator have been adapted to a lower starting mutation rate by the parameter $a = 5$.

3.1 Indicator-based selection strategy

The traditional selection concept of COSEA-MO is depicted in Fig 1. It starts with the tournament selection of ts individuals from the population. These individuals are ranked according to the Pareto dominance principle and the volume of each individual to the zero point as ideal reference point is calculated. From this ranked tournament set, the individuals with the lowest volume values are selected for the succeeding generation with a probability p_0, with the aim of guiding the search process in direction of high quality solutions. With a probability $1 - p_0$, the individuals are chosen from different fronts via SUS. The number of pointers in front-based SUS is equal to the number of fronts detected in the ranking process. The segments are equal in size to the number of individuals in each front. These steps repeat until the succeeding filial generation is complete. Consequently, this selection strategy has two parameters, the tournament size and the probability p_0 for choosing the individuals from the first front. Default values are $ts = 10$ and $p_0 = 50\%$.

3.2 General-aspect-based selection

This section describes the alternative aspect-based selection strategy. A MaOP is given by $f : P \longrightarrow \mathbb{R}^m$, $p \longrightarrow (f_1(p), f_2(p), ..., f_m(p))$, whereby $m > 3$ is the number of objectives f_i as molecular functions which have to be minimized, and P is the quantity of feasible molecules. The

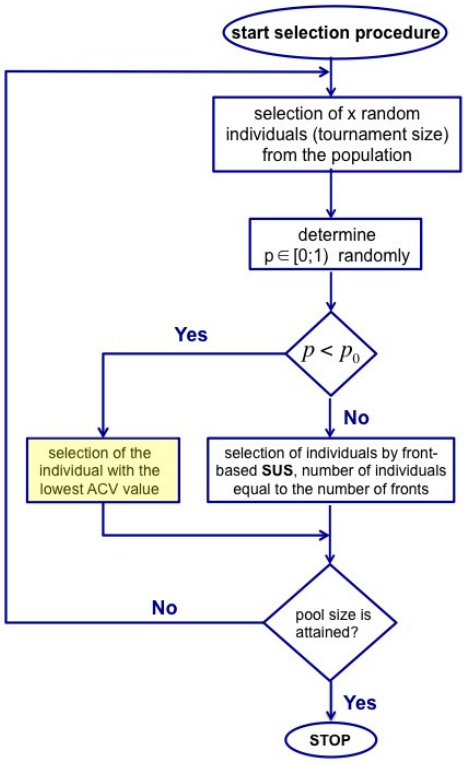


Fig. 1. Indicator-based selection strategy

procedure of the novel selection strategy is described in Algorithm1. The strategy is ranked and binary tournament based. The Pareto principle used for ranking is not directly applied on the objective values but on a two-dimensional aspect-based minimization problem (line 4). The first aspect reflects the solutions' quality by the calculation of the L_p-norm of the objective values to a reference point (RP) (line 2), which is either determined by the minimum of each objective provided by the population members (line 1). Therefore, this reference point varies with the population. Alternatively, in the experiments the zero point is selected as ideal reference point RP. The second aspect refers to the general idea of maintaining a high genetic dissimilarity within the populations. Needleman Wunsch Algorithm [10] is chosen as global sequence alignment (line 3).

Algorithm 1: Pseudo code of the aspect-based selection strategy

Input: Current population P_t with $|P_t| = 2N$, $P_{t+1} = \{\}$
Calculation of the two indicator values for each solution:
1: $RP := (min_{i_1} f_1(p_{i_1}), min_{i_2} f_2(p_{i_2}), ..., min_{i_m} f_m(p_{i_m}))$;
2: $\forall p \in P_t$: $f_{L_{p-norm}}(p) = L_p(f(p), RP)$;
3: $\forall p \in P_t$: $diss(p) = \frac{1}{|P_t|} \sum_{p \in P_t} SequenceAlignment(p, P_t - p)$;
Selection process:
4: Ranking of P_t according to $(f_{L_p-norm}, diss)$ into fronts F_i;
5: **while** $|P_{t+1}| + |F_i| < N$ **do**
  $P_{t+1} = P_{t+1} \cup F_i$; i++;
end
6: binary tournament selection: **while** $|P_{t+1}| < N$ **do**
  select $p_1, p_2 \in P_t \setminus \{P_{t+1}\}$:
  if $(f_{L_p}(p_1) * diss(p_1) < f_{L_p}(p_2) * diss(p_2))$ add p_1 to P_{t+1} ;
  else add p_2 to P_{t+1};
end

The N-best individuals are selected in the succeeding generation based on the rank (line 5) and the volume dominance principle via binary tournament selection (line 6). L_2-norm is used here in the experiments.

4 Experimental studies

This section introduces and motivates the generic physiochemical optimization problems, describes the simulation onsets and presents the experimental results. For performance comparison and discussion of the different selection configurations, four optimization problems with 3 up to 6 objective functions are applied predicting physiochemical peptide properties. These optimization problems are generic in the sense that the physiochemical properties of molecules are determined by descriptor values of each amino acid in the molecule sequence.

4.1 Physiochemical optimization problems

The optimization problems are composed of a basic peptide design composition comprising the charge, solubility in aqueous solutions and molecule size, as well as four advanced peptide design compositions with additional properties like molecule stability and structure. The six physiochemical functions work on the primary amino acid sequence of the peptides and are provided by the open source BioJava library [6]. A description of the determination methods of the following physiochemical properties is also available here [6]:

1. Needleman Wunsch Algorithm (NMW)
2. Molecular Weight (MW)
3. Average Hydrophilicity (Hydro)
4. Instability Index (InstInd)
5. Isoelectric Point (pI)
6. Aliphatic Index (aI)

NMW is a well known and used method for the global sequence alignment of a solution to a predefined reference individual. This algorithm refers to the common hypothesis that a high similarity between molecules refers to similar molecular properties [7].

MW is an important peptide property as a minimized MW ensures a good cell permeability. MW of a peptide sequence of the length l is determined by the sum of the mass of each amino acid (a_i) plus a water molecule:

$$\sum_{i=1}^{l} mass(a_i) + 17.0073(OH) + 1.0079(H), \tag{3}$$

where O (oxygen) and H (hydrogen) are the elements of the periodic system.

A common challenge of drug peptides is the solubility in aqueous solutions, especially peptides with stretches of hydrophobic amino acids. Therefore, Hydro is calculated by the hydrophilicity scale of Hopp and Woods [8] with a window size equal to the peptide length l. An average hydrophilicity value is assigned to each candidate peptide using the scales for each amino acid a_i:

$$\frac{1}{l} \cdot (\sum_{i=1}^{l} hydro(a_i)). \tag{4}$$

The use of molecules as therapeutic agents is potentially restricted by their instability and their potential degradation by enzymes in systemic application. The stability is addressed by the InstInd as stability is a very important feature of drug components. InstInd is determined by the Dipeptide Instability Weight Values (DIWV) of each two consecutive amino acids in the peptide sequence. DIWV are provided by the GRP-Matrix [9]. These values are summarized and the final sum is normalized by the peptide length l:

$$InstInd = \frac{10}{l} \sum_{i=1}^{l} DIWV(x_i, x_{i+1}) \tag{5}$$

pI of a peptide is defined as the pH at which a peptide has a net charge of zero. A peptide has its lowest solubility at its pI. Therefore, the charge of a peptide influence the solubility in aqueous solutions. The pI value is calculated as follows: Firstly, the net charge for $pH = 7.0$ is determined. If this charge is positive, the pH at $7 + 3.5$ is calculated; otherwise the pH at $7 - 3.5$ is determined. This process is repeated until the modules of the charge is less or equal 0.0001.

aI of a peptide is defined as the relative volume occupied by aliphatic side chains consisting of the amino acids alanine (Ala), valine (Val), isoleucine (Ile) and leucine (Leu). aI is regarded as a positive factor for the increase of thermostability of globular molecules. aI is calculated according to the formula:

$$aI = X(Ala) + a \cdot X(Val) + b \cdot (X(Ile) + X(Leu)), \tag{6}$$

where $X(Ala)$, $X(Val)$, $X(Ile)$ und $X(Leu)$ are mole percent of the amino acids. The coefficients a and b are the relative volume at the valine side chain ($a = 2.9$) and Lei, Ile side chains ($b = 3.9$) to the side chain Ala. Table 1 presents the composed physiochemical optimization compositions

Table 1. Physiochemical functions of the different optimization problems

dimension	abbr.	objective functions
3D	3D-MOP	NMW, MW, Hydro
4D	4D-MaOP	NMW, MW, Hydro, InstInd
5D	5D-MaOP	NMW, MW, Hydro, InstInd, pI
6D	6D-MaOP	NMW, MW, Hydro, InstInd, pI, aI

with the abbreviations used in the following. These six objective functions comparatively act to reflect reflect the similarity of a particular peptide and a pre-defined reference peptide:

$$f(\text{CandidatePeptide}) := |f(\text{CandidatePeptide}) - f(\text{ReferencePeptide})| \tag{7}$$

Therefore, the four objective functions have to be minimized and the optimization problems are minimization problems.

4.2 Simulation onsets

The experiments are generally performed with the default population size of 100 motivated by previous experimental studies, the start population is randomly initialized. The individuals are 20-mer peptides composed of the 20 canonical amino acids. For statistical reasons, each configuration is limited by 10 generations and is repeated 30 times. Firstly, the approximate Pareto optimal sets (PFs) of COSEA-MO and nCOSEA-MO in each generation are compared in terms of the established C-metric [5]

$$C(PF_1, PF_2) := \frac{|\ \{b \in PF_2 \mid \exists a \in PF_1 : a \preceq b\}\ |}{|\ PF_2\ |}. \tag{8}$$

$C(PF_1, PF_2) = 0$ means that no solution of PF_2 is weakly dominated by at least one solution of PF_1, whereas $C(PF_1, PF_2) = 1$ implicate that all points of PF_2 are weakly dominated by PF_1. This metric is usually not symmetric, therefore $C(PF_1, PF_2)$ is not a metric in a mathematical sense and consequently $C(PF_1, PF_2)$ and $C(PF_2, PF_1)$ have to be determined. Therefore, the C-metric value reflects the percentage of solutions which are weakly dominated by one individual of the other approximate Pareto set.

PF of COSEA-MO is determined according to the molecular optimization problem, whereas PF of COSEA-MO is determined according to the two-dimensional meta-modeled problem. The C-metric values are determined according to the objective values as usual. Table 2 to 10 depict the C-metric values $C_1 = C$(nCOSEA-MO, COSEA-MO) and $C_2 =$C(COSEA-MO, nCOSEA-MO) for the 3D-MOP to 6D-MaOP with different selection parameter settings p_0 and reference point (RP) based on empirical and experimental findings.

Table 2. 3D-MOP: $p_0 = 50\%$ and $RP = min$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.88	0.87	0.9	0.9	0.81	0.73	0.73	0.53	0.68	0.7
C_2	0.69	0.75	0.73	0.65	0.68	0.61	0.69	0.7	0.56	0.55

Table 3. 3D-MOP: $p_0 = 50\%$ and $RP = 0$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.78	0.83	0.83	0.79	0.64	0.67	0.69	0.58	0.6	0.6
C_2	0.72	0.69	0.66	0.64	0.65	0.55	0.55	0.68	0.48	0.47

Table 4. 4D-MaOP: $p_0 = 50\%$ and $RP = min$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.55	0.55	0.6	0.63	0.64	0.53	0.67	0.58	0.7	0.6
C_2	0.55	0.45	0.44	0.51	0.46	0.53	0.51	0.48	0.5	0.56

Table 5. 4D-MaOP: $p_0 = 50\%$ and $RP = 0$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.71	0.65	0.7	0.72	0.64	0.65	0.46	0.62	0.52	0.32
C_2	0.51	0.55	0.56	0.46	0.5	0.49	0.54	0.54	0.46	0.52

Table 6. 5D-MaOP: $p_0 = 50\%$ and $RP = min$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.48	0.36	0.38	0.38	0.33	0.29	0.24	0.22	0.18	0.22
C_2	0.6	0.6	0.65	0.64	0.63	0.6	0.59	0.66	0.62	0.64

Table 7. 5D-MaOP: $p_0 = 50\%$ and $RP = 0$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.5	0.45	0.35	0.32	0.3	0.22	0.3	0.23	0.21	0.17
C_2	0.59	0.66	0.63	0.63	0.68	0.67	0.68	0.62	0.54	0.55

Table 8. 5D-MaOP: $p_0 = 70\%$ and $RP = min$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.4	0.33	0.34	0.35	0.3	0.2	0.18	0.15	0.12	0.18
C_2	0.59	0.52	0.62	0.61	0.61	0.56	0.56	0.64	0.58	0.62

Table 9. 5D-MaOP: $p_0 = 70\%$ and $RP = 0$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.55	0.45	0.32	0.31	0.26	0.16	0.3	0.2	0.16	0.08
C_2	0.57	0.57	0.62	0.56	0.58	0.55	0.55	0.46	0.47	0.5

Table 10. 6D-MaOP: $p_0 = 50\%$ and $RP = min$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.32	0.26	0.43	0.29	0.26	0.24	0.25	0.25	0.22	0.25
C_2	0.51	0.55	0.76	0.51	0.52	0.51	0.45	0.54	0.61	0.54

Table 11. 6D-MaOP: $p_0 = 50\%$ and $RP = 0$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.4	0.4	0.41	0.38	0.37	0.36	0.28	0.19	0.28	0.37
C_2	0.57	0.61	0.72	0.6	0.46	0.55	0.52	0.6	0.56	0.48

Table 12. 6D-MaOP: $p_0 = 70\%$ and $RP = min$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.34	0.3	0.36	0.3	0.22	0.23	0.21	0.15	0.16	0.14
C_2	0.46	0.5	0.42	0.48	0.56	0.58	0.56	0.55	0.6	0.54

Table 13. 6D-MaOP: $p_0 = 70\%$ and $RP = 0$

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
C_1	0.4	0.38	0.36	0.37	0.35	0.31	0.23	0.14	0.2	0.25
C_2	0.53	0.54	0.48	0.53	0.52	0.59	0.57	0.61	0.52	0.5

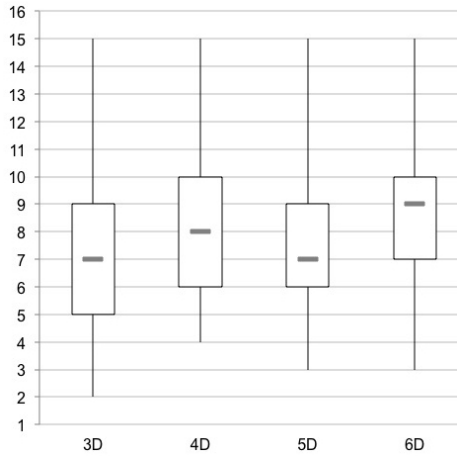


Fig. 2. Number of Pareto optimal solution in the case of nCOSEA-MO.

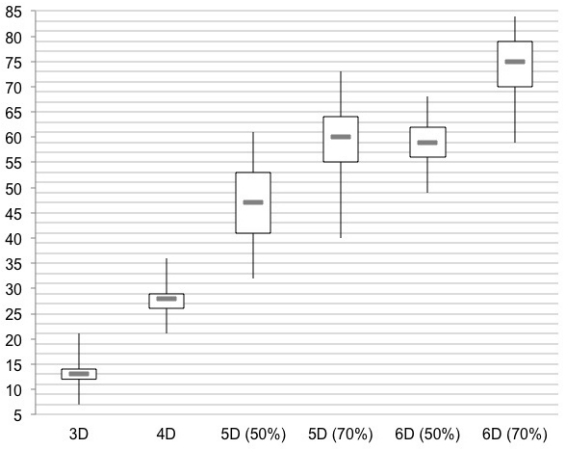


Fig. 3. Number of Pareto optimal solution in the case of COSEA-MO.

4.3 Experimental results

The experimental results are analyzed according to the following questions: firstly, does a higher problem dimension has a different impact on the performance of either or both selection strategies,

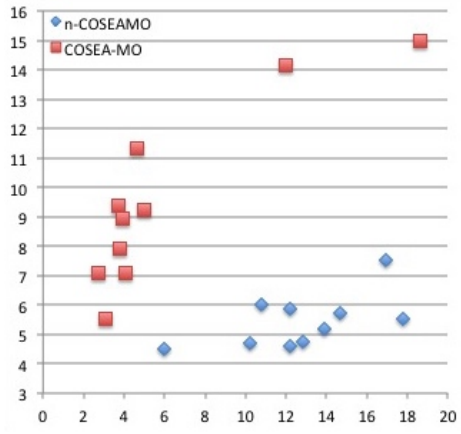


Fig. 4. Peptide quality in the case of 3D-MOP.

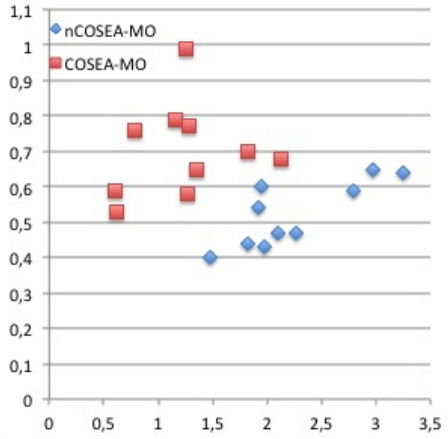


Fig. 5. Peptide quality in the case of 4D-MaOP.

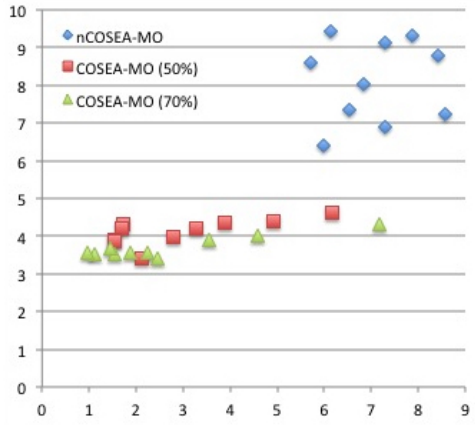


Fig. 6. Peptide quality in the case of 5D-MaOP.

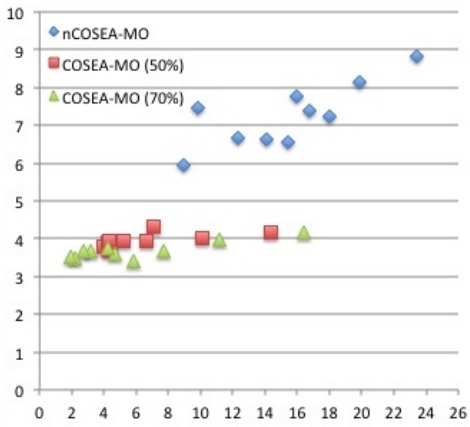


Fig. 7. Peptide quality in the case of 6D-MaOP.

indicator- or aspect-based? Secondly, is there an impact of the selection parameters p_0 or RP observable? Thirdly, is there a fundamental difference in the search behavior of the different selection configurations?

Table 3-13 depict the C-metric values of nCOSEA-MO and COSEA-MO with different parameter settings on the four optimization problems. A significant difference is observable in comparison of Table 3-6 with Table 7-13: in the case of 3D-MOP and 4D-MaOP, C_1 values are generally higher than those of C_2, revealing that more candidate solutions identified by nCOSEA-MO weakly dominate the solutions of COSEA-MO than vice versa within each of the 10 generations. Otherwise, in the case of 5D-MaOP and 6D-MaOP, C_2 values are generally higher than those of C_1, revealing that more solutions of COSEA-MO weakly dominate solutions of nCOSEA-MO independent of the parameter settings. Figure 2 and 3 give an insight into the number of approximate Pareto optimal solutions identified by COSEA-MO and nCOSEA-MO in the test runs. nCOSEA-MO provides a significantly lower and stable number of candidate solutions, whereas the solutions number of COSEA-MO is generally higher and increases with the problem dimension as a consequence of the Pareto dominance principle being directly applied to the objective values. The number of Pareto optimal solutions of COEA-MO is increasing linearly from 3D-MOP to 4D-MaOP and this increasing rate slows down by a further dimension increase. The average number of identified Pareto optimal solutions by COSEA-MO is generally higher in the case of $p_0 = 70\%$ than those of $p_0 = 50\%$. As the approximate Pareto optimal sets of COSEA-MO are significantly larger than those of nCOSEA-MO, the probability of identified promising peptides in the sets of COSEA-MO is clearly higher than in the case of nCOSEA-MO.

Referring to the second question, there is no significant impact of the choice of RP in the case of the aspect-based selection. Consequently, RP has no impact on the selection pressure independent of the problem dimension. In the case of 5D-MOP and 6D-MaOP, the probability of p_0 and therefore the selection probability of the individuals for the succeeding generation by the indicator is increased with the aim of raising the selection pressure. A slight influence of the raise is observable as the C-metric values are mainly slightly lower in the case of $p_0 = 70\%$, revealing that more peptides of the approximate Pareto optimal sets are indifferent to each other.

To address the third question, a specific mapping method is applied to visualize the peptide quality of each generation equally for all optimization problems with the aim of analyzing the search behavior. The peptide quality of each generation is mapped into a point $(x; y)$, where x is the volume of the gravity point of the scatter plot to zero point as an ideal reference point. The coordinate y is the standard deviation of the scatter plot points to the calculated reference point symbolizing the average distance of the solutions points in each generation to the calculated gravity point of these solutions. Scatter plots of these points are depicted in Figure 4 to 7 for each optimization problem. Generally, it is observable that nCOSEA-MO has higher volumes of the gravity point in the generations independent of the problem dimension but lower standard deviations of the peptides in each generation to the gravity point in the cases of 3D-MOP and 3D-MaOP. Figure 6 and 7 reveal that the results of nCOSEA are generally higher in the volume of the gravity points and die standard deviation. The results of COSEA-MO with the probabilities $p_0 = 50\%$ and $p_0 = 70\%$ are generally comparable in volume and standard deviation but remarkably low in both terms compared to nCOSEA-MO. The volume values of the configuration with $p_0 = 70\%$ are slightly lower, a consequence of the higher probabilities of the individuals to be selected into the succeeding generation by the ACV-indicator. The peptides identified by COSEA-MO are more clustered in the search space, which is a known property of the ACV-indicator. As a consequence, the indicator-based selection tends to identify higher quality solutions in the case of the 5D- and 6D-MaOP. nCOSEA-MO provides improved performance in the case of the 3D-MOP and 4D-MaOP. The analysis of the identified peptides of nCOSEA-MO and COSEA-MO according to their physiochemical function values reveals an interesting fact: the peptides identified by nCOSEA-MO are generally of significantly lower MW, but have a higher average hydrophilicity and in the cases of 5D- as well as 6D-MaOP higher pI values. As a consequence of the second aspect, the identified peptides of nCOSEA-MO tends to have lower NMW values and therefore provide a higher similarity to the reference peptide.

5 Discussion and conclusion

This works presents the performance comparison of a specific MOEA for molecular optimization with optionally two different selection strategy in four multi- and many-objective physiochemical optimization problems. The analysis of the results reveal that nCOESA-MO with the aspect-based selection provides a performance improvement in the case of the 3D-MOP and 4D-MaOP compared to COSEA-MO with the indicator-based selection according to the C-metric values. The performance is measured according to the C-metric values of the approximate optimal solution sets identified by nCOSEA-MO and COSEA-MO within 10 generations as well as a closer look on the peptide quality regarding the physiochemical properties by the gravity points of each generation and the standard deviation of the peptides in each generation to this gravity point. In the cases of 5D- and 6D-MaOP, COSEA-MO provides a higher number of qualified peptides compared to nCOSEA-MO having in mind that COSEA-MO has significantly higher approximate optimal solution sets due to the Pareto dominance principle. A further selection method is required to eliminate worse candidate solutions for a subsequent laboratory analysis. Consequently, these optimal sets of COSEA-MO have to be determined by a more sophisticated method in further research work. An interesting point is the difference of the optimal peptides identified by COSEA-MO and nCOSEA-MO regarding the physiochemical properties: the identified peptides of nCOSEA-MO are generally of a significantly lower MW, better NMW values on average but higher average hydrophilicity as well as pI values in higher dimensions. This makes the use of nCOSEA-MO interesting in a practical sense even in the cases of 5D- and 6D-MaOP. A low MW is an important peptide property and therefore referenced objective providing a good cell permeability. Better NMW values indicate a higher similarity to a predefined reference peptide and therefore potentially higher similarity in molecule properties.

A further improvement of nCOSEA-MO is therefore part of the future work as well as the evolution of a combination of these selection strategies for a robust and good performance of COSEA-MO in multi- and many-objective optimization.

References

1. Otvos, L.: Peptide-based Drug Design: Methods and Protocols. Humana Press Inc., 2000.
2. Rosenthal, S., Borschbach, M.: Design Perspectives of an Evolutionary Process for Multi-objective Molecular Optimization. Proc. of the 9th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2017), LNCS 10173 , March 2017, pp. 529-544.
3. Wang H., Jin Y., Yao X.: Diversity Assessment in Many-Objective Optimization. IEEE Transactions on Cybernetics, vol.PP, no.99, pp.1-13, 2016.
4. Rosenthal, S., Borschbach, M.: General Aspect-based Selection Concept for Multi- and Many-Objective Molecular Optimization. International Conference on Genetic an Evolutionary Computation (GECCO'17), pp. 45-46, 2017.
5. Zitzler, E., Thiele, L.: Multiobjective Optimization using Evolutionary Algorithms - A Comparative Case Study. Proc. of the 5th International Conference on Parallel Problem Solving from Nature (PPSN-V), 1998, pp. 292-301.
6. BioJava: CookBook, release 3.0 {http://www.biojava.org/wiki/BioJava}
7. Emmerich, M., Lee, B., Render, A. et al.: Analyzing Molecular Landscapes Using Random Walks and Information Theory. Chemistry Central Journal 3, vol. 1, 2009, pp. 20.
8. Hopp, T.P., Woods, K.R.: A computer program for predicting protein antigenic determinants. Mol. Immunol., 20(4), pp. 483-489, 1983.
9. Guruprasad, K., Reddy, B., Pandit, M.: Correlation between Stability of a Protein and its Dipeptide Composition: A Novel Approach for predicting In Vivo Stability of a Protein from its Primary Structure. Protein Engineering 4(2), pp. 155-161, 1990.
10. Needleman, S.B., Wunsch, C.D.: A General Method Application to the Research for Similarities in the Amino Acid Sequence of Two Proteins. Journal of Molecular Biology 48, vol. 3, 1970, pp.443-453.
11. Ishibuchi, H., Akedo, N., Nojima, Y.: Behavior of Multiobjective Evolutionary Algorithms on Many-objective Knapsack Problems. IEEE Transactions on Evolutionary Computation, vol. 19, no. 2, 2015, pp. 264?283.
12. Deb, K., Pratap, A.,Agarwal, S.,Meyarivan, T.: A fast and elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, 2002, pp. 182-197.
13. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm based on Decomposition. IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, 2007, pp. 712-731.
14. Bader, J., Zitzler, E.: HypE: Analgorithm for Fast Hypervolume-based Many-objective Optimization. Evolutionary Computation, vol. 19, no. 1, 2011, pp. 45-76.
15. Bandyopadhyay, S., Mukherjee, A.: An Algorithm for Many-objective Optimization with Reduced Objective Computations: A Study in Differential Evolution. IEEE Transactions on Evolutionary Computation, vol. 19, no. 3, 2015, pp. 400-413.
16. Yuan, Y., Ong, Y.-S., Gupta, A., Xu, H.: Objective Reduction in Many-objective Optimization: Evolutionary Multiobjective Approaches and Comprehensive Analysis. IEEE Transactions on Evolutionary Computation, 2017, in press, DOI: 10.1109/TEVC.2017.2672668.
17. Wang, R., Purshouse, R.C., Giagkiozis, I., Fleming, P.J.: The iPICEA-g: A New hybrid Evolutionary multi-criteria Decision Making Approach using the Brushing Technique. European Journal of Operational Research, vol. 243, no. 2, 2015, pp. 442-453.
18. Wang, R., Purshouse, R.C., Fleming, P.J.: Preference-inspired Coevolutionary Algorithms for Many-objective Optimization. IEEE Transactions on Evolutionary Computation, vol. 17, no. 4, 2013, pp. 474-494.
19. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. Evolutionary Computation, vol. 10, no. 3, 2002, pp. 263-282.
20. Zou, X., Chen, Y., Liu, M., Kang, L.,: A New Evolutionary Algorithm for Solving Many-objective Optimization Problems. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 38, no. 5, 2008, pp. 1402-1412.
21. Wang,G., Jiang, H.: Fuzzy-dominance and its Application in Evolutionary Many Objective Optimization. In International Conference on Computational Intelligence and Security Workshops (CISW 2007). IEEE, 2007, pp. 195-198.
22. .Yang, S., Li, M., Liu, X., Zheng, J.: A Grid-based Evolutionary Algorithm for Many-objective Optimization. IEEE Transactions on Evolutionary Computation, vol. 17, no. 5, 2013, pp. 721-736.
23. Deb, K., Jain, H.: An Evolutionary Many-objective Optimization Algorithm using Reference-point-based Nondominated Sorting Approach, Part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, 2014, pp. 577-601.

24. Li, B., Li, J., Tang, K., Yao, X.: Many-objective Evolutionary Algorithms: A survey. ACM Computing Surveys (CSUR), vol. 48, no. 1, 2015, p. 13.
25. Wang, H., Jin, Y., Yao, X.: Diversity Assessment in Many-Objective Optimization. IEEE Transactions in Cybernetics PP, 99 (May 2016), pp. 1?13. DOI: h ps://doi.org/10.1109/TCYB.2016.2550502, 2016.

Multi-Objective Design of Time-Constrained Bike Routes using Bio-inspired Meta-Heuristics

Eneko Osaba[1], Javier Del Ser[1,2,3], Miren Nekane Bilbao[2],
Pedro Lopez-Garcia[4], and Antonio J. Nebro[5]

[1] TECNALIA, Derio, Spain
[2] University of the Basque Country (UPV/EHU), Bilbao, Spain
[3] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain
[4] Deusto Institute of Technology (DeustoTech), University of Deusto, Bilbao, Spain
[5] Dept. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain
eneko.osaba@tecnalia.com

Abstract. This paper focuses on the design and implementation of a bike route optimization approach based on multi-objective bio-inspired heuristic solvers. The objective of this approach is to produce a set of Pareto-optimal bike routes that balance the trade-off between the length of the route and its safety level, the latter blending together the slope of the different street segments encompassing the route and their average road velocity. Additionally, an upper and lower restriction is imposed on the time taken to traverse the route, so that the overall system can be utilized for planning bike rides during free leisure time gaps. Instead of designing a discrete route encoding strategy suitable for heuristic operators, this work leverages a proxy software – Open Trip Planner, OTP – capable of computing routes based on three user-level preference factors (i.e. safety, inclination and duration), which eases the adoption of off-the-shelf multi-objective solvers. The system has been assessed in a realistic simulation environments over the city of Bilbao (Spain) using multi-objective bio-inspired approaches. The obtained results are promising, with route sets trading differently distance for safety of utmost utility for bike users to exploit fully their leisure time.

Keywords: Bike route planning, multi-objective optimization, time-constrained routing, Open Trip Planner, jMetal.

1 Introduction

Thanks to the rapid advance of technology, transportation networks have become increasingly complex along the last decade. This fact has led the mobility to be a crucial aspect for society, affecting its quality of life directly. In this way, the necessity for efficient transport means has increased the demand for Intelligent Transportation Systems (ITS), lying at the core of many initiatives focused on shedding smartness and intelligence in different paradigms related to transportation and mobility [1].

Among such paradigms, route planning has gained more and more importance in the last years [2]. It is an undoubted fact that daily transits (e.g. from home to work and return) have become a habit for many people worldwide. In this context, by virtue of well-connected, advanced transport networks, travelers and commuters change from

one transportation mode to another every day. Taking the underground, then the bus, and finishing the travel on foot is an example of the typical traveling routine for many people nowadays. This *multi-modal* transport aims at providing the traveler feasible routes between a certain origin and destination, involving diverse public and private transportation modes connected throughout different schedules [3].

In this context the so-called *mono-modal* route planning, which correspondingly consists of routes performed by a single transport type, are also demanded by people for very diverse purposes, not only by the need for reaching their jobs as postulated above. Mono-modal routes performed by car, bike or foot are often developed for leisure, last-mile packet delivery schedules, and public transport planning, among others [4, 5]. As in any other routing construction process several route planners are available in the market or in the Web to help users design optimal routes according to their needs and requirements. The study presented in this work focuses in this latter case, specifically, in routes performed by bike.

In the last couple of decades a growing number of route planning systems have been developed, which are freely available for the community of users. These tools, mostly accessible from different platforms (with deployable versions for computers, smartphones or tablets), are flexible enough to let users comfortably query routes in any place and time. In all cases, one of the characteristics shared by all route planners is that the provided portfolio of routes are strictly based on parameters that the user enters as an input. However, it is intuitive to think that the user could tolerate a certain degree of flexibility in his/her inputs to the routing system, should this flexibility lead to better routes under a certain optimality criterion (e.g. distance, travel time or exposure to traffic). As surveyed next, the relative scarcity of contributions in the literature exploring the implications of this flexibility in bike routes is what motivates to conduct this study.

1.1 Related Work

Some studies can be found in the literature focused on bike route planning. To begin with, a web-based platform is presented in [6] to help cyclists determine safe and efficient routes. The system developed in that study calculates routes using a weighted combination of five different metrics, which are considered to optimize a trade-off among various safety and distance-related factors. A heuristic multi-modal route planning system is introduced in [7], in which cycling trips are considered. The system in that work enforces the user to select both origin and destination of the route, along with other preferences. One of the tested scenarios is bike route from work to home in an energy-efficient manner. In any case, authors use a single objective aggregating all route metrics.

Most existing planning systems for bike routing do not formulate the optimality of explored routes from a multi-criteria or multi-objective approach, but rather opt for aggregate metric models as the ones mentioned above. As a result, planning systems cannot provide the users with diverse sets of suggested routes, hence narrowing the amount and diversity of information provided to cyclists for their decision making. A few exceptions have been published recently, as in [8], where a multi-criteria bicycle routing problem is tackled. In this work, authors develop a set of heuristics for speeding up the multi-criteria route search. Concretely, the objectives to optimize in this problem are the comfort, duration and inclination of the route. Additionally, the heuristic presented

in that paper is an extension of the standard multi-criteria label-setting algorithm [9]. Another interesting example of this kind is the one proposed by Caggiani *et al.* in [10]. In this work, a multi-objective biking route choice model is proposed for a bike-sharing mobile application. One of the research challenges addressed therein is to offer the user the appropriate starting and ending bike hiring station. To this end, the suggested origin station is set to the nearest one satisfying the requirements of the user, whereas a similar criterion is applied for selecting the destination station. The system developed in [10] informs the user with the starting and ending bike sharing stations, and the best path to follow according to time, distance, pollution and safety. Additionally, users can select an alternative route according to the parameter that they are willing to prioritize.

1.2 Research Contribution

This manuscript aims at contributing to the observed scarcity of references dealing with multi-objective bike route planning by undertaking the design and practical implementation of an bike path planning system for random bike routes generation, grounded on multi-objective bio-inspired optimization heuristics. Several novel ingredients are introduced in our problem formulation, the most relevant being 1) the consideration of lower and upper trip time constraints to model the case where the planning system is used for e.g. leisure/sport; 2) the derivation of a quantitative metric to evaluate the degree of safety associated to a given route with respect to its topological profile and the speed of motor vehicles along its segments; and 3) four different bio-inspired multi-objective solvers (namely, NSGA-II [11], MOEA/D [12], SMS-EMOA [13], and SMPSO [14]) to efficiently balance the Pareto trade-off between the safety level and the distance of the route, always subject to the imposed time constraints. The (pseudo) Pareto-optimal set of routes produced by any of these can be informed to the user if the system so that he/she has the freedom to choose the one that matches best his/her preferences with respect to the considered objectives.

The proposed system has been assessed in a realistic environment using Open Trip Planner (OTP [15]) as the simulation framework. Experimental results from three different use cases located in the city of Bilbao (Spain) are presented and discussed, all using real data sources, namely, the Open Street Map of the city and its Digital Elevation Model (DEM). The analysis of the obtained route portfolios evinces the practicality of the proposed approach, and paves the way towards extending the problem formulation so as to accommodate other manifold route metrics.

The rest of the paper is structured as follows. Section 2 formulates the bi-objective optimization problem considered in this study, whereas Section 3 elaborates on the considered multi-objective heuristics. Section 4 describes in detail the deployed simulation environment. Next, the experimentation performed is shown and discussed in Section 5. Finally, Section 6 ends the paper and outlines future research.

2 Problem Definition

As has been explained beforehand, the problem tackled in this work is the optimization of bike routes, bearing in mind two different objectives (distance and safety of the route) and the compliance with lower and upper time constraints. According to Fig. 1, the scenario model on which this problem is formulated gravitates on the position $(lat^{\circlearrowleft}, lon^{\circlearrowleft})$

where all bike routes depart from, emulating e.g. a bike ride that a user is willing to enjoy from home within his/her limited leisure time. For simplicity in subsequent algorithmic explanations, we assume that $(lat^{\circlearrowleft}, lon^{\circlearrowleft}) \in [lat_{min}, lat_{max}] \times [lon_{min}, lon_{max}]$, i.e. the scenario and the produced routes themselves are located within a maximum square area.

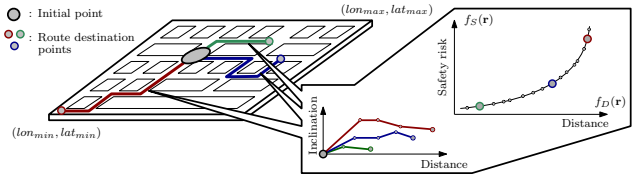


Fig. 1. Schematic diagram of the routing scenario tackled in this paper.

In this scenario a route $\mathbf{r}_i$ will be given by a variable-length sequence of *segments* $(\mathbf{s}_i^1, \ldots, \mathbf{s}_i^{N_i})$, where $\mathbf{s}_i^j$ denotes the j-th segment of route $\mathbf{r}_i$ and N_i the overall number of segments of the entire route. Each segment $\mathbf{s}_i^j$ is composed by a set of parameters,

$$\mathbf{s}_i^j \doteq \{(lat_i^{j,o}, lon_i^{j,o}), (lat_i^{j,d}, lon_i^{j,d}), d_i^j, t_i^j, \alpha_i^j, v_i^{j,max}\} \tag{1}$$

that characterize the segment in terms of its latitude/longitude extremes (^{o}: origin, ^{d}: destination), its length $d_i^j \in \mathbb{R}^+$, the time $t_i^j \in \mathbb{R}^+$ taken to traverse it by bike, its inclination profile $\alpha_i^j \in \mathbb{R}[0, 90]$, and maximum speed $v_i^{j,max} \in \mathbb{R}[0, V^{max}]$ of the road traffic along the segment, where V^{max} is the maximum admissible road vehicle speed as per the legislation of the scenario at hand. Given that we seek continuous routes, they should all fulfill $(lat_i^{1,o}, lon_i^{1,o}) = (lat^{\circlearrowleft}, lon^{\circlearrowleft})$ and

$$(lat_i^{j,d}, lon_i^{j,d}) = (lat_i^{j+1,o}, lon_i^{j+1,o}), \quad \forall j = 1, \ldots, N_i - 1. \tag{2}$$

Based on the above notation, the overall distance and time of the route will be given by $f_D(\mathbf{r}_i) \doteq \sum_{j=1}^{N_i} d_i^j$ and $T(\mathbf{r}_i) \doteq \sum_{j=1}^{N_i} t_i^j$, i.e. the sum of segments' distance/time.

Intuitively, the level of safety when a bike traverses route $\mathbf{r}_i$ should be driven by two different aspects: first, the inclination of its segments should be as close to 0 as possible (namely, a flat segment) so that the rider does not loose control of the bike due to either a high speed and risk to encounter unavoidable moving obstacles along the segment (downhill), or a physically demanding uphill segment that could put in danger the health of the biker and his/her capability to react against vehicles in the opposite direction. All in all, it should be clear that the inclination of the segment as per α_i^j plays a crucial role when quantifying the degree of safety of a segment. Based on this rationale and the notation introduced above, we propose a measure of route safety as

$$f_S(\mathbf{r}_i) \doteq \sum_{j=1}^{N_i} \left(d_i^j / \cos \alpha_i^j \right)^{v_i^{j,max}/V^{max}} \tag{3}$$

from where it is straightforward to note that the higher the value of $f_S(\mathbf{r}_i)$ is, the less safe route $\mathbf{r}_i$ will be. In other words, $f_S(\mathbf{r}_i)$ must be conceptually conceived as a measure of the risk assumed by the biker when traversing route $\mathbf{r}_i$, which should be minimized in the problem formulation. The baseline user parameters required for stating the

problem also include an upper bound T^{max} for the trip time $T(\mathbf{r}_i)$ needed to complete route $\mathbf{r}_i$. Routes $\mathbf{r}_{i'}$ for which $T(\mathbf{r}_i) > T^{max}$ should not be allowed to appear in the eventually output set of routes. Correspondingly, the minimum trip time is defined as a fraction $\rho \in \mathbb{R}[0,1]$ of T^{max}, such that a feasible route $\mathbf{r}_i$ should meet $T(\mathbf{r}_i) \geq \rho \cdot T^{max}$ in all cases. This parameter ρ is input by the user to reflect his/her tolerance respect to the admissible maximum time for the bike ride.

This notation being defined, the bike routing problem addressed in this paper can be formulated as the discovery of a set of routes that balances their safety and distance values in a Pareto optimal fashion satisfying, at the same time, lower and upper bounds in regards to their total duration. Mathematically:

$$\underset{\mathbf{R} \in \boldsymbol{\mathcal{R}}}{\text{minimize}} \quad f_S(\mathbf{r}), \quad \underset{\mathbf{R} \in \boldsymbol{\mathcal{R}}}{\text{maximize}} \quad f_D(\mathbf{r}), \tag{4a}$$

$$\text{subject to} \quad T(\mathbf{r}_i) \leq T^{max}, \tag{4b}$$

$$T(\mathbf{r}_i) \geq T^{min} = \rho \cdot T^{max}, \tag{4c}$$

$$(lat_i^{1,o}, lon_i^{1,o}) = (lat^{\circlearrowleft}, lon^{\circlearrowleft}) \; \forall \mathbf{r}_i \in \mathbf{R}, , \tag{4d}$$

$$(lat_i^{j,d}, lon_i^{j,d}) = (lat_i^{j+1,o}, lon_i^{j+1,o}), \; \forall j = 1, \ldots, N_i - 1, \tag{4e}$$

where $\mathbf{R}$ denotes a variable-length set of trip routes rooted on $(lat^{\circlearrowleft}, lon^{\circlearrowleft})$, and $\boldsymbol{\mathcal{R}}$ the number of all possible route sets satisfying this latter constraint.

3 Considered Solvers

We have chosen four population-based bio-inspired algorithms to solve the above bi-objective problem in a computationally efficient fashion. As a result, not only we obtain an insight of the solutions that can be obtained, but we can also determine which solver provides routes with best Pareto quality in terms of convergence and diversity.

However, before proceeding with the explanation of the utilized solvers, we delve into the strategy adopted to encode routes so that they can be handled by their heuristic operators. In this regard the numerical encoding does not represent a route by itself, but rather a set of factors that can be used to calculate a route by means of the OTP route generation engine. This way, each candidate route ($\mathbf{r}_p$) within the P-sized populations these algorithms is a vector comprising the following five values:

- *Latitude* $lat_p^{N_p}$ and *longitude* $lon_p^{N_p}$ of the destination location of route $\mathbf{r}_p$. Rather than using the true coordinates in the candidate, the relative difference between the origin location $(lat^{\circlearrowleft}, lon^{\circlearrowleft})$ and the coordinates $(lat_p^{N_p}, lon_p^{N_p})$ is instead used.
- *Safety preference* ($S_p \in \mathbb{R}[0,1]$), which stands for the priority that the OTP route planner should grant to the safety of the route. If this value is high, routes with a high safety will be better rated and output by the engine.
- *Inclination preference* ($I_p \in \mathbb{R}[0,1]$): the importance that the planner endows to the aggregate inclination of the route.
- *Duration preference* ($D_p \in \mathbb{R}[0,1]$): the importance that the route planner gives to the duration of the route.

These values are modified along the execution by means of the bio-inspired operators of every solver, repaired to ensure that $S_p + I_p + D_p = 1 \; \forall \mathbf{r}_p$ in the population,

and delivered to the OTP engine as new routing requests to produce routes based on the new set of preferences. Once created, every newly produced route is evaluated in terms of safety and distance, and then ranked and sorted as defined by the heuristic algorithm at hand.

In this regard, three of the selected algorithms are NSGA-II [11], MOEA/D [12], and SMS-EMOA [13], which are archetypal of Pareto dominance based, decomposition based, and indicator based evolutionary algorithms, respectively. We also added a particle swarm optimization technique, SMPSO [16], which has shown a remarkable performance in solving continuous multi-objectives problems as the one we are dealing with. We briefly describe these metaheuristics next:

- NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a generational genetic algorithm which has become the most well-known and widely used multi-objective algorithm since it was first proposed. It applies a Pareto ranking scheme to foster the convergence to the Pareto front and the crowding distance density estimator to promote the diversity of the front of solutions it is managing.
- MOEA/D (Multi-Objective Evolutionary Algorithm Based on Decomposition) is a steady-state evolutionary algorithm based on an aggregative approach with the aim of decomposing a multi-objective problem in a set of single-objective subproblems that are solved at the same time by taking into account information of a number of neighbors. We use in this paper the MOEA/DE version [17], which uses differential evolution instead of the mutation and crossover operators of the original proposal.
- SMS-EMOA (S-Metric Selection EMOA) is also an steady-state evolutionary algorithm which is based on NSGA-II but, instead of using the crowding distance density estimator, it applies the concept of hypervolume contribution. The idea is that, after applying the ranking procedure, the solution belonging to the last rank having the lowest contribution to the hypervolume of the set of solutions of that rank is removed.
- SMPSO (Speed-constrained Multi-objective PSO) is particle swarm optimization algorithm whose main features is the use of a velocity constraint mechanism, to avoid the particles to fly beyond the limits of the search space, and an external bounded-sized archive to store the non-dominated solutions found during the search. This archive is used also for leader selection and the crowding distance density estimator is used to remove solutions when it becomes full.

4 Description of the Simulation Environment

As has been mentioned in previous sections, the developed route planning system hinges on the route generation functionality provided by the OTP platform, an open source framework for mono and multi-modal journey planning. It follows a client-server model, and provides a map-based web and smartphone interface, as well as a REST API for its use with third-party applications. OTP operates with different open data standards, such as GeoTIFF, Protocol Buffers, General Transit Feed Specification (GTFS) and Open Street Map (OSM). Different reason have motivated the use of this platform:

- OTP is open source in its entirety, easing its adaptation to the specific simulation scenarios considered in this study.
- OTP efficiently works with OSM, providing the structure to automatically build the street network.

– OTP is well documented, updated, with an active, growing community of developers. These situation facilitates the understanding and maintenance of the platform.

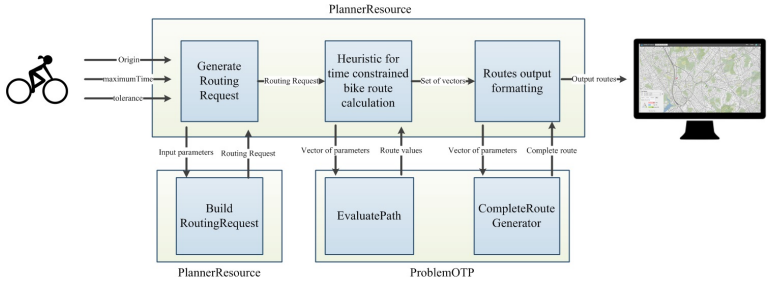


Fig. 2. Architecture of the deployed system.

The simulation platform developed in this study and architecturally illustrated in Fig. 2 relies on the Java project of OTP available in [15]. Specifically, this repository contains the complete code of the OTP system and the client for testing purposes. As can be read in the documentation of the project, *"it includes a REST API for journey planning as well as a map-based Javascript client. Open Trip Planner can also create travel time contour visualizations and compute accessibility indicators for planning and research applications"*. As OTP is open source, both parts have been adapted and modified to accommodate the requirements of this study and to enable constraining the routes in time, and the output of all segments comprising the generated route. Consequently, several classes have been created in order to deploy these functionalities.

In order to focus the scope of this manuscript strictly on the heuristic domain, implementation details on the modified OTP Java classes are not provided. Instead, we just mention and describe two newly developed methods, which are crucial for the understanding of the whole system: 1) `evaluatePath()`, recurrently called by the algorithm to return the length and safety of the route provided as an input; and 2) `completeRouteGenerator()`, which returns the entire path calculated by the OTP engine from its encoded representation for visualization and further analysis. Finally, it should be noted that the architecture also integrates the jMetal framework [18] jointly with OTP for implementing the considered multi-objective solvers.

5 Experimental Setup

In order to shed light on the empirical performance of the 4 multi-objective optimization algorithms considered to deal with the posed routing problem, several computer experiments have been carried out over different square areas located in the city of Bilbao (Spain), bounded by coordinates $(lat^{min}, lat^{max}) = (lat^{\circlearrowleft} - 0.1425, lat^{\circlearrowleft} + 0.1425)$ and $(lon^{min}, lon^{max}) = (lon^{\circlearrowleft} - 0.0665, lon^{\circlearrowleft} + 0.0665)$. Differences between scenarios yield from the selection of different initial points $(lon^{\circlearrowleft}, lat^{\circlearrowleft})$ for the routes, so that topological changes in the urban areas enclosed by such squares are expected to arise from the performed experiments. These selected scenarios are characterized by a flat, a highly slopped (hilly) and a hybrid terrain profile. This tailored choice permits to verify how the proposed system and the considered heuristic solvers behave in geographically

diverse setups. Two different open data sources have been used for the deployment of these real-world scenarios:

– *OSM map files*: the maps and street networks of the simulated scenarios have been retrieved in the form of OSM map files from the Planet OSM public repository [19], using the open tool BBBike [20] to download them in Protocolbuffer Binary Format (PBF). The downloaded OSM tiles contain all nodes, ways and relations required to build the map. OSM files are directly consumed by OTP, which automatically constructs the full road network.
– *Digital Elevation Model*: the elevations of the streets of Bilbao is downloaded from SRTM Tile Grabber[6] and directly consumed by OTP in GeoTIFF format. This format is a public domain metadata standard, which allows georeferencing information to be embedded within a TIFF file [21]. OTP uses these files for assigning the corresponding elevation to the entire street network, and it is employed for calculating the route flatness and safety.

Table 1. Parameter setting of the heuristics considered in the experimental benchmark.

Algorithm	Parameter	Value
All	*Population/swarm size*	100 individuals/particles
	Evaluations	5000
	Independent runs	15
	Mutation	Polynomial mutation
	| *Probability*	0.2 (once per every 5 decision variables)
	| *Distribution index* η_m	20.0
NSGA-II SMS-EMOA	*Crossover*	Simulated binary crossover
	| *Probability*	1.0
	| *Distribution index* η_m	20
MOEA/D	Differential evolution scheme	rand/1/bin
	| *CR*	1.0
	| *F*	0.5
	Neighborhood size	20
	Neighborhood selection probability	0.9
	Max. number of replaced solutions	2
SMPSO	*Archive size*	100
	Density estimator	Crowding distance

As the bike routing is a continuous optimization problem, we have configured the algorithms with commonly accepted settings, without any attempt at finding their best parameter configuration. A summary of the parameters is included in Table 1. All the algorithms have a population size of 100 individuals (or particles in the case of SMPSO) and use a polynomial mutation operator which is applied with a probability of 0.2 (according to the typical value of 1.0 / L, where $L = 5$ is the number of decision variables of the problem) and a distributed index equal to 20.0; the maximum number of function evaluations has been fixed to 5000. Both NSGA-II and SMS-EMOA apply a simulated binary crossover, with a probability of 1.0 and a distributed index equal to 20.0. MOEA/D follows a rand/1/bin differential evolution scheme, with parameters $CR = 1.0$ and $F = 0.5$. The values of the neighborhood size, the neighborhood selection probability, and maximum number of replace solutions are 20, 0.9, and 2, respectively. SMSPO has an external archive of a maximum size of 100 particles and applies the crowding distance density estimator.

[6] http://dwtkns.com/srtm/

For proving the robustness of the methods and extracting fair and rigorous conclusions, 15 independent runs have been made per algorithm for all problem scenarios (see next section). To assess the performance of the algorithms, we have used the so-called hypervolume [22], a Pareto compliant quality indicator that takes into account both the convergence and diversity of the Pareto front approximations returned by the solvers included in the benchmark.

Since we deal with an optimization problem whose true Pareto front is unknown, we have generated a reference Pareto front for each instance by combining all the non-dominated solutions computed in all the executions of all the algorithms. This front will be used as a reference to compute the hypervolume. Furthermore, in order to assess whether the differences between the algorithm results have statistical significance, we have applied the Wilcoxon rank-sum text, a non-parametric statistical hypothesis test which allows for a pairwise comparison between two samples. A significance level of 5% has been considered, meaning that the differences are unlikely to have occurred by chance with a probability of 95%.

5.1 Results and Discussion

To illustrate the fronts that each of the four compared algorithms have produced, we include in Fig. 3 (next page) the approximations corresponding to the best hypervolume values for the flat (first row), hybrid (second row) and hilly scenario (third row). To ease the visualization of the solutions, the reference Pareto front is included as a continuous line. We can observe how SMPSO excels at generating a set of solutions that are on top of the reference Pareto front and that are uniformly spread, including the extreme solutions. By contrast, MOEA/D fails to generate a front with accurate convergence and widespread diversity.

The results of the hypervolume values obtained by the four metaheuristics are presented in Table 2, which includes the median and interquartile range of the 15 independent runs per algorithm and problem instance. Those cells with a dark and light gray backgrounds indicates, respectively, the best and second best indicator values. We can observe that the particle swarm optimization algorithm SMPSO has produced the best (highest) values in the three considered scenarios. NSGA-II and SMS-EMOA have yielded, respectively, two and one second best values.

Table 2. Median and Inter Quartile Range (IQR) of the hypervolume values obtained by the algorithms. Best and second best median results have dark and light gray backgrounds, respectively.

	SMPSO	NSGA-II	MOEA/D	SMS-EMOA
Flat scenario	$5.80e-01_{8.8e-03}$	$5.68e-01_{6.2e-03}$	$5.27e-01_{7.5e-03}$	$5.64e-01_{1.2e-02}$
Hybrid scenario	$5.95e-01_{3.8e-03}$	$5.79e-01_{1.0e-02}$	$5.30e-01_{1.8e-02}$	$5.74e-01_{1.2e-02}$
Hilly scenario	$7.00e-01_{8.4e-03}$	$6.13e-01_{5.0e-02}$	$5.71e-01_{2.2e-02}$	$6.20e-01_{8.2e-02}$

To determine whether the differences between pair of algorithms are statistically significant, we have applied the Wilcoxon rank-sum test; the obtained results are included in Table 3. In each cell, the three considered scenarios (plain, medium, and high) are represented with one of the following symbols: "–" indicates that there not statistical significance between the algorithms, "▲" means that the algorithm in the row

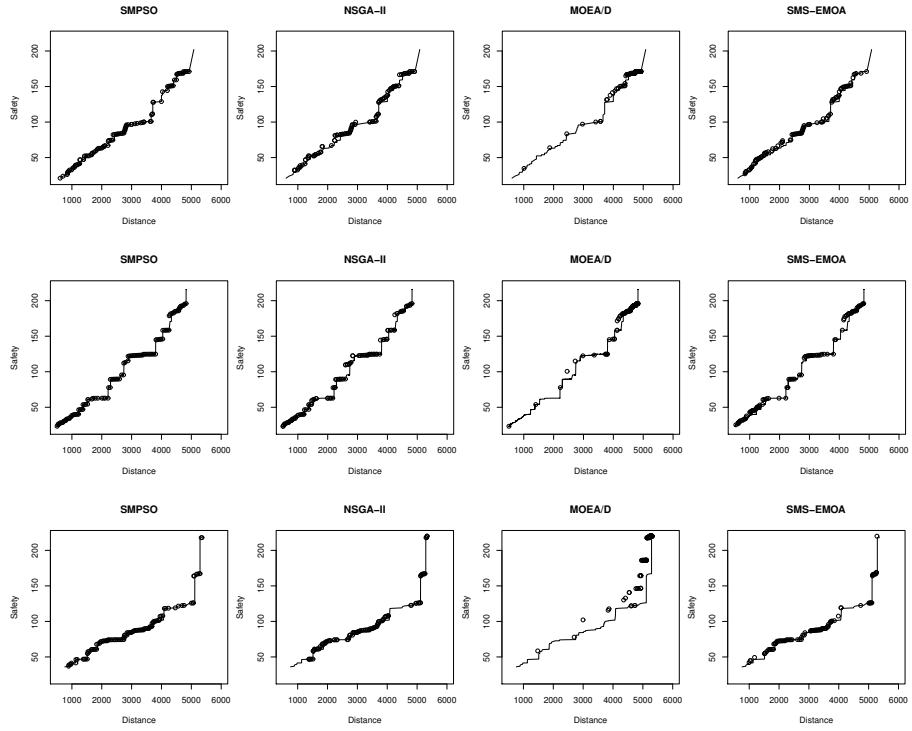


Fig. 3. Estimated Pareto fronts with the best hypervolume values obtained by the four compared algorithms for the flat (first row), hybrid (second row) and hilly (third row) scenarios. The line stands for the reference Pareto front approximation.

has yielded better results than the algorithm in the column with confidence, and "▽" is used when the algorithm in the column is statistically better than the algorithm in the row. We can observe that all the differences are significant but the results of NSGA-II and SMS-EMOA, so we can claim that SMPSO is the algorithm providing the best overall performance in the context of the study carried out.

Table 3. Wilcoxon test results. Each cell contains a symbol per problem (flat, hybrid, hilly).

	NSGA-II	MOEA/D	SMS-EMOA
SMPSO	▲ ▲ ▲	▲ ▲ ▲	▲ ▲ ▲
NSGA-II		▲ ▲ ▲	– – –
MOEA/D			▽ ▽ ▽

The results provided by the hypervolume indicator are quantitative, in the sense that they indicate which algorithm generates the fronts with better degrees of convergence and diversity. However, they do not provide any insight about the quality of the solutions. For that reason, we next include in Fig. 4 the reference Pareto fronts for each of the three scenarios, and a visual representation of the routes corresponding to the points in the estimated Pareto front of the hilly scenario. Indeed differences are visually clear

in regards to the distance of every route: interesting is to note that the one in green traverses a urban area with sharp slopes (namely, the urban core of the town of Portugalete within the metropolitan area of *Gran Bilbao*).

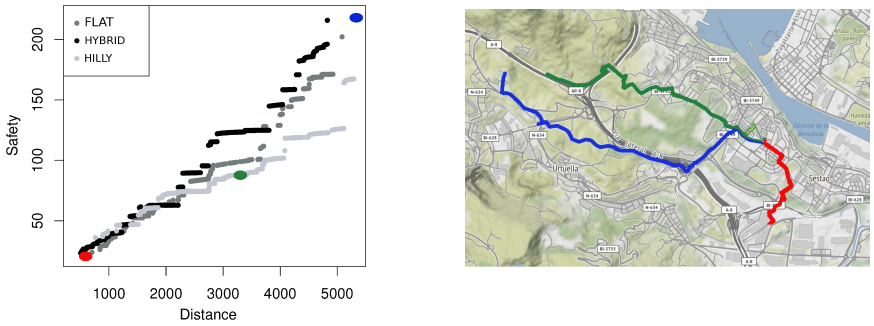


Fig. 4. (Left) Reference Pareto fronts obtained for the three considered scenarios; (Right) routes corresponding to the colored points in the reference Pareto front of the hilly scenario.

6 Conclusions and Further Work

In this work the design of time-constrained bike routes has been studied and approached from a multi-criteria perspective. The focus has been placed on generating a group of open-destination bike routes based on three input parameters: origin, maximum trip time and tolerance. The problem has been modeled as a bi-objective paradigm balancing two conflicting objectives: the distance of the route and its safety level, the latter blending together the inclination of segments composing the route, their length and the speed of vehicles along each segment. For efficiently tackling this problem, four bio-inspired multi-objective optimization methods have been used (namely, NSGA-II, SMS-EMOA, MOEA/D, and SMPSO), and applied to three different real-world scenarios placed in Bilbao, Spain. Experiments have been conducted in a realistic simulation environment based on Open Trip Planner as the software simulation platform. The obtained results reveal that the SMPSO solver outperforms its counterparts in the benchmark in terms of Pareto optimality as gauged by their hypervolume indicator.

Several research lines will be tackled in the near future. In short term, additional bio-inspired approaches are planned to be included in the benchmark to assess whether they get to obtain better results in terms of Pareto spread and dominance. In the longer term we intend to extend the problem formulation to add new objectives such as the energy consumed by the user and air pollution in the route. Real open data will be utilized to model this improved setup.

Acknowledgments

E. Osaba and J. Del Ser would like to thank the Basque Government for its funding support through the EMAITEK program. This work is also partially funded by Grants TIN2017-86049-R and TIN2014-58304 (Ministerio de Ciencia e Innovación), and P11-TIC-7529 and P12-TIC-1519 (Plan Andaluz I+D+I).

References

1. Wang, F.Y.: Scanning the issue and beyond: Transportation and mobility transformation for smart cities. IEEE Transactions on Intelligent Transportation Systems **16**(2) (2015) 525–533
2. Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., Werneck, R.F.: Route planning in transportation networks. In: Algorithm Engineering. Springer (2016) 19–80
3. Zografos, K.G., Androutsopoulos, K.N.: Algorithms for itinerary planning in multimodal transportation networks. IEEE Trans. on Intelligent Transportation Systems **9**(1) (2008) 175–184
4. Staunton, C.E., Hubsmith, D., Kallins, W.: Promoting safe walking and biking to school: the marin county success story. American Journal of Public Health **93**(9) (2003) 1431–1434
5. Wang, S., Lin, W., Yang, Y., Xiao, X., Zhou, S.: Efficient route planning on public transportation networks: A labelling approach. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM (2015) 967–982
6. Turverey, R.J., Cheng, D.D., Blair, O.N., Roth, J.T., Lamp, G.M., Cogill, R.: Charlottesville bike route planner. In: Systems and Information Engineering Design Symposium, IEEE (2010) 68–72
7. Bucher, D., Jonietz, D., Raubal, M.: A heuristic for multi-modal route planning. In: Progress in Location-Based Services 2016. Springer (2017) 211–229
8. Hrnčíř, J., Žileckỳ, P., Song, Q., Jakob, M.: Practical multicriteria urban bicycle routing. IEEE Transactions on Intelligent Transportation Systems **18**(3) (2017) 493–504
9. Martins, E.Q.V.: On a multicriteria shortest path problem. European Journal of Operational Research **16**(2) (1984) 236–245
10. Caggiani, L., Camporeale, R., Ottomanelli, M.: A real time multi-objective cyclists route choice model for a bike-sharing mobile application. In: IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, IEEE (2017) 645–650
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation **6**(2) (2002) 182–197
12. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **8**(11) (2008) 712–731
13. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research **181**(3) (September 2007) 1653–1669
14. Nebro, A.J., Durillo, J.J., Garcia-Nieto, J., Coello, C.C., Luna, F., Alba, E.: Smpso: A new pso-based metaheuristic for multi-objective optimization. In: IEEE Symposium on Computational intelligence in multi-criteria decision-making, IEEE (2009) 66–73
15. Open Trip Planner. https://github.com/opentripplanner . Accessed: 2017-11-30.
16. Nebro, A., Durillo, J., García-Nieto, J., Coello Coello, C., Luna, F., Alba, E.: Smpso: A new pso-based metaheuristic for multi-objective optimization. In: 2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009), IEEE Press (2009) 66–73
17. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. IEEE Trans. on Evolutionary Computation **12**(2) (April 2009) 284–302
18. Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software **42**(10) (2011) 760–771
19. Planet OSM. https://planet.osm.org/ . Accessed: 2017-11-30.
20. BBike tool. https://extract.bbbike.org/ . Accessed: 2017-11-30.
21. Hart, C., Koupal, J., Giannelli, R.: Epa's onboard analysis shootout: Overview and results. Technical report, United States Environmental Protection Agency (2002)
22. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. on Evol. Comp. **3**(4) (1999) 257–271

New Techniques for Inferring L-systems Using Genetic Algorithm*

Jason Bernard and Ian McQuillan

Department of Computer Science,
University of Saskatchewan, Saskatoon, Canada
jason.bernard@usask.ca, mcquillan@cs.usask.ca

Abstract. Lindenmayer systems (L-systems) are a formal grammar system that iteratively rewrites all symbols of a string, in parallel. When visualized with a graphical interpretation, the images have been particularly successful as a concise method for simulating plants. Creating L-systems to simulate a given plant manually by experts is limited by the availability of experts and time. This paper introduces the Plant Model Inference Tool (PMIT) that infers deterministic context-free L-systems from an initial sequence of strings generated by the system using a genetic algorithm. PMIT is able to infer more complex systems than existing approaches. Indeed, while existing approaches can infer D0L-Systems where the sum of production successors is 20, PMIT can infer those where the sum is 140. This was validated using a testbed of 28 known D0L-system models, in addition to models created artificially by bootstrapping larger models.

Keywords: L-systems, inductive inference, genetic algorithm, plant modeling

1 Introduction

Lindenmayer systems (L-systems), introduced in [7], are a formal grammar system that produces self-similar patterns that appear frequently in nature, and especially in plants [11]. L-systems produce strings that get rewritten over time in parallel. Certain symbols can be interpreted as instructions to create sequential images, which can be visually simulated by software such as the "virtual laboratory" (vlab) [14]. Such simulations are useful as they can incorporate different geometries [11], environmental factors [1], and mechanistic controls [10], and are therefore of use to simulate and understand plants. L-systems often consist of small textual descriptions that require little storage compared to real imagery. Certainly also, they can produce a simulation extremely quickly with low cost computers in comparison to actually growing a plant.

An L-system is denoted by a tuple $G = (V, \omega, P)$, which consists of an alphabet V (a finite set of allowed symbols), an axiom ω that is a word over V,

* This research was supported in part by a grant from the Plant Phenotyping and Imaging Research Centre.

and a finite set of productions, or rewriting rules, P. A deterministic context-free L-system or a D0L-system, has exactly one rule for each symbol in V of the form $A \to x$, where $A \in V$ (the predecessor) and x is a word over V (the successor, denoted by $succ(A)$). Words get rewritten according to a derivation relation, $\Rightarrow$, whereby $A_1 \cdots A_n \Rightarrow x_1 \cdots x_n$, where $A_i \in V, x_i$ is a word, and $A_i \to x_i$ is in P, for each i, $1 \leq i \leq n$. Normally, one is concerned with derivations starting at the axiom, $\omega \Rightarrow \omega_1 \Rightarrow \omega_2 \Rightarrow \cdots \Rightarrow \omega_n$. The sequence $(\omega_1, \ldots, \omega_n)$ is known as the developmental sequence of length n.

One common alphabet for visualization is the turtle graphics alphabet [11], so-called as it is imagined that each word generated contains a sequence of instructions that causes a turtle to draw an image with a pen attached. The turtle has a state consisting of a position on a (usually) 3D grid and an angle, and the common symbols that cause the turtle to change states and draw are: F (move forward with pen down), f (move forward with pen up), $+$ (turn left), $-$ (turn right), [(start a branch),] (end a branch), & (pitch up), $\wedge$ (pitch down), \(roll left), / (roll right), | (turn around 180°). For branching models, [causes the state to be pushed on a stack and] causes the state to be popped and the turtle switches to it. It is assumed that the right hand side of rewriting rules have parantheses that are properly nested. Additional symbols are added to the alphabet, such as A and B, to represent the underlying growth mechanics. The "Fractal Plant" L-system is inferred commonly [9, 13] and so is shown here as an example: $G = (\{X, F\}, X, \{X \to F[+X]F[-X]+X, F \to FF\})$. After 7 generations, "Fractal Plant" can produce the image in Fig. 1 after 7 generations. More realistic 3D models may be produced with extensions of D0L-systems.

Fig. 1. Fractal Plant after 7 generations [11].

A difficult challenge is to determine an L-system that can accurately simulate a plant. In practice, this often involves manual measurements over time, scientific knowledge, and is done by experts [12]. Although this approach has been successful, it does have notable drawbacks. Producing a system manually requires an expert that are in limited supply, and it does not scale to producing arbitrarily many models. Furthermore, the more complex plant models require a priori knowledge of the underlying mechanics of the plant, which are difficult and time consuming to acquire. To address this, semi-automated (used as an aide for the expert) [6, 8], and fully automated approaches [9, 13], have been introduced to find an L-system that matches observed data. This approach has the potential to scale to constructing thousands of models, and also has the potential to expose biomechanics rather than requiring its knowledge beforehand.

The ultimate goal of this research is to automatically determine a model from a sequence of plant images over time. An intermediate step is to infer the model from a sequence of strings used to draw the images. This is known as the inductive inference problem, defined as follows. Given a sequence of strings

$\alpha = (\omega_1, \ldots, \omega_n)$, find a D0L-system, or if it exists, $G = (V, \omega, P)$ such that $\omega = \omega_0 \Rightarrow \omega_1 \Rightarrow \cdots \Rightarrow \omega_n$ where α is the developmental sequence of length n.

This paper introduces the Plant Model Inference Tool (PMIT) that aims to be a fully automated approach to inductive inference of L-systems. Towards that goal, PMIT uses a genetic algorithm (GA) to search for an L-system to match the words produced. This paper presents a different encoding scheme than previous approaches, and shows that it is more effective for inferring D0L-systems. Additionally, some logical rules based on necessary conditions are used as heuristics to shrink the solution space. Between these two techniques, it is determined that PMIT is able to infer L-systems where the sum of the production successors is approximately 140 symbols in length; whereas, other approaches are limited to about 20 symbols. Moreover, the testbed used to test PMIT is significantly larger than previous approaches. Indeed, 28 previously developed D0L-systems are used, and for these systems that PMIT properly inferred, it did so in an average of 17.762 seconds. Furthermore, additional (in some sense "artificial") models are created by combining the existing models where the combined length of the successors is longer than 140 symbols (which PMIT does not solve), and then randomly removing "F" symbols until it can solve them. This work can be seen as a step towards the goal of 3D scanning a plant over time, converting the images into strings that describe how to draw them, then inferring the L-system from the sequence of strings.

The remainder of this paper is structured as follows. Section 2 describes some existing automated approaches for inferring L-systems. Section 3 describes the logical rules used to shrink the solution space, and Section 4 discusses the genetic algorithm. Section 5 will discuss the methodology used to evaluate PMIT and the results. Finally, Section 6 concludes the work and discusses future directions. Some details are omitted due to space constraints, but appear online [4].

2 Background

This section briefly describes some notation used throughout the paper, contains a brief description of genetic algorithms since they are used as the search mechanism here, then describes some existing approaches to L-system inference.

An alphabet is a finite set of symbols. Given an alphabet V, a word over V is any sequence of letters written $a_1 a_2 \cdots a_n$, $a_i \in V, 1 \leq i \leq n$. The set of all words over V is denoted by V^*. Given a word $x \in V^*$, $|x|$ is the length of x, and $|x|_A$ is the number of A's in x, where $A \in V$. Given two words $x, y \in V^*$, then x is a substring of y if $y = uxv$, for some $u, v \in V^*$ and in this case y is said to be a superstring of x. Also, x is a prefix of y if $y = xv$ for some $v \in V^*$, and x is a suffix of y if $y = ux$ for some $u \in V^*$.

The GA is an optimization algorithm, based on evolutionary principles, used to efficiently search N-dimensional (usually) bounded spaces [2]. In evolutionary biology, increasingly fit offspring are created over successive generations by intermixing the genes of parents. An encoding scheme is applied to convert a problem into a virtual genome consisting of N genes. Each gene is either a bi-

nary, integer, or real value and represents, in a problem specific way, an element of the solution to the problem. One common type of encoding is a real mapped encoding, where the genes have a real value from 0 and 1 and different ranges within are mapped contextually [2]. This encoding works best when the options at each step of the problem are unknown or dependent on prior choices.

The GA functions by first creating an initial population (P) of random solutions. Each member of the population is assessed using a problem specific fitness function. Then the GA, controlled by certain parameters, performs a selection, crossover, mutation, and survival step until a termination condition is reached. In the selection step, a set of pairs of genomes are selected from the population with odds in proportion to their fitness, i.e. preferring more fit genomes. During the crossover step, for each selected pair, a random selection of genes are copied between the two; thereby, producing two offspring. Each gene has a chance of being swapped equal to the control parameter *crossover weight*. The mutation step takes each offspring and randomly changes zero or more genes to a random value with each gene having a chance of being mutated equal to the *mutation weight*. Then each offspring is evaluated using the fitness function. The offspring are placed into the population and genomes are culled until the population is of size P again. Usually, the most fit members are kept (elite survival). The termination condition may be based on such criteria as finding a solution with sufficient fitness, or hitting a pre-determined maximum number of generations.

Various approaches to L-system inference were surveyed in [3]. There are several different broad approaches towards the problem: building by hand [11,12], algebraic approaches [5, 9], using logical rules [9], and search approaches [13]. Since PMIT is a hybrid approach incorporating a search algorithm, GA, together with logical rules to reduce intractability by shrinking the search space, this section will examine some existing logic-based and search-based approaches.

Inductive inference has been studied theoretically (without implementation) by several authors [3], e.g. Doucet [5]. He devised a method that uses solutions to Diophantine equations to, in many cases, find a D0L-system that starts by generating the input strings. A similar approach was implemented with a tool called LGIN [9] that infers L-systems from a single observed string ω. They devise a set of equations that relate the number of each symbol observed in ω to the linear combination of the production values in the growth matrix. LGIN is limited to two symbol alphabets, which is still described as "immensely complicated" [9], and was evaluated on six variants of "Fractal Plant" [11] and had a peak execution time of four seconds.

Runqiang et al. [13] propose to infer an L-system from an image using a GA. Each gene is encoded to represent a symbol in each successor. The fitness function matches the candidate system to the observed data using image processing. Their approach is limited to an alphabet size of 2 and a maximum total length of all successors of 14. Their approach is 100% successful for a variant of "Fractal Plant" [11] with $|V| = 1$, and has a 66% success rate for a variant of "Fractal Plant" [11] with $|V| = 2$. Although they do not list timings, their GA converged after a maximum of 97 generations, which suggests a short runtime.

3 PMIT Methodology for Logically Deducing Facts about Successors

In this section, the methodology that is used by PMIT to reduce the size of the solution space with heuristics — all of which are based on necessary conditions for D0L-systems — will be described. Indeed, the success and efficiency of a search algorithm is generally tied to the size of the solution space. As all these conditions are mathematically true, this guarantees that a correct solution is in the remaining search space (if there is a D0L-system that can generate the input). In PMIT, logical rules are used to reduce the dimensional bounds in two contexts. The first context is to determine a lower bound ℓ and upper bound u on the number of each symbol B produced by each symbol A for each $A, B \in V$, henceforth called growth of B by A. Thus, two programming variables $(A, B)_{min}$ and $(A, B)_{max}$ are created that change such that $(A, B)_{min} \leq |succ(A)|_B \leq (A, B)_{max}$. A second context is a separate lower ℓ and upper bound u on the length of each successor for each $A \in V$. Then, two programming variables A_{min} and A_{max} are used such that $A_{min} \leq |succ(A)| \leq A_{max}$ and their values improve as the program runs. The bounds on growth and on lengths depend on each other, so all the rules are run in a loop until the bounds stop improving.

For this paper, it is assumed that if a turtle graphic symbol has an identity production (e.g. $+ \rightarrow +$), then this is known in advance. Typically, these symbols do have identity productions. There are some instances where "F" may not, (some variants of "Fractal Plant" [11]). In such a case, "F" is treated as a non-turtle graphics symbol for the purposes of inferring the L-system. Also, all successors are assumed to be non-empty, which are commonly used in practice when developing models [11]. This implies that A_{min} is initialized to 1 for each $A \in V$. For each turtle symbol $T \in V$, $T_{min} = T_{max} = 1$,$(T, T)_{min} = (T, T)_{max} = 1$ and $(T, A)_{min} = (T, A)_{max} = 0$ for every $A \in V, A \neq T$.

3.1 Deducing Growth

Consider input $\alpha = (\omega_0, \ldots, \omega_n), \omega_i \in V^*, 0 \leq i \leq n$ with alphabet V. Deduction of growth in PMIT is based on two mechanisms; the first being the determination of so-called *successor fragments*, of which there are four types.

- A word ω is an A-subword fragment if ω must be a subword of $succ(A)$.
- A word ω is an A-prefix fragment if ω must be a prefix of $succ(A)$.
- A word ω is an A-suffix fragment if ω must be a suffix of $succ(A)$.
- A word ω is an A-superstring fragment if ω must be a superstring of $succ(A)$.

As PMIT runs, it can determine additional successor fragments, which can help to deduce growth. Certain prefix and suffix fragments can be found for the first and last symbols in each input word by the following process. Consider two words such that $\omega_1 \Rightarrow \omega_2$. It is possible to scan ω_1 from left to right until the first non-turtle graphics symbol is scanned (say, A, where the word scanned is αA). Then, in ω_2, PMIT skips over the graphical symbols in α (since each symbol in α has

a known identity production), and the next A_{min} symbols, β, (the current value of the lower bound for $|succ(A)|$) must be an A-prefix fragment. Furthermore, since branching symbols must be paired and balanced within a successor, if a [symbol is met, the prefix fragment must also contain all symbols until a matching] symbol is met. Similarly, an A-superstring fragment can be found by skipping α symbols, then taking the next A_{max} symbols from ω_2 (the upper bound on $|succ(A)|$). If a superstring fragment contains a [symbol without the matching] symbol, then it is reduced to the symbol before the unmatched [symbol. Then, lower and upper bounds on the growth of B by A ($(A,B)_{min}$ and $(A,B)_{max}$) for each $B \in V$ can be found by counting the number of B symbols in any prefix and superstring fragments respectively and changing them if the bounds are improved. For a suffix fragment, the process is identical except from right to left starting at the end of ω_1. An example of this process appears in [4].

The second mechanism for deduction of growth is based on calculating the number of times each symbol $A \in V$ appears in word ω_i above the number implied from ω_{i-1} together with the current values of each lower bound $(B,A)_{min}$, for each $B \in V$. Formally, a programming variable for the *accounted for growth* of a symbol $A \in V$ for $1 \leq i \leq n$, denoted as $G_{acc}(i,A)$ is:

$$G_{acc}(i,A) := \sum_{B \in V} (|\omega_{i-1}|_B \cdot (B,A)_{min}). \tag{1}$$

The *unaccounted for growth* for a symbol A, denoted as $G_{ua}(i,A)$, is computed as $G_{ua}(i,A) := |\omega_i|_A - G_{acc}(i,A)$.

Then, $(B,A)_{max}$ is set (if it can be reduced) under the assumption that all unaccounted for A symbols are produced by B symbols. Furthermore, $(B,A)_{max}$ is set to be the lowest such value computed for any word from 1 to n, where B occurs, as any of the n words can be used to improve the maximum. And, $|succ(B)|_A$ must be less than or equal to $(B,A)_{min}$ plus the additional unaccounted for growth of A divided by the number of B symbols (if there is at least one; also the floor function is used since $|succ(B)|_A$ is a positive integer) in the previous word, as computed by

$$(B,A)_{max} := \min_{\substack{1 \leq i \leq n, \\ |\omega_{i-1}|_B > 0}} \left((B,A)_{min} + \left\lfloor \frac{G_{ua}(i,A)}{|\omega_{i-1}|_B} \right\rfloor \right). \tag{2}$$

An example is presented in [4].

Once $(B,A)_{max}$ has been determined for every $A, B \in V$, the observed words are re-processed to compute possibly improved values for $(B,A)_{min}$. Indeed for each (B,A), if $x := \sum_{\substack{C \in V \\ C \neq B}} (C,A)_{max}$, and $x < |\omega_i|_A$, then this means that $|succ(B)|_A$ must be at least $\left\lceil \frac{|\omega_i|_A - x}{|\omega_{i-1}|_B} \right\rceil$, and then $(B,A)_{min}$ can be set to this value if its bound is improved. For example, if ω_{i-1} has 2 A's and 1 B, and ω_i has 10 A's, and $(A,A)_{max} = 4$, then at most two A's produce eight A's, thus one B produces at least two A's (10 total minus 8 produced at most by A), and $(B,A)_{min}$ can be set to 2.

3.2 Deducing Successor Length

The deduction of A_{min} and A_{max} are found from two logical rules, one involving the sum of the minimum and maximum growth over all variables, and one by exploiting a technical mathematical property. The first rule simply states that A_{min} is at least the sum of $(A, B)_{min}$ for every $B \in V$ and similarly A_{max} is at most the sum of $(A, B)_{max}$ for every $B \in V$. The second rule is trickier but often improves the bounds for A_{max} and A_{min} for $A \in V$. This takes place in steps. First, the maximum number of symbols that can be produced by A in ω_i is computed by: $x := |\omega_i| - \sum_{B \in V, B \neq A}(B_{min} \cdot |\omega_{i-1}|_B)$. If $|\omega_{i-1}|_A > 0$, let:

$$A^i_{max} := \left\lfloor \frac{x}{|\omega_{i-1}|_A} \right\rfloor \tag{3}$$

if its value is improved. It follows that A_{max} can be set to $\min_{\substack{1 \leq i \leq n, \\ |\omega_{i-1}|_A > 0}} A^i_{max}$, if its value is improved. Next, now that these A^i_{max} values have been calculated, it is sometimes possible to further improve the A_{max} and A_{min} values. Let $Y^i \in V$, $1 \leq i \leq n$ be such that Y^i occurs the least frequently in ω_{i-1} with at least one copy. The current value of Y^i_{max} will be examined as computed by Equation 3; note $Y^1, \ldots, Y^n$ can be different. Let $V^i_{max} := Y^i_{max} + \sum_{\substack{B \in V, \\ B \neq Y^i}} B_{min}$. Then, V^i_{max} can allow refinement of the upper bound for each successor, as A_{max} may be improved by assuming all other symbols produce their minimum and subtracting from V^i_{max}. Mathematically this is expressed as:

$$A_{max} := V^i_{max} - \sum_{\substack{B \in V, \\ B \neq A}} B_{min} \tag{4}$$

for $1 \leq i \leq n$, if A occurs in ω_{i-1}, and if the new value is smaller, which has the effect of the minimum over all i, $1 \leq i \leq n$. Although it is not immediately obvious that this formula is an upper bound on $|succ(A)|$, a mathematical proof has been completed (omitted due to space constraints), and appears in [4] along with an example of its use. Thus, A_{max} can be set in this fashion. Similarly, A_{min} can be set by taking Y^i that occurs most frequently.

4 Encoding for the L-system Inference Problem

In this section, the GA and encoding used by PMIT is described and contrasted with previous approaches.

The efficient search of a GA is controlled, in part, by the settings of the control parameters: population size, crossover weight, and mutation weight. The process of finding the optimal control parameter settings is called *hyperparameter search*. It was found via Random Search (details on methodology used in [4]) that the optimal parameter settings were 100 for population size, 0.85 crossover weight, and 0.10 for mutation weight. These parameters are henceforth used.

The fitness function for PMIT compares the symbols in the observed data to the symbols in the words produced by the candidate solution position by

position. An error is counted if the symbols do not match or if the candidate solution is too long or short. The base fitness is the number of errors divided by total number of symbols. If the candidate solution produces more than double the number of symbols expected, it is not evaluated and assigned an extremely high fitness so that it will not pass the survival step. Since errors early on in the input words $\omega_0, \ldots, \omega_n$ will cause errors later, a word ω_i, is only assessed if there are no errors for each preceding word, and 1.0 is added to F for each unevaluated word. This encourages the GA to find solutions that incrementally match more of the observed words. PMIT is also evaluated using brute force, which is guaranteed to find the most fit solution and it was found that the solution found by the GA matches that found by brute force, showing that this fitness function is effective at finding an optimal solution.

PMIT uses three termination conditions to determine when to stop running. First, PMIT stops if a solution is found with a fitness of 0.0 as such a solution perfectly describes the observed data. Second, PMIT stops after 4 hours of execution if no solution has been found. Third, PMIT stops when the population has converged and can no longer find better solutions. This is done by recording the current generation whenever a new best solution is found as Gen_{best}. If after an additional Gen_{best} generations, no better solutions are found, then PMIT terminates. To prevent PMIT from terminating early due to random chance, PMIT must perform at least 1,000 generations for the third condition only. This third condition is added to prevent the GA from becoming a random search post-convergence and finding an L-system by chance skewing the results.

The encoding scheme used most commonly in literature (e.g., [8, 13]) is to have a gene represent each possible symbol in a successor. The number of genes for the approaches in literature varies due the specific method they use to decode the genome into an L-system, although they are approximately the total length of all successors combined. With this approach, each gene represents a variable from V (encoded as an integer from 1 to $|V|$). However, in some approaches (and PMIT) the decoding step needs to account for the possibility that a particular symbol in a successor *does not exist* (represented by $\oslash$). When the possibility of an $\oslash$ exists, such genes have a range from 1 to $|V| + 1$. As an example, assume $V = \{A, B\}$ and $A_{min} = 2, A_{max} = 3, B_{min} = 1, B_{max} = 3$. For A, it is certain to have at least two symbols in the successor and the third may or may not exist. So, the first three genes represent the symbols in $succ(A)$, where the first two genes have each possible values from $\{A, B\}$ and the third gene has $\{A, B, \oslash\}$.

Next the improvements made to the genomic structure defined by the basic encoding scheme will be described. Although they are discussed separately for ease of comprehension, all the improvements are used together.

PMIT uses the bounds and successor fragments to create a genomic structure. For example, if $V = \{A, B\}$, $A_{min} = 1$, and $A_{max} = 3$, then $succ(A)$ can be expressed as the genomic structure of $\{A, B\}, \{A, B, \oslash\}, \{A, B, \oslash\}$. If there is an A-prefix of B, then the genomic structure can change to $\{B\}, \{A, B, \oslash\}, \{A, B, \oslash\}$ since the first symbol in $succ(A)$ is B, essentially eliminating the need for the first gene. This is similar for an A-suffix. The second improvement to the basic

encoding scheme further reduces the solution space by eliminating impossible solutions. When building the successor, PMIT first places the symbols known to be in $succ(A)$. For each successor, $\sum_{A,B \in V}(A,B)_{min}$ genes are created with a real value range between 0 and 1. Since these symbols must exist, the mapping selects an unused position within the successor. After these symbols are placed, if any additional symbols are needed up to the value of A_{max}, then PMIT selects the remainder allowing the option of using $\oslash$, ensuring that $(A,B)_{max}$ is not violated for any $A, B \in V$; i.e. the bounds computed by the heuristics shown in Section 3 ensure that the candidate solutions are always valid, and the genes' values are dynamically interpreted to ensure that the bounds are not violated. Further details and examples are in [4].

Lastly, it was determined that since new non-graphical symbols can only be produced by non-graphical symbols, it is possible to, at first, ignore the graphical symbols over a smaller alphabet V_{im}. Then, one can search for the successors over V_{im}^*, which is a simpler problem. For example, if $A \rightarrow F[+F]B$ and $B \rightarrow F[\text{-}F]A$, then with $V_{im} = \{A, B\}$ it is only necessary to find $A \rightarrow B$ and $B \rightarrow A$. Each graphical symbol can be added in one at a time. In the example above, the second step might add + to V_{im} and find $A \rightarrow +B$ and $B \rightarrow A$. Solving these smaller problems is more efficient as the individual search spaces are smaller and when summed are smaller than the solution space when trying to find the full successor in one step. Additional details, including the use of successor fragments to further simplify the number of genes needed, are omitted and appear in [4].

5 Data, Evaluation, and Results

To evaluate PMIT's ability to infer D0L-systems, ten fractals, six plant-like fractal variants inferred by LGIN [9,11], and twelve other biological models were selected from the vlab online repository [14]. The biological models consist of ten algaes, apple twig with blossoms, and a "Fibonacci Bush". The dataset compares favourably to similar studies where only some variants of one or two models are considered [9, 13]. The data set is also of greater complexity by considering models with alphabets from between 2 to 31 symbols compared to two symbol alphabets [9,13]. However, there remain gaps both in terms of successor lengths and alphabet size. Hence, additional L-systems are created by bootstrapping; that is, by combining successors from multiple L-systems to create new "fake" systems with every combination of alphabet size from 3 to 25 in increments of 2 and longest successor length from 5 to 25 in increments of 5. To get successors of the proper length some "F" symbols were trimmed from longer successors. These are called *generated L-systems*.

Two metrics are used to measure success. *Success rate* (SR) is the percentage of times PMIT can find any L-system that describes the observed data. *Mean time to solve* (MTTS) is the time taken to solve the models (measured using a single core of an Intel 4770 @ 3.4 GHz with 12 GB of RAM on Windows 10). PMIT stops execution at 4 hours (14400 seconds) calling the search a failure, as more than this time is not practical relative to other tools in the literature.

5.1 Results

Three programs were evaluated. The first is PMIT (implemented in C++ using Windows 10), the second is a restriction of PMIT that uses a brute force algorithm without the GA or logical rules, and the existing program LGIN. No comparison is made to the work by Runqiang et al. [13] as LGIN is strictly better; indeed, LGIN is the best approach that could be found in literature making it the best algorithm to which PMIT can be compared. The comparison between brute force and GA shows the effects of using GA on MTTS. Results are shown in Table 1. No SR is shown for LGIN as it is not explicitly stated; however, it is implied to be 100% [9] for all rows where a time is written. The variants used by LGIN [9, 11] are the six Fractal Plants. In general, PMIT is fairly successful at solving the fractals, the "Fractal Plant" variants, and also *Ditria reptans*. The success rates are all either 0% or 100%, indicating that a problem is either solved or not. It was observed that PMIT was able to solve many other models excluding the F and f symbols, as indicated in the "Infer Growth" column. For example, PMIT inferred for *Aphanocladia* that $A \to BA$, $B \to U[-C]UU[+/C/]U$; however, it was not able to then infer $C \to FFfFFfFFfFF[\text{-}F^4]fFFfFF[+F^3]fFFfFF[\text{-}FF]fFFf$. This is interesting as the growth mechanisms might be more complicated for a human to infer than the lines represented by the F and f symbols. Therefore, PMIT is a useful aide to human experts even when it cannot infer the complete L-system.

For the generated models, Figure 2 gives one point for every L-system (generated or not) tested with PMIT. A model is considered *solved* if there is a 100% success rate and *unsolved* otherwise. It is evident that the figure shows a region described by alphabet size and longest successor length that PMIT can reliably solve. PMIT can infer L-systems with $|V| = 17$, if the successors are short (5) and can infer fairly long successors (25) when $|V| = 3$. Computing the *sum of successor words* $\sum_{A \in V} |succ(A)|$, then PMIT is able to infer L-systems where such a sum is less than 140, which compares favorably to approaches in literature where the sum is at most 20. Overall, in terms of MTTS, PMIT is generally slower than LGIN [9] for $|V| = 2$ although is still practically fast for these L-systems (less than 35 seconds); however, PMIT can reliably infer L-systems with larger alphabet sizes and successor lengths and still does so with an average of 17.762 seconds. Finally, the brute force algorithm required a MTTS of 621.998 seconds. Hence, the logical rules and the GA provide considerable improvement.

6 Conclusions and Future Directions

This paper introduced the Plant Model Inference Tool (PMIT) as a hybrid approach, combining GA and logical rules, to infer deterministic context-free L-systems. PMIT can infer systems where the sum of the successor lengths is less than or equal to 140 symbols. This compares favourably to existing approaches that are limited to one or two symbol alphabets, and a total successor length less than or equal to 20 [9, 13]. Although PMIT is slower than existing approaches for "Fractal Plant" which has a small (2) alphabet [9, 13] with a MTTS of 35

Model	PMIT			Brute Force		LGIN [9]
	SR	MTTS (s)	Infer Growth	SR	MTTS (s)	MTTS (s)
Algae [11]	100%	0.001	n/a	100%	0.001	-
Cantor Dust [11]	100%	0.001	n/a	100%	0.001	-
Dragon Curve [11]	100%	0.909	n/a	100%	4.181	-
E-Curve [11]	0%	14400	Yes	0%	14400	-
Fractal Plant v1 [9, 11]	100%	33.680	n/a	100%	163.498	2.834
Fractal Plant v2 [9, 11]	100%	0.021	n/a	100%	5.019	0.078
Fractal Plant v3 [9, 11]	100%	0.023	n/a	100%	5.290	0.120
Fractal Plant v4 [9, 11]	100%	0.042	n/a	100%	6.571	0.414
Fractal Plant v5 [9, 11]	100%	34.952	n/a	100%	171.003	0.406
Fractal Plant v6 [9, 11]	100%	31.107	n/a	100%	174.976	0.397
Gosper Curve [11]	100%	71.354	n/a	100%	921.911	-
Koch Curve [11]	100%	0.003	n/a	100%	0.023	-
Peano [11]	0%	14400	Yes	0%	14400	-
Pythagoras Tree [11]	100%	0.041	n/a	100%	2.894	-
Sierpenski Triangle v1 [11]	100%	2.628	n/a	100%	267.629	-
Sierpenski Triangle v2 [11]	100%	0.086	n/a	100%	128.043	-
Aphanocladia [14]	0%	54.044	Yes	0%	14400	-
Dipterosiphonia v1 [14]	0%	14400	No	0%	14400	-
Dipterosiphonia v2 [14]	0%	14400	Yes	0%	14400	-
Ditira Reptans [14]	100%	73.821	n/a	100%	6856.943	-
Ditira Zonaricola [14]	0%	74.006	Yes	0%	14400	-
Herpopteros [14]	0%	81.530	Yes	0%	14400	-
Herposiphonia [14]	0%	298.114	Yes	0%	14400	-
Metamorphe [14]	0%	14400	Yes	0%	14400	-
Pterocladellium [14]	0%	14400	No	0%	14400	-
Tenuissimum [14]	0%	14400	No	0%	14400	-
Apple Twig [14]	0%	14400	No	0%	14400	-
Fibonacci Bush [14]	0%	14400	Yes	0%	14400	-

Table 1. Results for PMIT, Brute Force, and LGIN [9], on existing L-system models.

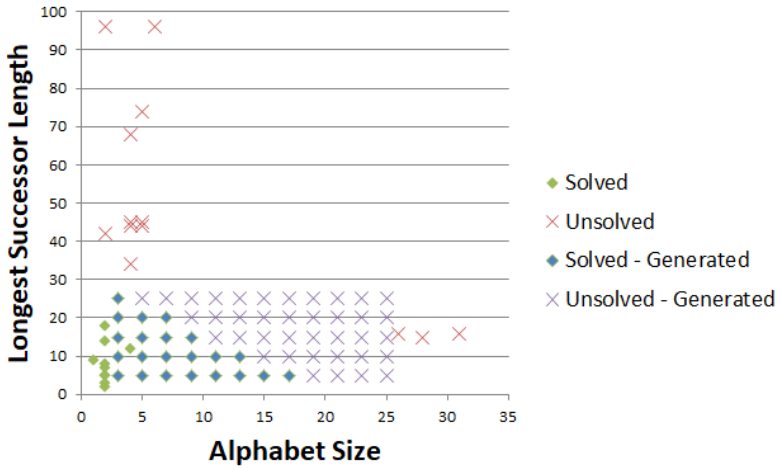


Fig. 2. L-Systems Solved with 100% SR by alphabet size and longest successor length.

seconds or less compared to 2 seconds or less, PMIT is still practically fast. Furthermore, existing approaches are limited to 2 symbol alphabets while PMIT can infer some L-systems with up to 17 symbol alphabets with longer successors.

For future work, methods will be investigated to further extend the limits of alphabet size and successor length. Also, a main focus will be on the ability to properly infer the drawing pattern likely using image processing techniques, perhaps taking advantage of techniques devised here to sub-divide alphabets.

References

1. Allen, M.T., Prusinkiewicz, P., DeJong, T.M.: Using L-systems for modeling source-sink interactions, architecture and physiology of growing trees: The L-PEACH model. New Phytologist 166(3), 869–880 (2005)
2. Back, T.: Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press (1996)
3. Ben-Naoum, F.: A survey on L-system inference. INFOCOMP Journal of Computer Science 8(3), 29–39 (2009)
4. Bernard, J., McQuillan, I.: New techniques for inferring L-Systems using genetic algorithm (2017), `https://arxiv.org/pdf/1712.00180.pdf`
5. Doucet, P.: The syntactic inference problem for D0L-sequences. L Systems pp. 146–161 (1974)
6. Jacob, C.: Genetic L-system programming: breeding and evolving artificial flowers with Mathematica. In: Proceedings of the First International Mathematica Symposium. pp. 215–222 (1995)
7. Lindenmayer, A.: Mathematical models for cellular interaction in development, parts I and II. Journal of Theoretical Biology 18, 280–315 (1968)
8. Mock, K.J.: Wildwood: The evolution of L-system plants for virtual environments. In: Proceedings of the 1998 IEEE World Congress on Computational Intelligence. pp. 476–480. IEEE (1998)
9. Nakano, R., Yamada, N.: Number theory-based induction of deterministic context-free L-system grammar. In: International Conference on Knowledge Discovery and Information Retrieval. pp. 194–199. SCITEPRESS (2010)
10. Prusinkiewicz, P., Crawford, S., Smith, R., Ljung, K., Bennet, T., Ongaro, V., Leyser, O.: Control of bud activation by an auxin transport switch. Proceedings of the National Academy of Sciences 106(41), 17431–17436 (2009)
11. Prusinkiewicz, P., Lindenmayer, A.: The Algorithimic Beauty of Plants. Springer Verlag, New York (1990)
12. Prusinkiewicz, P., Mündermann, L., Karwowski, R., Lane, B.: The use of positional information in the modeling of plants. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. pp. 289–300. ACM (2001)
13. Runqiang, B., Chen, P., Burrage, K., Hanan, J., Room, P., Belward, J.: Derivation of L-system models from measurements of biological branching structures using genetic algorithms. In: Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. pp. 514–524. Springer (2002)
14. University of Calgary: Algorithmic Botany, `http://algorothmicbotany.org`

Surrogate-Assisted Particle Swarm with Local Search for Expensive Constrained Optimization

Rommel G. Regis

Saint Joseph's University, Philadelphia PA 19131, USA,

rregis@sju.edu

Abstract. This paper develops a surrogate-assisted particle swarm optimization framework for expensive constrained optimization called *CONOPUS (CONstrained Optimization by Particle swarm Using Surrogates)*. In each iteration, CONOPUS considers multiple trial positions for each particle in the swarm and uses surrogate models for the objective and constraint functions to identify the most promising trial position where the expensive functions are evaluated. Moreover, the current overall best position is refined by finding the minimum of the surrogate of the objective function within a neighborhood of that position and subject to surrogate inequality constraints with a small margin and with a distance requirement from all previously evaluated positions. CONOPUS is implemented using radial basis function (RBF) surrogates and the resulting algorithm compares favorably to alternative methods on 12 benchmark problems and on a large-scale application from the auto industry with 124 decision variables and 68 inequality constraints.

Keywords: particle swarm optimization, constrained optimization, surrogate model, radial basis function expensive function

1 Introduction

In many engineering optimization problems, the objective and constraint functions are *black-box* in that their mathematical expressions are not explicitly available. Moreover, accurate gradient information is often not available and so classical optimization methods are not applicable. Particle swarm optimization (PSO) (Kennedy and Eberhart [1]) is among the most popular metaheuristics for solving these problems. In the PSO paradigm, the population of solutions simulates the behavior of a swarm of agents or particles, such as a flock of birds or a school of fish, as they collectively attempt to find some optimal state.

Numerous variants of PSO have been proposed and shown to be effective on a wide variety of problems (e.g., [2], [3], [4], [5]). Moreover, many PSO methods have been developed to handle constraints (e.g., [6],[7],[8],[9]). Now there are many optimization problems for which the objective and constraint function values are obtained from time-consuming computer simulations. In these situations, only a relatively small number of simulations can be carried out for the optimization process. Hence, surrogates have been used to assist PSO by reducing the

number of function evaluations needed to obtain good solutions (e.g., Parno et al. [10], Jiang et al. [11], Tang et al. [12], Sun et al. [13]). These surrogate-assisted PSO methods are designed for bound-constrained problems where only the objective function is expensive. There are very few, if any, surrogate-assisted PSO approaches where surrogates are used to approximate both the objective and constraint functions. Moreover, there are relatively few surrogate-assisted PSO methods that can be used for high-dimensional problems with over 100 decision variables (e.g., Sun et al. [13]). However, there are surrogate-assisted evolutionary algorithms for constrained problems (e.g., Regis [14]) and non-evolutionary methods that use surrogates to model the objective and constraints (e.g., Basudhar et al. [15], Regis [16], Bagheri et al. [17]).

This paper solves constrained optimization problems of the form:

$$\min \{f(x) \ : \ G(x) = (g_1(x), \ldots, g_m(x)) \leq 0, \ \ell \leq x \leq u\} \tag{1}$$

where $f, g_1, \ldots, g_m$ are functions whose values at an input $x \in \mathbb{R}^d$ are obtained from a deterministic and expensive computer simulation. The region $[\ell, u] \subset \mathbb{R}^d$ defined by the bounds is referred to as the *search space* for problem (1). Here, one *simulation* for a given input $x \in [\ell, u]$ yields the values of $f(x)$ and $G(x)$. This paper assumes that accurate gradient information for the objective and constraint functions are not available. Problem (1) is denoted by CBOP$(f, G, [\ell, u])$.

Since standard PSO is not expected to be effective when the objective and constraint functions are expensive, this paper develops a surrogate-based approach called *CONOPUS (CONstrained Optimization by Particle swarm Using Surrogates)* to reduce the number of simulations in PSO for constrained problems. This method can be used for problems involving hundreds of decision variables and many black-box inequality constraints. In each iteration, CONOPUS considers multiple trial positions for each particle in the swarm and then uses surrogate models for the objective and constraint functions to identify the most promising trial position. The simulations yielding the objective and constraint function values are then performed only at these promising trial positions. Moreover, the current overall best position is refined by finding the minimum of the surrogate of the objective function within some search radius of that position and subject to surrogate inequality constraints with a small margin and with a distance requirement from previously visited positions. In the numerical experiments, CONOPUS is implemented using RBF surrogates and the resulting *CONOPUS-RBF* algorithm is compared to alternative methods, including APSO (Accelerated Particle Swarm Optimization) (Yang [18]) and another PSO for constrained problems, an RBF-assisted PSO without local refinement, and an RBF-assisted evolutionary algorithm called CEP-RBF (Regis [14]), on 12 benchmark problems and on the large-scale MOPTA08 problem from the auto industry (Jones [19]) with 124 decision variables and 68 black-box inequality constraints. The results show that CONOPUS-RBF outperforms the other PSO-based approaches and is competitive with CEP-RBF on the problems used.

2 Constrained Particle Swarm Using Surrogates

2.1 Overview of the Proposed Method

As mentioned above, the use of surrogates in PSO have mostly been limited to bound constrained problems where only the objective function is expensive. This paper develops a new surrogate-assisted PSO framework for constrained black-box optimization called *CONOPUS (CONstrained Optimization by Particle swarm Using Surrogates)* that extends the OPUS framework for bound-constrained black-box optimization (Regis [20]). As in OPUS, multiple trial positions for each particle are considered in each iteration. However, in CONOPUS, there are now surrogate models for each inequality constraint function in addition to the surrogate for the objective function. These surrogates are updated in every iteration and, for each particle, they are used to identify the most promising among a large number of trial positions for this particle. Then, each particle is moved to the most promising trial position and then the expensive simulation is carried out only at these promising positions. In addition, CONOPUS refines the current overall best position by finding a minimizer of the updated surrogate model of the objective function within a relatively small radius around that position (and within the bounds), subject to surrogate inequality constraints with a small margin, and subject to a distance requirement from previously evaluated points. The idea of a margin for the constraints was introduced in Regis [16] and it is meant to facilitate the generation of feasible sample points, while the distance requirement is meant to prevent the algorithm from generating sample points that are close to previous sample points. The solution to this optimization subproblem is referred to as a *local refinement point*. The expensive simulation is then also carried out at this point. Hence, CONOPUS is essentially an accelerated PSO for constrained problems with the surrogates guiding where each particle should go and helping to refine the current overall best position.

2.2 Algorithmic Framework

A constrained optimization algorithm needs to be able to compare two infeasible solutions in the search space $[\ell, u]$ and determine which one is more desirable. This can be accomplished by means of a *constraint violation (CV) function*, denoted by $V_G(x)$, which measures the degree of constraint violation of a point $x \in [\ell, u]$ with respect to the constraint function $G(x)$. Commonly used examples are $V_G(x) = \sum_{j=1}^{m} [\max\{g_j(x), 0\}]$ and $V_G(x) = \sum_{j=1}^{m} [\max\{g_j(x), 0\}]^2$, and here, the former is used. Now given the objective function f and a CV function V_G, the definition below clarifies what is meant by an improving solution.

Definition 1. *Let $[\ell, u] \subseteq \mathbb{R}^d$ be the search space and let $G(x)$ be the constraint function for problem (1). Moreover, let $\mathcal{D} = \left\{x \in \mathbb{R}^d : \ \ell \le x \le u, \ G(x) \le 0\right\}$ be the feasible region of the problem. A point $x_1 \in [\ell, u]$ is an improvement over $x_2 \in [\ell, u]$ if one of the following conditions hold: (a) $x_1, x_2 \in \mathcal{D}$ and $f(x_1) < f(x_2)$; (b) $x_1 \in \mathcal{D}$ but $x_2 \notin \mathcal{D}$; or (c) $x_1, x_2 \notin \mathcal{D}$ and $V_G(x_1) < V_G(x_2)$.*

Below (Algorithm 1) is the CONOPUS framework for constrained PSO using surrogates that extends the OPUS framework in Regis [20] to constrained optimization. In every iteration of a PSO algorithm, each particle is represented as a point in the search space with an associated velocity vector. This velocity vector is updated by using a linear combination of the velocity in the previous iteration, the direction of the best position so far of the particle, and the direction of the best position so far of any of the particles. The weights for the last two components of this linear combination vary randomly from iteration to allow for the exploration of the search space. Assume for now that a feasible starting point is given but the method can be extended to deal with infeasible starting points by considering a two-phase approach as in Regis [16] where the first phase consists of finding a feasible point while the second phase proceeds in the same manner described below.

In the notation below, s denotes the number of particles and t denotes the time period. Here, only discrete time periods $t = 0, 1, 2, \ldots$ are considered. Moreover, $x^{(i)}(t)$ represents the position of particle i, where $i = 1, \ldots, s$, during time t and $x_j^{(i)}(t)$ represents the jth coordinate or component of $x^{(i)}(t)$, where $j = 1, \ldots, d$. That is, $x^{(i)}(t) = (x_1^{(i)}(t), x_2^{(i)}(t), \ldots, x_d^{(i)}(t))$. Moreover, $y^{(i)}(t)$ is the best position visited by particle i while $\widehat{y}(t)$ is the best position visited by any of the particles up to time t. In addition, $v^{(i,\ell)}(t)$ and $x^{(i,\ell)}(t)$ are the ℓth trial velocity and ℓth trial position, respectively, for particle i during time t.

The CONOPUS framework begins by evaluating the points of the given space-filling design over the search space $[\ell, u]$ (Step 1). Then the initial swarm positions are selected to be the s best points of the space-filling design with respect to the objective function f and constraint violation function V_G (Step 2). In Step 3, the initial particle velocities are determined using the half-diff method [21]. Next, in Step 4, the best position for each particle is initialized to the starting position of the particle. Moreover, the overall best position is initialized to the best position among the starting positions in terms of f and V_G. In addition, the collection of local refinement points $\mathcal{E}_0$ is initialized to the empty set.

Next, Step 5 fits $m + 1$ surrogate models $s_t^{(0)}(x), s_t^{(1)}, \ldots, s_t^{(m)}$, one for the objective function and one for each of the constraint functions, using all available data points. These data points come from all positions visited by any particle (given by $\bigcup_{j=0}^{t} \bigcup_{i=1}^{s} \{x^{(i)}(j)\}$) and from all local refinement points (given by $\mathcal{E}_t$). Then, Step 6 determines the new position of each particle by first considering multiple trial velocities within the velocity limits for that particle (Step 6.1(a)), generating the corresponding trial positions (Step 6.1(b)), projecting the trial positions into the bounds in case they leave the search space (Step 6.1(c)), and then using the surrogate to select the most promising among the trial positions and then choosing this to be the new position of the given particle (Step 6.2). Once the new positions for the particles have been determined, the simulator is then run at these positions to obtain the objective and constraint function values (Step 7). Again, the best position for each particle and the overall best position by any particle are updated (Step 8).

Algorithm 1 CONstrained Optimization by Particle swarm Using Surrogates.

Inputs: (1) CBOP$(f, G, [\ell, u])$; (2) CV function $V_G(x)$; (3) population size: s; (4) space-filling design: $\{z^{(1)}, \ldots, z^{(k)}\} \subseteq [\ell, u]$ with $k \geq s$; (5) inertial weighting factor for each iteration: $i(t)$, where t is the iteration number; (6) cognition parameter: μ; (7) social parameter: ν; (8) minimum and maximum velocities: v_{min} and v_{max}; (9) number of trial positions for each particle: r; (10) type of surrogate model; (11) optimization solver for local refinement; (12) search radius for local refinement: $\Delta > 0$; (13) distance requirement from previous sample points: $\xi > 0$; (14) initial margin for the surrogate inequality constraints: $\epsilon > 0$; (15) distance threshold to determine if points are too close: $\delta > 0$; (16) maximum iterations: T_{max}

Output: The best point found by the algorithm.

1. Evaluate Design. For $i = 1, \ldots, k$, run simulator to obtain $f(z^{(i)})$ and $G(z^{(i)})$.

2. Determine Initial Swarm Positions. Choose initial swarm positions $x^{(1)}(0)$, $\ldots, x^{(s)}(0)$ to be the s best points from $\{z^{(1)}, \ldots, z^{(k)}\}$ according to f and V_G.

3. Determine Initial Particle Velocities. For $i = 1, \ldots, s$, generate $u^{(i)}$ uniformly at random on $[\ell, u]$ and set $v^{(i)}(0) = \frac{1}{2}(u^{(i)} - x^{(i)}(0))$.

4. Initialize Best Position for Each Particle and Overall Best. Set $y^{(i)}(0) = x^{(i)}(0)$, $i = 1, \ldots, s$, and let $\widehat{y}(0)$ be the best point in $\{y^{(1)}(0), \ldots, y^{(s)}(0)\}$ with respect to f and V_G. Set the iteration counter $t = 0$ and $\mathcal{E}_t = \emptyset$.

5. Fit Surrogates. Use all previous sample points $\left(\bigcup_{j=0}^{t} \bigcup_{i=1}^{s} \{x^{(i)}(j)\}\right) \bigcup \mathcal{E}_t$ to build surrogates $s_t^{(0)}(x), s_t^{(1)}, \ldots, s_t^{(m)}$ for the objective and constraint functions.

6. Determine New Particle Positions. For $i = 1, \ldots, s$

6.1 Generate Trial Positions. For $\ell = 1, \ldots, r$

(a) **(Generate Trial Velocities)** For $j = 1, \ldots, d$

$$v_j^{(i,\ell)}(t+1) = i(t) v_j^{(i)}(t) + \mu \omega_{1,j}^{(i)}(t)(y_j^{(i)}(t) - x_j^{(i)}(t)) + \nu \omega_{2,j}^{(i)}(t)(\widehat{y}_j(t) - x_j^{(i)}(t)), \text{ where } \omega_{1,j}^{(i)}(t),\ \omega_{2,j}^{(i)}(t) \sim U[0,1]$$

$$v_j^{(i,\ell)}(t+1) = \min(\max(v_{min}, v_j^{(i,\ell)}(t+1)), v_{max})$$

End for.

(b) **(Generate Trial Positions)** $x^{(i,\ell)}(t+1) = x^{(i)}(t) + v^{(i,\ell)}(t+1)$

(c) **(Project Trial Positions)** $x^{(i,\ell)}(t+1) = \text{proj}_{[\ell,u]}(x^{(i,\ell)}(t+1))$

End for.

6.2 Select Promising Position Using Surrogate. Use the surrogate model $s_t(x)$ to select the most promising trial position for particle i among the points $\{x^{(i,1)}(t+1), x^{(i,2)}(t+1), \ldots, x^{(i,r)}(t+1)\}$. Let $x^{(i)}(t+1)$ be the most promising trial position and let $v^{(i)}(t+1)$ be the associated trial velocity.

7. Evaluate Swarm Positions. For each $i = 1, \ldots, s$, run the simulator to obtain $f(x^{(i)}(t+1))$ and $G(x^{(i)}(t+1))$.

8. Update Best Position for Each Particle and Overall Best. Set $\widehat{y}(t+1) = \widehat{y}(t)$. For $i = 1, \ldots, s$

(a) If $x^{(i)}(t+1)$ is an improvement over $y^{(i)}(t+1)$, then

Set $y^{(i)}(t+1) = x^{(i)}(t+1)$.

If $y^{(i)}(t+1)$ is an improvement over $\widehat{y}(t+1)$, then $\widehat{y}(t+1) = y^{(i)}(t+1)$.

(b) Else

Set $y^{(i)}(t+1) = y^{(i)}(t)$.

End if.

9. Refit Surrogates. Use all previous sample points $\left(\bigcup_{j=0}^{t+1} \bigcup_{i=1}^{s} \{x^{(i)}(j)\}\right) \bigcup \mathcal{E}_t$ to refit surrogates $s_t^{(0)}(x), s_t^{(1)}, \ldots, s_t^{(m)}$ for the objective and constraint functions.

Algorithm 2 CONOPUS algorithm (continued)

10. Perform Local Refinement of Overall Best Position. Relabel all previous sample points by $v_1, \ldots, v_n$ and let v_n^* be the best feasible point so far. Solve the subproblem:

$$\begin{gathered} \min\ s_t^{(0)}(x) \\ \text{s.t.} \quad x \in \mathbb{R}^d,\ \ell \leq x \leq u \\ \|x - v_n^*\| \leq \Delta, \quad \|x - v_j\| \geq \xi,\ j = 1, \ldots, n \\ s_t^{(i)}(x) + \epsilon \leq 0,\ i = 1, 2, \ldots, m \end{gathered} \tag{2}$$

11. Check if Feasible Solution to Subproblem was Found. If a feasible solution is found for Problem (2), then let x_{t+1}^* be the solution obtained. Otherwise, let x_{t+1}^* be the best solution with respect to f and V_G (infeasible for (2)) among a set of randomly generated points within the search region $\{x \in [\ell, u] \ : \ \|x - v_n^*\| \leq \Delta\}$.

12. Determine if Minimizer of Surrogate is Far From Previous Points. If x_{t+1}^* is at least of distance δ from all previously evaluated points, then do

12.1 Evaluate Minimizer of Surrogate. Run the simulator to obtain $f(x_{t+1}^*)$ and $G(x_{t+1}^*)$.

12.2 Update Overall Best Position and Local Refinement Points. If x_{t+1}^* is an improvement over $\widehat{y}(t+1)$, then $\widehat{y}(t+1) = x_{t+1}^*$ and set $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{x_{t+1}^*\}$. Else, set $\mathcal{E}_{t+1} = \mathcal{E}_t$.

13. Check Termination Condition. If $t < T_{max}$, then reset $t \leftarrow t + 1$ and go back to Step 5. Else, STOP.

The algorithm then refits the surrogate models $s_t^{(0)}(x), s_t^{(1)}, \ldots, s_t^{(m)}$ to incorporate the newly evaluated points (Step 9) in preparation for local refinement of the overall best point (Step 10). In this step, an optimization solver finds a global minimizer x_{t+1}^* of the surrogate for the objective $s_t^{(0)}(x)$ within a ball of radius Δ centered at the current overall best point $\widehat{y}(t+1)$, within the search space, subject to surrogate inequality constraints with a margin ϵ, and with a distance requirement of ξ from all previous sample points. In the numerical implementation, it is enough to find an approximate solution to this optimization subproblem. If the subproblem solution (local refinement point) is not too close to any previous sample point (at least distance δ), then the simulator is run at this point (Step 12.1), and then the overall best position and the set of local refinement points are also updated (Step 12.2). Finally, the algorithm goes back to Step 5 if the termination condition has not been satisfied. Otherwise, the algorithm stops and returns the overall best position found (Step 13).

2.3 A Radial Basis Function Model

CONOPUS can be implemented using any type of surrogate model that is continuously differentiable and whose gradients are easy to calculate. This study uses the radial basis function (RBF) interpolation model in [22] and the resulting algorithm is referred to as CONOPUS-RBF. This RBF model has been used in

various surrogate-based optimization methods (e.g., [14, 16]). Fitting this model involves solving a linear system with good theoretical properties and it differs from the typical training methods for RBF networks in machine learning.

Suppose we are given n distinct points $u^{(1)}, \ldots, u^{(n)} \in \mathbb{R}^d$ with function values $h(u^{(1)}), \ldots, h(u^{(n)})$, where h is the objective or one of the constraint functions. CONOPUS-RBF uses an interpolant of the form

$$s(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - u^{(i)}\|) + p(x), \qquad x \in \mathbb{R}^d, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm, $\lambda_i \in \mathbb{R}$ for $i = 1, \ldots, n$, $p(x)$ is a linear polynomial in d variables, and ϕ has the *cubic* form: $\phi(r) = r^3$. Other possible choices for ϕ include the thin plate spline, multiquadric and Gaussian forms. We use a cubic RBF because of previous success with this model (e.g., [14, 20]).

3 Numerical Experiments

CONOPUS-RBF is compared with alternative methods on the MOPTA08 benchmark problem [19] from the auto industry. The MOPTA08 problem involves finding the values of the decision variables (e.g., shape variables) that minimize the mass of the vehicle subject to performance constraints (e.g., crashworthiness, durability). It has one black-box objective function to be minimized, 124 decision variables that take values on a continuous scale from 0 to 1, and 68 black-box inequality constraints that are well normalized [19]. A Fortran code for this problem is available at `http://www.miguelanjos.com/jones-benchmark`.

CONOPUS-RBF is also compared with the alternative methods on 12 test problems used in Regis [14]. These include G7, G8, G9, G10, four 30-D problems from the CEC 2010 benchmark [23] (C07, C08, C14 and C15) and four design problems, namely, Welded Beam, GTCD (Gas Transmission Compressor Design), Pressure Vessel, and Speed Reducer. The number of decision variables, number of inequality constraints, the region defined by the bounds, and the best known feasible objective values for these problems are given in Regis [14].

To evaluate the effectiveness of the RBF surrogate strategy, CONOPUS-RBF is compared with CONPSO, which is a standard PSO for constrained problems obtained by removing the trial solutions and RBF surrogates in CONOPUS-RBF and also the local refinement phase. To assess the effectiveness of the local refinement strategy, CONOPUS-RBF is also compared with an RBF-assisted extension of CONPSO without local refinement called CONPSO-RBF. Moreover, it is compared with an RBF-assisted evolutionary algorithm called CEP-RBF [14] and with Accelerated Particle Swarm Optimization (APSO)[18].

All computational runs are carried out in Matlab 8.2 on an Intel(R) Core(TM) i7-4770 CPU 3.40 GHz 3.00 GHz desktop machine. Each method is run for 30 trials on all problems. The RBF-assisted methods are all initialized using an affinely independent Latin Hypercube Design (LHD) with $d + 1$ points. The initial population of particles is chosen as a subset of the LHD with the best

objective function values. If there are not enough LHD points to form the initial population, it is augmented by uniform random points over the search space. The LHD is not needed by CONPSO and other non-surrogate methods. However, experiments on the test problems suggest that the performance of PSO when initialized by uniform random points over the search space is similar to its performance when initialized by an LHD. To ensure fair comparison, all methods use the same LHD in a given trial, but different LHDs are used in different trials.

In the numerical experiments, the population size for CONOPUS-RBF, CONPSO-RBF and CONPSO is set to $s = 5, 10, 20$ (e.g., [7]). In some PSO implementations, the inertial weighting factor $i(t)$ varies with the iterations from a high value (close to 1) to a low value. Here, it is fixed at $i(t) = 0.72984$ and the cognition and social parameters are set to $\mu = \nu = 1.496172$ as recommended in [24]. The minimum and maximum values for the components of the velocity vectors are set to $\mp \min_{1 \le i \le d} (u_i - \ell_i)/4$, respectively. For CONOPUS-RBF, the number of trial positions for each particle is $r = 10d$, the search radius for local refinement is $\Delta = 0.05 \min_{1 \le i \le d} (u_i - \ell_i)$, and the distance requirement from previous sample points is $\xi = 0.0005 \min_{1 \le i \le d} (u_i - \ell_i)$. For CEP-RBF, the parameters are $\mu = 5$ parent solutions in each generation, and the number of trial offspring for each parent in each generation is $\nu = \min(1000d, 10000)$.

Parameter tuning can be used to obtain better algorithm performance when the computational budget is limited [25]. However, for truly expensive functions, this may not always be feasible and one can use parameter settings that are reasonable based on previous algorithm performance. Besides, finding the best parameter settings for CONOPUS-RBF is beyond the scope of this paper, and our goal is *not* to show that it always outperforms other methods. Rather, we wish to demonstrate that surrogates dramatically improve the performance of PSO on constrained problems and that the resulting CONOPUS-RBF is a promising approach when the number of simulations is limited.

4 Results and Discussion

CONOPUS-RBF is compared with alternatives on the 12 test problems using data profiles [26]. To make it easier to present the results, two sets of comparisons were performed: (1) CONOPUS-RBF vs CONPSO-RBF with different population sizes ($s = 5, 10, 20$); and (2) CONOPUS-RBF vs other methods including APSO, CONPSO, CONPSO-RBF and CEP-RBF.

Now the *data profile of a solver s* [26] is the function

$$d_s(\alpha) = |\{p \in \mathcal{P} \ : \ t_{p,s} \le \alpha(n_p + 1)\}| \, / |\mathcal{P}|, \qquad \alpha > 0, \tag{4}$$

where $t_{p,s}$ is the number of simulations required by solver s to satisfy the convergence test defined below on problem p and n_p is the number of variables in problem p. For a given solver s and any $\alpha > 0$, $d_s(\alpha)$ is the fraction of problems "solved" by s within $\alpha(n_p + 1)$ simulations (equivalent to α simplex gradient

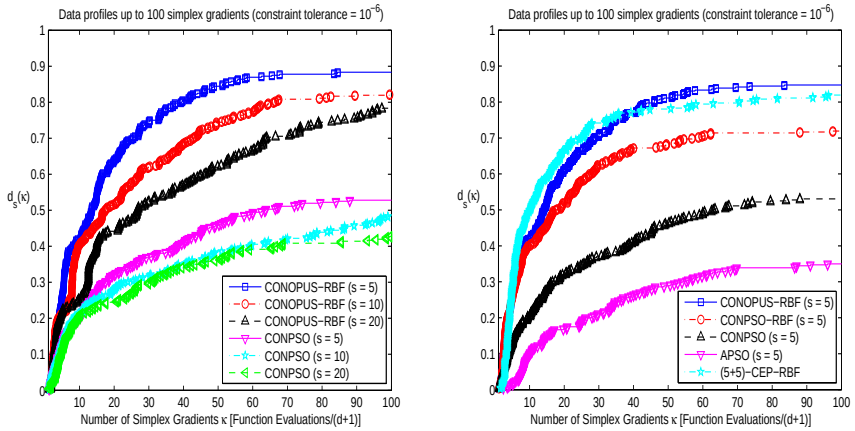


Fig. 1: Data profiles for optimization methods on the test problems.

estimates [26]). Here, "solved" means the solver generated a point satisfying the convergence test in Moré and Wild [26]. This test uses a tolerance $\tau > 0$ and the minimum feasible objective function value f_L obtained by *any* of the solvers on a particular problem within a given number of simulations μ_f and it checks if a feasible point x obtained by a solver satisfies $f(x^{(0)}) - f(x) \geq (1-\tau)(f(x^{(0)}) - f_L)$, where $x^{(0)}$ is the feasible starting point corresponding to the given problem. Here, x is required to achieve a reduction that is $1 - \tau$ times the best possible reduction $f(x^{(0)}) - f_L$. In this study, $\tau = 0.05$.

Figure 1 shows the data profiles of the various solvers on the test problems. These profiles clearly show that CONOPUS-RBF is a dramatic improvement over CONPSO for each of the population sizes $s = 5, 10, 20$. Moreover, for the test problems considered and when the computational budget is limited, the best results for CONOPUS-RBF and CONPSO are obtained when $s = 5$, followed by $s = 10$ and then $s = 20$. A possible explanation for this is that with a smaller population size, these algorithms are able to perform more iterations.

Next, Figure 1 shows that CONOPUS-RBF with $s = 5$ is slightly better than (5+5)-CEP-RBF after about 40 simplex gradient estimates and it is much better than both CONPSO and CONPSO-RBF with $s = 5$. In particular, after 100 simplex gradient estimates, CONOPUS-RBF with $s = 5$ satisfied the convergence test for about 85% of the problems compared to about 82% for (5+5)-CEP-RBF, about 72% for CONPSO-RBF with $s = 5$, about 53% for CONPSO and 35% for APSO. Additional comparisons between CONOPUS-RBF and CONPSO-RBF for $s = 5, 10, 20$ (not shown here) indicate that local refinement improves performance when the population size is small.

Friedman's nonparametric statistical test followed by a multiple comparison procedure was also performed to determine if the mean rank of CONOPUS-RBF is significantly better than that of other algorithms when $s = 5$ at a fixed computational budget of $15(d+1)$ for the CEC 2010 problems and $30(d+1)$ for the

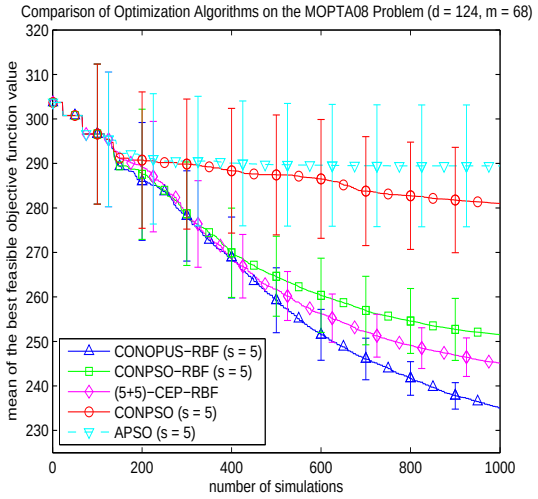


Fig. 2: Mean of the best feasible objective function value (over 10 trials) vs number of simulations for various optimization methods on the MOPTA08 optimization problem. Error bars represent 95% t-confidence intervals for the mean.

other test problems. The results show that CONOPUS-RBF is significantly better than CONPSO and APSO on most of the test problems and it is significantly better than CEP-RBF on three of the four CEC 2010 problems used.

Figure 2 shows the plot of the mean of the best objective function value (over 10 trials) obtained by each algorithm on the MOPTA08 problem as the number of simulations increases. The error bars are 95% t confidence intervals for the mean. This plot shows that on the MOPTA08 problem, CONOPUS-RBF with $s = 5$ is better than $(5 + 5)$-CEP-RBF followed by CONPSO-RBF with $s = 5$ and these RBF-assisted methods are dramatically much better than CONPSO and APSO with $s = 5$.

The advantage of CONOPUS-RBF over CEP-RBF may be partly due to the local refinement procedure. Incorporating local refinement in CEP-RBF might also improve its performance and the resulting algorithm might even outperform CONOPUS-RBF. However, the results suggest that CONOPUS-RBF will still be competitive with CEP-RBF with local refinement.

5 Summary and Future Work

This paper introduced the CONOPUS framework for a surrogate-assisted PSO for computationally expensive constrained optimization. This method generates a large number of trial positions for each particle in the swarm and uses surrogates for the objective and constraint functions to identify the most promising trial position for each particle. The function evaluations are then carried out only

on the promising trial positions. Moreover, at the end of each iteration, CONOPUS refines the current best position of all particles by finding an approximate minimizer of the surrogate of the objective function within some neighborhood of that best position and subject to surrogate inequality constraints with a small margin and with a distance requirement from all previously visited positions. CONOPUS was implemented using RBF surrogates and was shown to outperform the APSO algorithm and another constrained PSO with and without RBF surrogates (CONPSO and CONPSO-RBF) for small population sizes of $s = 5, 10$ and 20 on the test problems. Moreover, CONOPUS-RBF with $s = 5$ is competitive with a surrogate-assisted evolutionary algorithm CEP-RBF with the same population size. In addition, CONOPUS-RBF with $s = 5$ outperforms all these other alternatives on the large-scale MOPTA08 problem with 124 decision variables and 68 black-box inequality constraints. Overall, CONOPUS-RBF is a promising algorithm for constrained expensive black-box optimization.

Future work will explore other ways to incorporate surrogates within the PSO framework for constrained optimization and compare with other approaches such as CEP-RBF with local refinement. Moreover, one can consider extensions of the CONOPUS framework to multiobjective optimization and to problems where there is noise in the objective and constraint functions. Finally, one can also apply CONOPUS to other real-world optimization problems.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway, New Jersey, IEEE Service Center (1995) 1942–1948
2. Ismail, A., Engelbrecht, A.P.: Self-adaptive particle swarm optimization. In: Simulated Evolution and Learning, Lecture Notes in Computer Science. Volume 7673.
3. Qu, B.Y., Liang, J.J., Suganthan, P.N.: Niching particle swarm optimization with local search for multi-modal optimization. Information Sciences **197** (2012) 131–143
4. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization: an overview. Swarm Intelligence **1**(1) (2007) 33–57
5. Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization, part i: background and development. Natural Computing **6**(4) (2007) 467–484
6. He, Q., Wang, L.: A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation **186**(2) (2007) 1407–1422
7. Hu, X., Eberhart, R.C.: Solving constrained nonlinear optimization problems with particle swarm optimization. In Callaos, N., ed.: Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics. (2002) 203–206
8. Munoz-Zavala, A.E., Aguirre, A.H., Diharce, E.R.V.: Constrained optimization via particle evolutionary swarm optimization algorithm (peso). In Beyer, H.G., ed.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005). Volume 1., New York, NY, ACM Press (2005) 209216
9. Toscano-Pulido, G., Coello, C.A.C.: A constraint-handling mechanism for particle swarm optimization. In: Proceedings of the Congress on Evolutionary Computation

2004 (CEC 2004). Volume 2., Piscataway, New Jersey, IEEE Service Center (2004) 1396–1403
10. Parno, M.D., Hemker, T., Fowler, K.R.: Applicability of surrogates to improve efficiency of particle swarm optimization for simulation-based problems. Engineering Optimization **44**(5) (2012) 521–535
11. Jiang, P., Cao, L., Zhou, Q., Gao, Z., Rong, Y., Shao, X.: Optimization of welding process parameters by combining kriging surrogate with particle swarm optimization algorithm. The International Journal of Advanced Manufacturing Technology **86**(9) (2016) 2473–2483
12. Tang, Y., Chen, J., Wei, J.: A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions. Engineering Optimization **45**(5) (2013) 557–576
13. Sun, C., Jin, Y., Cheng, R., Ding, J., Zeng, J.: Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Transactions on Evolutionary Computation **21** (2017) 644–660
14. Regis, R.G.: Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. IEEE Transactions on Evolutionary Computation **18**(3) (2014) 326–347
15. Basudhar, A., Dribusch, C., Lacaze, S., Missoum, S.: Constrained efficient global optimization with support vector machines. Structural and Multidisciplinary Optimization **46**(2) (2012) 201–221
16. Regis, R.G.: Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Engineering Optimization **46**(2) (2014) 218–243
17. Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Applied Soft Computing **61** (2017) 377–393
18. Yang, X.S.: Nature-inspired metaheuristic algorithms, 2nd edition. Luniver Press (2010)
19. Jones, D.R.: Large-scale multi-disciplinary mass optimization in the auto industry. In: MOPTA 2008, Modeling and Optimization: Theory and Applications Conference, Ontario, Canada, MOPTA (August 2008)
20. Regis, R.G.: Particle swarm with radial basis function surrogates for expensive black-box optimization. Journal of Computational Science **5**(1) (2014) 12–23
21. Helwig, S., Wanka, R.: Theoretical analysis of initial particle swarm behavior. In: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN08), Dortmund, Springer (2008) 889–898
22. Powell, M.J.D.: The theory of radial basis function approximation in 1990. In Light, W., ed.: Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions. Oxford University Press, Oxford, U.K. (1992) 105–210
23. Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore (2010)
24. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: 2007 IEEE Swarm Intelligence Symposium. (April 2007) 120–127
25. Cáceres, L.P., López-Ibáñez, M., Stützle, T.: Ant colony optimization on a limited budget of evaluations. Swarm Intelligence **9** (2015) 103–124
26. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization **20**(1) (2009) 172–191

Evolutionary algorithms for scheduling of crude oil preheating process under linear fouling

Dimbalita Deka and Dilip Datta[0000−0002−9275−7697]

Department of Mechanical Engineering, Tezpur University, Tezpur - 784 028, India.
dimbalitadeka@gmail.com; ddatta@tezu.ernet.in/ datta_dilip@rediffmail.com

Abstract. The crude oil preheating process in refineries is required to be scheduled in a way to minimize the processing cost involved with it, subject to the satisfaction of various process related constraints. The process forms a mixed-integer optimization problem as the scheduling of the processing units involves binary variables, while the discharges from the running units are real valued. The two parts of such problems are usually handled by two different algorithms, where the optimum scheduling obtained by one algorithm is fed to another algorithm for optimizing its discharge process. In the present work, formulating the crude oil preheating process under the effect of linear fouling as a mixed-integer nonlinear programming (MINLP) model, three binary-real coded evolutionary algorithms (EAs) are investigated in order to demonstrate that a single EA can successfully tackle its both binary and real parts. Further, the statistical analysis of the performances of the EAs are also presented through their application to a benchmark instance of the problem.

Keywords: evolutionary algorithms, optimization, crude oil preheating process

1 Introduction

Evolutionary algorithms (EAs) are known to have the ability to find approximate solutions in reasonable time for such problems also, where classical optimization methods either become too expensive or even ineffective. EAs are usually independent of problem domains unlike classical optimization methods, which are restricted to specific classes of problems only. Hence, EAs have found applications in a wide range of real-life problems, including linear and nonlinear, convex and non-convex, continuous and discrete, and many more.

However, EAs still could not be generalized in case of many classes of discrete or mixed-discrete problems, but require the incorporation of some problem information for their effective performance. Unit scheduling of continuous flow process systems in industries is such a problem, which consists of two optimization sub-problems. The first part is the integer-valued scheduling of the processing units, while the second part is concerned with the optimization of the discharge process based on the scheduling of the first part. Accordingly, the optimization of an industrial continuous flow process system essentially becomes a

mixed-integer non-linear programming (MINLP) problem involving both integer variables to represent operational status of the units and real variables to represent the flows from the running units. Due to the complexities involved with such MINLP problems, two parts of a problem are often tackled separately through two different algorithms, where the first algorithm is employed to schedule the processing units over a time horizon and then the second algorithm optimizes the flow processes in the schedule of the first algorithm [1, 2]. However, such an isolating system may suffer from the drawback of missing better solutions as the possibility of more promising solutions cannot be denied if both the parts of the problem were tackled interactively [3, 4].

In view of above, three EAs are investigated here for handling an MINLP based two-step continuous flow process system by a single EA. The studied problem is the optimum scheduling of the crude oil preheating process arising in refineries, which is carried out through a crude preheat train (CPT) over a time horizon. The aim of preheating is to increase the crude oil temperature to a certain degree before its entry into a furnace, so that the energy (fuel) requirement in the furnace gets reduced. The CPT consists of a network of heat exchangers, commonly known as the heat exchanger network (HEN), to run a productive heat treatment process. The heat exchangers of HEN require periodic shutting down for the purpose of cleaning or other maintenance. This demands the effective scheduling of the HEN in order to get the optimum performance from the active units.

2 Literature review

In the case of EAs, mixed-integer problems involving distinct real and integer valued parts are often solved by hybridizing two optimization techniques, allowing one technique to handle the integer part and another to handle the real part. As an example, Trivedi et al. [5] solved the mixed-integer unit commitment problem, where binary variables are evolved using a genetic algorithm (GA) and the continuous variables using a differential evolution (DE). Similar hybridization procedures are found in many other works, such as hybridization of GA and particle swarm optimization (PSO) [6, 7], artificial bee colony (ABC) and GA [8], and DE and PSO [9].

Some works are also found where both integer and real parts of mixed-integer problems are handled by a single algorithm [3, 4]. However, no such work on scheduling the crude oil preheating process in refineries could be found in specialized literature.

3 Problem description and formulation

The studied problem of crude oil preheating process in a CPT is adopted from Smaili et al. [10], which is shown schematically in Fig. 1. In this problem, the raw crude oil passes through 14 heat exchangers (marked in Fig. 1 by 1 to 14), where it is preheated by 7 heating streams (marked in Fig. 1 by H1 to H7)

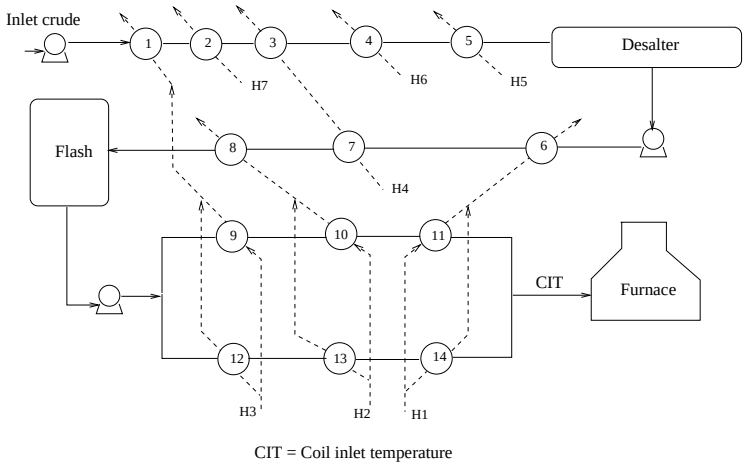


Fig. 1. Crude oil preheating process in a CPT [10].

prior to entering into the furnace. Heat exchangers 1–8 are connected in series; while the remaining 6 heat exchangers are arranged in two parallel lines, one containing heat exchangers 9–11 and the other connecting heat exchangers 12–14. The desalter and flash used in the processing line remove, respectively, any salt and vapour dissolved in the crude oil. The preheated crude oil is then burnt in the furnace at a higher temperature, after which it is distilled into different products.

During the preheating process of the crude oil, some impurities mixed with the crude oil get precipitated/deposited on the inner surfaces of the heat exchangers, which is called fouling. Such deposition forms a thick layer over time, which gradually reduces the performance of heat exchangers. In other words, the crude oil cannot be heated up to the possible level, which consequently increases the energy requirement in the furnace, thus increasing the energy cost. Further, the periodic cleaning of the heat exchangers for mitigating fouling is associated with cleaning cost. Hence, the process needs optimization for minimizing the total operational cost (i.e., the total of energy cost and cleaning cost) subject to some processing constraints.

Since the process is operated continuously over several years without any interruption, it can be considered that a cycle of a shorter time period is repeated in the entire time horizon. For the purpose of analysis, the cycle can further be divided equally into a certain number of time instants. At a time instant, a unit (heat exchanger) will remain either in full operation or partially/fully shutdown. In a shutting down instant, a unit may go through cleaning process also. Accordingly, the general optimization problem of a cycle of the process can be defined as follows:

– Determine
  1. Operational status of each unit at every time instant.

2. Outlet temperature of the crude from the CPT at every time instant.

– To minimize total cost (cleaning cost plus energy cost).
– Subject to
  1. Limit on operational units in each series segment at every instant.
  2. Limit on operational units in each parallel segment at every instant.
  3. Limit on operational units in each heating line at every instant.
  4. Limit on crude oil temperature from the outlet of the CPT.
  5. Limit on cleaning instants of each unit in the entire time horizon.
  6. Limit on cleaning a unit at consecutive time instants.

The above optimization problem is formulated in Eqs. (1) and (2).

$$\text{Minimize} \qquad f = C^{\text{cl}} \sum_{i=1}^{N} \sum_{t=1}^{T} \beta_{it} + C^{\text{ener}} F_N^{\text{f}} c_N^{\text{f}} \sum_{t=1}^{T} \left(\Theta^{\text{fmax}} - \Theta_{Nt}^{\text{f,out}} \right) \tag{1}$$

$$\text{Subject to} \qquad g_1 \equiv \sum_{j=1}^{\text{nsu}_i} u_{\text{su}_{ij},t} \geqslant \text{su}_i^{\text{on}} \; ; \qquad t = 1, 2, \cdots, T \; ; \; i = 1, 2, \cdots, \text{ns} \tag{2a}$$

$$g_2 \equiv \sum_{k=1}^{\text{npsu}_{ij}} u_{\text{psu}_{ijk},t} \geqslant \text{psu}_{ij}^{\text{on}} \; ; \; t = 1, 2, \cdots, T \; ; \; j = 1, 2, \cdots, \text{nps}_i \; ; \; i = 1, 2, \cdots, \text{npl} \tag{2b}$$

$$g_3 \equiv \sum_{j=1}^{\text{nhlu}_i} u_{\text{hlu}_{ij},t} \geqslant \text{hlu}_i^{\text{on}} \; ; \qquad t = 1, 2, \cdots, T \; ; \; i = 1, 2, \cdots, \text{nhl} \tag{2c}$$

$$g_4 \equiv \Theta_t^{\text{ffinal}} \leqslant \Theta^{\text{fmax}} \; ; \qquad t = 1, 2, \cdots, T \tag{2d}$$

$$g_5 \equiv \sum_{t=1}^{T} v_{it} \geqslant 1 \; ; \qquad i = 1, 2, \cdots, N \tag{2e}$$

$$g_6 \equiv (1 - u_{ip})(1 - u_{it}) \neq 0 \; ; \; p = \begin{cases} T \; ; & \text{if } t = 1 \\ t - 1 \; ; & \text{otherwise.} \end{cases} \; t = 1, 2, \cdots, T \; ; \; i = 1, 2, \cdots, N \tag{2f}$$

The objective function, f, in Eq. (1) represents the total operational cost, where the two summing terms on the right side represent the cleaning cost and energy cost, respectively. The constraints, g_1–g_3, in Eqs. (2a)–(2c) represent, respectively, the minimum number of operational units in series segments, parallel segments, and heating medium flow lines; while the constraints, g_4–g_6, in Eqs. (2d)–(2f) ensure the specified temperature of the crude oil at the outlet of the CPT, cleaning of each unit at least once in the entire time horizon, and avoiding the cleaning of a unit at two consecutive time instants, respectively.

In Eqs. (1) and (2), N and T are respectively the total number of units (heat exchangers) and time instants in a production cycle, C^{cl} is the cleaning cost coefficient per cleaning instant for the ith unit at the tth time instant (in practice, C^{cl} may remain same in all units and time instants), C^{ener} is the energy cost coefficient per unit of energy requirement, F_N^{f} is the flow rate in the last unit (Nth unit), c_N^{f} is the specific heat transfer capacity of the crude oil in the last unit, $\Theta_{Nt}^{\mathrm{f,out}}$ is the crude outlet temperature from the last unit at the tth time instant, Θ^{fmax} is the temperature up to which the crude is to be heated in the furnace, ns is the number of series segments, nsu_i is the number of units in the ith series segment with su_{ij} as its jth unit and $\mathrm{su}_i^{\mathrm{on}}$ as the required minimum number of operational units, npl is the number of parallel segments, nps_i is the number of branches in the ith parallel segment with npsu_{ij} as the number of units in its jth branch and psu_{ijk} as the kth unit while $\mathrm{psu}_{ij}^{\mathrm{on}}$ as the required minimum number of operational units in that branch, nhl is the number of heating lines with nhlu_i as the number of units in the ith heating line and hlu_{ij} as the jth unit and $\mathrm{hlu}_i^{\mathrm{on}}$ as the required minimum number of operational units in that heating line.

The cleaning time (β_{it}), operational status (u_{it}) and cleaning status (v_{it}) of the units, as used in Eqs. (1) and (2), are expressed by Eq. (3), where $v_{it} = 1$ means that the ith unit will be cleaned at the tth time instant.

$$u_{it} = \begin{cases} 1\ ; & \text{if the } i\text{th unit is fully in operation} \\ 0\ ; & \text{if the } i\text{th unit is shutdown partially} \\ & t = 1, 2, \cdots, T\ ;\ i = 1, 2, \cdots, N\ . \end{cases} \tag{3a}$$

$$v_{it} = \begin{cases} 0\ ; & \text{if } \ u_{it} = 1 \\ \{0, 1\}\ ; & \text{otherwise} \\ & t = 1, 2, \cdots, T\ ;\ i = 1, 2, \cdots, N\ . \end{cases} \tag{3b}$$

$$\beta_{it} = \begin{cases} 0\ ; & \text{if } \ v_{it} = 0 \\ \in [0, \alpha_{it}]\ ; & \text{otherwise} \\ & t = 1, 2, \cdots, T\ ;\ i = 1, 2, \cdots, N\ . \end{cases} \tag{3c}$$

For obtaining the crude oil outlet temperatures from the last unit at different time instants, $\Theta_{Nt}^{\mathrm{f,out}}$ used in Eq. (1) and (2d), the same for different units are computed using Eq. (4a), where $t = 1, 2, \cdots, T$ and $i = 1, 2, \cdots, N$.

$$\Theta_{it}^{\mathrm{f,out}} = \begin{cases} \Theta_{it}^{\mathrm{f,in}}\ ; & \text{if } u_{it} = 0 \\ \phi_{it}^{h}\Theta_{i}^{\mathrm{hinit}}(1 - \alpha_{it}) + \{\alpha_{it} + (1 - \alpha_{it})\phi_{it}^{c}\}\,\Theta_{it}^{\mathrm{f,in}}\ ; & \text{otherwise.} \end{cases} \tag{4a}$$

$$\text{where,} \qquad \Theta_{it}^{\mathrm{f,in}} = \begin{cases} \Theta^{\mathrm{f,inlet}}\ ; & \text{if } i = 1 \\ \Theta_{i-1,t}^{\mathrm{f,out}}\ ; & i \in \{2 - 14\ ; i \neq 6, 12\} \\ \Theta_{5,t}^{\mathrm{f,out}} - \Theta^{\mathrm{desalter}}\ ; & \text{if } i = 6 \\ \Theta_{8,t}^{\mathrm{f,out}}\ ; & \text{if } i = 12 \end{cases} \tag{4b}$$

$$\phi_{it}^{h} = \frac{(1-x)C_{\min}}{(1-xR_i)c_i^{\mathrm{f}}} \tag{4c}$$

$$\phi_{it}^{c} = \frac{(1-xR_i)c_i^{\mathrm{f}} - (1-x)C_{\min}}{(1-xR_i)c^{\mathrm{f}}} \tag{4d}$$

$$C_{\min} = \min\{F_i^{\mathrm{f}} c_i^{\mathrm{f}}, F_i^{\mathrm{h}} c_i^{\mathrm{h}}\} \tag{4e}$$

$$x = \exp\left\{-\frac{h_{it}A_i}{C_{\min}}(1-R_i)\right\} \tag{4f}$$

$$\mathrm{R}_i = \frac{C_{\min}}{C_{\max}} \tag{4g}$$

$$\alpha_{it} = \begin{cases} 0\ ; & \text{if } \ u_{it} = 1 \\ \in (0,1)\ ; & \text{otherwise} \\ & t = 1, 2, \cdots, T\ ;\ i = 1, 2, \cdots, N\ . \end{cases} \tag{4h}$$

In Eq. (4), $\Theta_{it}^{\mathrm{f,out}}$ is the crude outlet temperature from ith unit at tth time instant, $\Theta_{it}^{\mathrm{f,in}}$ is the crude oil inlet temperature of ith unit at tth time instant, $\Theta_i^{\mathrm{hinit}}$ is the initial temperature of heating medium of ith unit, α_{it} is the partial shutdown time during operation, $\Theta^{\mathrm{f,inlet}}$ is the crude oil temperature at the inlet of the CPT, $\Theta^{\mathrm{desalter}}$ is the temperature drop in desalter, $C_{\min}$ is the minimum heat capacity rate, $C_{\max}$ is the maximum heat capacity rate, F_i^{f} is the flow rate of crude oil of ith unit, c_i^{f} is the specific heat capacity of crude oil of ith unit, F_i^{h} is the flow rate of heating medium of ith unit, c_i^{h} is the specific heat capacity of heating medium of ith unit, h_{it} is the heat transfer co-efficient of ith unit at tth time instant and A_i is the area of ith unit.

The heat transfer co-coefficients for cleaning/shutdown sub-period (h_{it}^{cl}) and processing sub-period (h_{it}^{pr}) can be obtained from the linear fouling rates ($\dot{R}_t^{\mathrm{f}}$), which are expressed by Eq. (5).

$$\dot{R}_{it}^{\mathrm{f,pr}} = \dot{R}_{it}^{\mathrm{f,cl}} = \dot{R}_t^{\mathrm{f}} \tag{5a}$$

$$h_{it}^{\mathrm{cl}} = \frac{h_{i,t-1}^{\mathrm{pr}}}{1 + \{h_{i,t-1}^{\mathrm{pr}} \dot{R}_{i,t-1}^{\mathrm{f,pr}} (1-\alpha_{it}) \Delta t\}} \tag{5b}$$

$$h_{it}^{\mathrm{pr}} = \frac{h_{it}^{\mathrm{cl}}}{1 + (h_{it}^{\mathrm{cl}} \dot{R}_{i,t-1}^{\mathrm{f,cl}} \beta_{it} \Delta t)} + (v_{it} h_{it}^{\mathrm{clean}}) \tag{5c}$$

$$\tag{5d}$$

In Eq. (5), $\dot{R}_{it}^{\mathrm{f,pr}}$ is the fouling resistance under processing sub-period, $\dot{R}_{it}^{\mathrm{f,cl}}$ is the fouling resistance under cleaning sub-period and Δt is the duration of each time interval.

4 Evolutionary algorithms (EAs) for solving the problem

The optimization problem studied in the present work seeks the scheduling of the crude oil preheating process in a CPT of N heat exchangers over T time instants, so as to minimize the total cost arising from the requirement of external

energy for additional heating of the crude oil and periodic cleaning of the heat exchangers. The scheduling of the heat exchangers needs $2NT$ number of {0,1} binary variables (u_{it} and v_{it}; $i = 1, \ldots, N$ and $t = 1, \ldots, T$), while the crude preheating process requires NT number of real variables (α_{it}; $i = 1, \ldots, N$ and $t = 1, \ldots, T$). For solving the problem, three mixed-binary EAs are investigated here, which are genetic algorithm (GA), differential evolution (DE) and particle swarm optimization (PSO). In all the three EAs, an individual (solution representation) for the problem at hand consists of two one-dimensional arrays, the first one of size $2NT$ takes the {0,1} binary variables and the second one of size NT takes the real variables.

The investigated binary-real coded GA (brGA) is the one applied by Datta [3] to a problem of similar nature, namely the unit commitment problem arising in the area of power systems, which involves the scheduling of given power generating units and optimization of discharge from the operational units in a way to meet the hourly power demand at a minimum production cost subject to a series of system related fixed and dynamic constraints. In the brGA, the standard binary tournament selection operator, single-point crossover operator and swapping mutation operators are used for handling the {0,1} valued binary variables; while the binary tournament selection operator, simulated binary crossover (SBX) operator [11] and polynomial mutation operator [11] are used for handling the real variables of a problem.

Datta and Figueira [12] proposed a real-integer-discrete coded differential evolution (ridDE) algorithm for working with any type of variables (real, binary, integer, or discrete) without any conversion, which was also applied successfully to the unit commitment problem by Datta and Dutta [2]. The ridDE replaces the real valued mutation operator of the 'DE/rand/1/bin' variant of DE [13] by a binary valued mutation operator, which generates only {0,1} valued binary mutant elements with a mutation probability based on some basic properties of DE and such binary numbers. The ridDE is investigated here as another EA for solving the problem at hand.

Similar to the ridDE [12], Datta and Figueira [14] proposed a real-integer-discrete coded particle swarm optimization (ridPSO) algorithm for working with any type of variables (real, binary, integer, or discrete) without any conversion, whose application was demonstrated on various engineering design problems. The ridPSO defines particle vectors by {0,1} valued binary elements with a mutation probability, based on some basic properties of PSO and such binary numbers. The ridPSO is investigated here as the third EA for solving the crude oil preheating problem.

Since all the three EAs are stochastic in nature, there is no guarantee that the new individuals formed in a generation (iteration) will be better than those of the current individuals from where they were generated. Hence, in order to prevent the search from moving opposite to the optimum in worst cases, the elite individuals at every generation are preserved using the mechanism proposed by Deb et al. [15]. In this case, instead of forming the population for the next generation directly with the newly generated individuals, they are first combined

with the existing individuals of the current population. Then the best 50% of them, based on their objective values, are taken as the population for the next generation.

5 Numerical experimentation

The EAs stated in Section 4 are coded in the `C` programming language by incorporating the optimization problem formulated in Eqs. (1) and (2). Then the performances of the EAs are evaluated with the help of a case study.

5.1 Case study

The investigated case study of crude oil preheating is taken from Smaili et al.[10]. As shown in Fig. 1, the CPT in the case study consists of 14 number of shell and tube heat exchangers ($N = 14$) of the type of counter-current flow. The heat exchanger network (HEN) starts with two series segments (ns = 2); the first one contains units (heat exchangers) 1–5, followed by a desalter, and then the second series segment containing units 6–8. Fixing a flash after the second series segment, the remaining six units are then arranged in a parallel segment (npl = 1) having two branches; the first one contains units 9–11 and the second one contains units 12–14. At the end of the HEN, a furnace is placed for further heating of the crude oil, if required.

There are seven heating lines (nhl = 7) in the HEN, which are marked in Fig. 1 as H1–H7. The units (heat exchangers) covered by the heating lines are as follows — H1: (1, 9, 12), H2: (8, 10, 13), H3: (6, 11, 14), H4: (3, 7), H5: (2), H6: (4) and H7: (5).

The case study is subjected to some operational constraints in the form of minimum number of units to be made always fully operational. Each of the two series segments and the two branches of the parallel segment requires minimum of two of its units to be made fully operational. Some heating lines also have similar requirement, which are as follows — H1: 2, H2: 2, H3: 2 and H4: 1.

For solving the problem, a repeating production cycle of 3 years is considered, which is divided into 36 time instants ($T = 36$), i.e., each time instant is of a duration of one month. Except the cost coefficients, the problem related other input parameters are taken from Smaili et al. [10] and given in Table 1 in terms of the notations used in the problem formulation in Eqs. (1)–(5). The cleaning cost coefficient (C^{cl}) and energy cost coefficient (C^{ener}) are taken from Tian et al. [16], which are 20000 \$ per cleaning instant and 15.5 \$ per MWh, respectively. Further, the initial temperature of the crude oil at any time instant is considered to be 26 °C ($\Theta_t^{\mathrm{finit}} = 26\,^{\circ}\mathrm{C}$), requiring it to be preheated up to 250 °C ($\Theta^{\mathrm{fmax}} = 250\,^{\circ}\mathrm{C}$) with a drop of 10 °C in the desalter ($\Theta^{\mathrm{desalter}} = 10\,^{\circ}\mathrm{C}$).

5.2 Experimental setup

The considered EA related parameter values are given in Table 2, where a non-applicable value is marked by (–). Since the performance of a stochastic optimizer

Table 1. Design and fouling data for the case study (source: Smaili et al. [10])

Unit	Θ_i^{hinit} (°C)	F_i^{h} (kg/s)	F_i^{f} (kg/s)	c_i^{h} (kJ/kgK)	c_i^{f} (kJ/kgK)	h_{it}^{clean} (W/m^{2}K)	A_i (m^2)	$R_t^{\text{f}} \times 10^{-7}$ (m^{2}K/J)
HE-1	194	19.1	95	2.8	1.92	0.5	56.6	0.6
HE-2	296	3.3	95	2.9	1.92	0.5	8.9	0.9
HE-3	197	55.8	95	2.6	1.92	0.5	208.3	0.6
HE-4	170	49.7	95	2.6	1.92	0.5	112.9	0.8
HE-5	237	49.7	95	2.6	1.92	0.5	121.6	0.8
HE-6	285	34.8	95	2.8	2.3	0.5	110.1	1.5
HE-7	205	55.8	95	2.6	2.3	0.5	67.2	1.1
HE-8	254	45.5	95	2.9	2.3	0.5	67.1	1.5
HE-9	249	9.5	46	2.8	2.4	0.5	91.0	1.6
HE-10	286	22.8	46	2.9	2.4	0.5	61.3	1.8
HE-11	334	17.4	46	2.8	2.4	0.5	55.6	1.9
HE-12	249	9.5	46	2.8	2.4	0.5	91.0	1.6
HE-13	286	22.8	46	2.9	2.4	0.5	61.3	1.8
HE-14	334	17.4	46	2.8	2.4	0.5	55.6	1.9

Table 2. User-defined parameter values for the investigated EAs.

Parameter	brGA	ridDE	ridPSO
Population size	100	100	100
Maximum number of generations performed	7000	7000	7000
Crossover probability	90%	(0,90%]	–
Distribution index for SBX operator	20	–	–
Mutation probability	(0,1%]	–	–
Distribution index for polynomial mutation operator	35	–	–
Mutation probability (for binary variables only)	–	(0,15%]	(0,15%]
Scaling factor (for real variables only)	–	(0,70%]	–
Inertia constant (for real variables only)	–	–	(0,0.75]
Cognitive factor (for real variables only)	–	–	(0,1.5]
Social factor (for real variables only)	–	–	(0,2]
Number of runs	30	30	30

is likely to be influenced by the user-defined algorithmic parameter setting, instead of fixed values, some parameter values in Table 2 are made self-adaptive within given ranges with an attempt to reduce their influences on the performance of an EA. In this process, every time a random value for such a parameter is generated within its given range. Further, in order to analyze the statistical performance, 30 number of independent runs of each EA are performed with different sets of initial individuals (solutions).

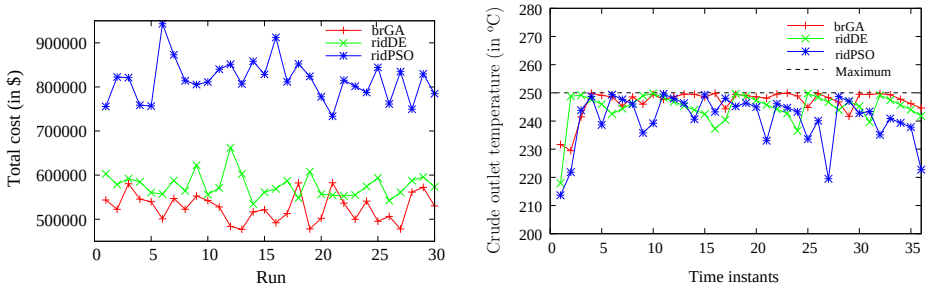
5.3 Results and discussion

With the above problem and algorithm related input information, each of the EAs are executed for 30 independent runs. For the purpose of illustration, the best schedule obtained by the brGA is given in Table 3, where '1' in the schedule means that the particular unit (heat exchanger) is in fully operation at the corresponding time instant, while '0' means it was partially shutdown during which period the unit may go through cleaning also. The schedule shows that no two units are cleaned at two consecutive time instants and each unit is cleaned at least once in the entire production cycle.

The overall costs, i.e., the values of the objective function expressed by Eq. (1), obtained from 30 runs of each of the EAs are visualized in Fig. 2(a),

Table 3. The best schedule of the case study obtained by the brGA.

Time instant	Schedule	Number of operating units	Time instant	Schedule	Number of operating units
1	01111101111111	12	19	11111110111111	13
2	11111111111110	13	20	11111111111011	13
3	11111111111101	13	21	11110111011111	12
4	11101111011111	12	22	11111110110111	12
5	01111111110111	12	23	11111101111111	13
6	11111111111111	14	24	11111111111111	14
7	11011111101111	12	25	11111011011111	12
8	11111111101111	13	26	11111111111111	14
9	11110111011111	12	27	11111111011111	13
10	11111111101111	13	28	11111111111111	14
11	11111110011111	12	29	11111111111110	13
12	11111011101111	12	30	11110111111111	13
13	11011111101111	12	31	11111111101011	12
14	11101111111011	12	32	11101111111111	13
15	11111110111111	13	33	11111111110111	13
16	11111111101111	13	34	11111111111111	14
17	11111111111101	13	35	11111111111111	14
18	11110111101111	12	36	10111111101111	12



(a) Total cost. (b) Crude oil outlet temperature.

Fig. 2. Total cost over 30 runs and crude outlet temperatures over different time instants of a random run.

where it is observed that the lowest cost could be obtained by the brGA among the three EAs. Further, the obtained crude oil outlet temperatures from the heat exchanger network at different time instants of a random run are shown in Fig. 2(b), where the outlet temperatures obtained from the brGA are found to be almost close to the required maximum furnace temperature, while those obtained from the ridPSO are found to be the worst ones.

For further detail of the performances of the EAs, a statistical analysis of the overall costs (best, worst, mean, and standard deviation) over 30 independent runs of the EAs is performed and the obtained results are presented in Table 4. It is seen in Table 4 that the brGA has better objective values (best, worst as well as mean), followed by those of the ridDE. However, the ridDE has better standard deviation than those of the brGA and ridPSO. Therefore, finally the EAs are statistically compared by conducting pair-wise t-test between the mean objective value and standard deviation at a significance level of 5%. The obtained

Table 4. Statistical analysis of the overall cost over 30 independent runs of the EAs.

EA	Overall cost (in $)			
	Best	Worst	Mean	Standard deviation
brGA	477150	582772	526507	31555
ridDE	534629	662064	576553	26633
ridPSO	733464	912245	815510	46860

t values are given in Table 5, by marking a value with a '-ve' sign if the second EA in a pair is not better than the first one. Accordingly, it can be concluded

Table 5. The t-test values for the solutions of the EAs at a significance level of 5%.

EA	brGA-vs-ridDE	brGA-vs-ridPSO	ridDE-vs-ridPSO
t-value	-6.64	-28.02	-24.28

that the brGA outperforms the ridDE and ridPSO, and the ridDE outperforms the ridPSO.

6 Conclusion

A typical crude oil preheating process arising in refineries is formulated as a constrained mixed-integer nonlinear programming (MINLP) problem for minimizing total of the cost of additional energy requirement and the cost for cleaning the heat exchangers of the process. It involves two separate optimization subproblems, the integer valued scheduling of the heat exchangers and the real valued heating levels in the operational heat exchangers. Such problems are usually handled by two separate algorithms, one for the integer part and another for the real part. The potentiality of thee mixed-binary evolutionary algorithms (EAs), namely genetic algorithm (GA), differential evolution (DE) and particle swarm optimization (PSO), are investigated here for handling both the parts of the problem by a single EA. From statistical analysis of the results for a benchmark problem, the GA is found outperforming both the DE and PSO, followed by the DE outperforming the PSO.

References

1. Chandrasekaran, K., Simon, S.: Multi-objective unit commitment problem using Cuckoo search Lagrangian method. International Journal of Engineering, Science and Technology **4** (2012) 89–105
2. Datta, D., Dutta, S.: A binary-real-coded differential evolution for unit commitment problem. Electrical Power and Energy Systems **42** (2012) 517–524
3. Datta, D.: Unit commitment problem with ramp rate constraint using a binary-real-coded genetic algorithm. Applied Soft Computing **13** (2013) 3873–3883

4. Deka, D., Datta, D.: Multi-objective optimization of the scheduling of a heat exchanger network under milk fouling. Knowledge-Based Systems **121** (2017) 71–82
5. Trivedi, A., Srinivasan, D., Biswas, S., Reindl, T.: A genetic algorithm differential evolution based hybrid framework: Case study on unit commitment scheduling problem. Information Sciences **354** (2016) 275–300
6. Juang, C.F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE transactions on systems, man, and cybernetics-part B: cybernetics **34** (2004) 997–1006
7. Kuo, R.J., Syu, Y.J., Chen, Z.Y., Tien, F.C.: Integration of particle swarm optimization and genetic algorithm for dynamic clustering. Information Sciences **195** (2012) 124–140
8. Ozturk, C., Hancer, E., Karaboga, D.: A novel binary artificial bee colony algorithm based on genetic operators. Information Sciences **297** (2015) 154–170
9. Epitropakis, M.G., Plagianakos, V.P., Vrahatis, M.N.: Evolving cognitive and social experience in Particle Swarm Optimization through Differential Evolution: A hybrid approach. Information Sciences **216** (2012) 50–92
10. Smaili, F., Vassiliadis, V.S., Wilson, D.I.: Mitigation of fouling in refinery heat exchanger networks by optimal management of cleaning. Energy and Fuels **15** (2001) 1038–1056
11. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons Ltd, Chichester, England (2001)
12. Datta, D., Figueira, J.R.: A real-integer-discrete-coded differential evolution. Applied Soft Computing **13** (2013) 3884–3893
13. Storn, R., Price, K.: Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11** (1997) 341–354
14. Datta, D., Figueira, J.R.: A real-integer-discrete-coded particle swarm optimization for design problems. Applied Soft Computing **11** (2011) 3625–3633
15. Deb, K., Agarwal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
16. Tian, J., Wang, Y., Feng, X.: Simultaneous optimization of flow velocity and cleaning schedule for mitigating fouling in refinery heat exchanger networks. Energy **109** (2016) 1118–1129

Robustness of Metaheuristic Algorithms in Optimum Design of Reinforced Concrete Beams

Gebrail Bekdaş[1] and Sinan Melih Nigdeli[1]

Istanbul University, Civil Engineering Department Avcılar Istanbul 34320, Turkey,
bekdas@istanbul.edu.tr, melihnig@istanbul.edu.tr

Abstract. In this study, RC beams under flexural effects are optimized by using new generation metaheuristic algorithms. The selected algorithms are Flower pollination algorithm (FPA), Teaching learning based optimization (TLBO) and Jaya algorithm (JA), which are quite new ones. The methodologies employing these algorithms are used in the numerical examples for different runs in order to evaluate the robustness of the algorithms. All algorithms are effective on the problem, but new methods may be developed to improve the robustness of the optimization.

Keywords: RC beams, Optimization, Flower pollination algorithm (FPA), Teaching learning based optimization (TLBO), Jaya algorithm (JA)

1 Introduction

The cost optimization of reinforced concrete (RC) structural members is one of the major objectives of civil engineers and it is a challenging design since concrete and reinforcement steel has different behavior in stress and prices. In that case, the best design ensuring the security measures with the minimum cost can be only found by trying different sizes of members. By using heuristic approaches, the amount of the trials is enhanced. Thus, global optimum results can be found. The optimum design of RC beams have been investigated by using metaheuristic based methods [1, 2], but these studies generally present a proposed without robustness evaluation. In this study, the algorithms such as Flower pollination algorithm (FPA) [3], Teaching learning based optimization (TLBO) [4], Jaya algorithm (JA) [5] are investigated for the robustness and efficiency.

2 The optimization problem

The design variables are breadth (b_w), height (h), number of reinforcements in different sections (n_1, n_2, n_3 and n_4) and size of the reinforcement (ϕ_1, ϕ_2, ϕ_3 and ϕ_4) as seen in Fig. 1. The objective is to minimize the total material cost by considering constraints according to American Concrete Institute (ACI318) [6]. The design constants defined as the clear cover of the reinforcement (35 mm),

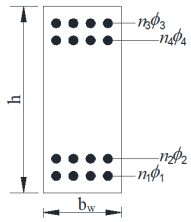


Fig. 1. The optimization problem with design variables

the maximum size of the aggregate diameter (16 mm), the specified compressive strength of concrete (20 MPa), the specified yield strength of reinforcement (420 MPa), the diameter of stirrup (10 mm), the cost of the concrete (40 \$/m^3) and reinforcement bars (400 \$/t). The discrete optimization was done for the ranges; 10-30 mm, 250-350 mm and 350-500 mm for diameters of the main reinforcement bars, breadth and height, respectively.

3 Discussion and Conclusion

The robustness evaluation and optimum results are presented in Table 1 for different moment capacities. According to the results, the best one is JA. It must be noted that all algorithms shows different performances for different flexural moment goals and it validates the no free-lunch theorem. The best solutions can be found if the results of the 20 runs are checked, but minor local optima problem is observed for the algorithms since the standard deviation values are not so small. The new generation algorithms are feasible for the problem, but hybrid and new methods are needed to improve the robustness of the optimization. In the future studies, the hybrid of these algorithms can be considered and additonal robustness measures can be investigated.

References

1. Koumousis, V. K., Arsenis, S. J. (1998), Genetic Algorithms in Optimal Detailed Design of Reinforced Concrete Members, Comput-Aided Civ. Inf., 13, 43-52.
2. Govindaraj, V., Ramasamy, J. V. (2005), Optimum detailed design of reinforced concrete continuous beams using Genetic Algorithms, Comput. Struct., 84, 3448.
3. Yang, X.-S. (2012), Flower pollination algorithm for global optimization. In unconventional computation and natural computation, 240-249.
4. Rao, R. V., Savsani, V. J. and Vakharia, D. P. (2011). Teachinglearning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Design, 43(3), 303-315.
5. Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7(1), 19-34.
6. American Concrete Institute (2008). Building code requirements for structural concrete (ACI 318-08) and commentary.

Table 1. The optimum results

					JA					
Moment (kNm)	50	100	150	200	250	300	350	400	450	500
h (mm)	350	400	500	500	500	500	500	500	500	500
b_w (mm)	250	250	250	250	250	300	300	350	350	400
ϕ_1 (mm)	14	16	16	28	26	22	26	26	28	26
ϕ_3 (mm)	20	28	30	22	14	12	16	12	16	12
n_1	2	3	4	2	3	5	4	5	5	6
n_3	0	0	0	0	2	2	3	6	5	9
ϕ_3 (mm)	12	10	12	10	12	14	12	10	10	12
ϕ_4 (mm)	18	16	14	16	18	22	18	26	14	30
n_2	2	4	2	3	2	2	4	3	2	4
n_4	0	0	0	0	0	0	0	0	0	0
Mu (kNm)	57.31	111.22	167.71	222.42	279.29	333.48	388.94	445.15	500.11	555.96
Best Cost ($/m)	5.16	6.85	8.20	9.55	11.60	13.56	15.87	18.08	20.16	22.45
Num. of analyses	225	100	75	100	125	200	1475	950	175	2100
Ave. Cost ($/m)	5.17	6.86	8.25	9.59	11.65	13.70	16.05	18.62	20.58	22.93
Standard Dev.	0.01	0.02	0.07	0.07	0.06	0.12	0.19	0.43	0.28	0.21
					TLBO					
Moment (kNm)	50	100	150	200	250	300	350	400	450	500
h (mm)	350	400	500	500	500	500	500	500	500	500
b_w (mm)	250	250	250	250	250	250	300	300	350	400
ϕ_1 (mm)	14	16	16	28	26	28	26	28	28	26
ϕ_3 (mm)	30	30	16	20	14	14	14	16	16	12
n_1	2	3	4	2	3	3	4	4	5	6
n_3	0	0	0	0	2	4	4	5	5	9
ϕ_3 (mm)	12	10	12	10	12	12	14	12	12	12
ϕ_4 (mm)	28	14	16	10	18	18	18	12	16	20
n_2	2	4	2	3	2	3	3	4	2	4
n_4	0	0	0	0	0	0	0	0	0	0
Mu (kNm)	57.31	111.22	167.71	222.42	279.29	333.50	390.51	445.40	506.81	555.96
Best Cost ($/m)	5.16	6.85	8.20	9.55	11.60	13.70	15.94	18.17	20.38	22.45
Num. of analyses	925	50	475	625	1125	100	1125	925	350	900
Ave. Cost ($/m)	5.16	6.85	8.23	9.56	11.71	13.87	16.15	18.43	20.68	22.97
Standard Dev.	0.01	0.01	0.04	0.01	0.06	0.10	0.12	0.19	0.26	0.22
					FPA					
Moment (kNm)	50	100	150	200	250	300	350	400	450	500
h (mm)	350	400	500	500	500	500	500	500	500	500
b_w (mm)	250	250	250	250	250	300	300	450	400	450
ϕ_1 (mm)	14	16	16	28	26	22	26	20	24	26
ϕ_3 (mm)	24	18	28	16	14	10	20	22	14	14
n_1	2	3	4	2	3	5	4	8	6	6
n_3	0	0	0	0	2	3	2	0	5	5
ϕ_2 (mm)	12	10	12	10	12	14	12	12	10	10
ϕ_4 (mm)	20	18	26	18	14	30	24	28	22	24
n_2	2	4	2	3	2	2	4	4	7	6
n_4	0	0	0	0	0	0	0	0	0	0
Mu (kNm)	57.31	111.22	167.71	222.42	279.29	334.02	389.55	445.09	500.11	557.23
Best Cost ($/m)	5.16	6.85	8.20	9.55	11.60	13.59	15.95	18.21	20.52	22.74
Num. of analyses	675	1225	1950	1450	450	1575	1400	1150	2025	2225
Ave. Cost ($/m)	5.30	6.87	8.25	9.56	11.78	13.95	16.30	18.60	20.84	23.07
Standard Dev.	0.13	0.02	0.05	0.01	0.11	0.15	0.19	0.22	0.23	0.21

Data-driven Preference-based Deep Statistical Ranking for Comparing Multi-Objective Optimization Algorithms

T. Eftimov[1,2], P. Korošec[1,3], and B. Koroušić Seljak[1]

[1] Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
[2] Jožef Stefan Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia
tome.eftimov@ijs.si
[3] Faculty of Mathematics, Natural Sciences and Information Technologies, Glagoljaška ulica 8, 6000 Koper, Slovenia
peter.korosec@ijs.si, barbara.korousic@ijs.si

Abstract. To find the strengths and weaknesses of a new multi-objective optimization algorithm, we need to compare its performance with the performances of the state-of-the-art algorithms. Such a comparison involves a selection of a performance metric, a set of benchmark problems, and a statistical test to ensure that the results are statistical significant. There are also studies in which instead of using one performance metric, a comparison is made using a set of performance metrics. All these studies assume that all involved performance metrics are equal. In this paper, we introduce a data-driven preference-based approach that is a combination of multiple criteria decision analysis with deep statistical rankings. The approach ranks the algorithms for each benchmark problem using the preference (the influence) of each performance metric that is estimated using its entropy. Experimental results show that this approach achieved similar rankings to a previously proposed method, which is based on the idea of the majority vote, where all performance metrics are assumed equal. However, as it will be shown, this approach can give different rankings because it is based not only on the idea of counting wins, but also includes information about the influence of each performance metric.

Keywords: Multiple criteria decision analysis, multi-objective optimization, quality indicators, deep statistical ranking, statistical comparison, data-driven

1 Introduction

When working on a new optimization algorithm, a crucial task is to compare its performance with state-of-the-art algorithms [4]. In single-objective optimization, the performance of algorithms is analyzed using the best algorithmic solution. For example, in the case of minimization problems, the solution with the

lowest value is the best. However, in multi-objective optimization algorithms (MOAs), it is not clear what the quality of a solution means in the presence of several optimization criteria. This is because the result is an approximation of the *Pareto-optimal* front, called an approximation set, which can be analyzed according to different quality aspects related to properties of convergence and diversity e.g., the closeness to the optimal front, coverage of a wide range of diverse solutions [3]. Quality indicators can be used to evaluate the performance of MOAs. Each quality indicator maps an approximation set to a real number [13]. In comparative studies, algorithms are used to solve a number of benchmark problems followed by the application of quality indicators to assess their performance [4]. Meta-heuristics are non-deterministic techniques, meaning there is no guarantee that the result will be the same for every run. To test the quality of an algorithm, it is not enough to perform just one run, but many runs of the algorithm on the same problem are needed, from which conclusions can be drawn. Additionally, this data must be analyzed with some statistical tests to ensure that the results are significant.

The aim of this study is to compare the performance of MOAs using a data-driven preference-based approach with a set of quality indicators. In Section 2, an overview of the related works is presented. Section 3 introduces the data-driven preference-based methodology. In Section 4 the experimental study is presented, while Section 5 gives a discussion of the proposed methodology. The conclusions of the paper are presented in Section 6.

2 Related work

Many studies that address the problem of how to compare approximation sets in a quantitative manner have been conducted. Riquelme et al. [13] presented a study of a large number of metrics for comparing the performance of different multi-objective optimization algorithms, and presented a review and an analysis of 54 multi-objective optimization metrics and a discussion about the advantages/disadvantages of the most citied metrics in order to give researchers sufficient information for choosing them. A lot of the presented metrics use quality indicators to evaluate the quality of the solutions. Additionally, after calculating the quality indicator of interest, the data must be analyzed using a statistical test to ensure that the results are significant [8, 9]. In [7], Eftimov et al. presented a study on how to compare the performance of MOAs using quality indicators and a Deep Statistical Comparison (DSC) approach. They used the DSC approach because it gives more robust statistical results to compare MOAs regarding the data obtained for a single quality indicator. However, there are also studies that use more than one quality indicator to evaluate the performance of MOAs. In [15], Yen and He presented a double-elimination tournament using a quality indicator ensemble to rank MOAs. The tournament contains approximation sets obtained from MOAs for the same initial population and involves a series of binary tournament selections and in each one a quality indicator from an ensemble is randomly chosen for comparison. The result of the tournament is one

winning approximation set, so the corresponding MOA is ranked one. Then the approximations sets that are generated by the winning MOA are removed and the remaining approximation sets will go through another double elimination tournament to identify the second best algorithm and so on. The results of the evaluation show that the method is performing more or less as a majority vote. The same idea was used by Ravber et al. [12], where instead of double elimination tournament, they used the chess rating system based on the Glicko-2 system [10]. The comparison between two approximation sets was made by a randomly selected quality indicator from the ensemble. In both approaches, the selection of the quality indicator that is used for a binary tournament is random and comes from a uniform distribution, such that all quality indicators in the ensemble are equal. Eftimov et al., also presented a comparative study of MOAs using an ensemble of quality indicators together with DSC [6]. This study used two ensemble combiners to rank and compare MOAs. Using one of them, each algorithm obtains a ranking for each problem, which is the average of its DSC rankings for each quality indicator for that problem. The other proposed ensemble is a hierarchical majority vote, which is a recursive approach where each algorithm is checked for the number of wins. In both scenarios, there is no preference between the quality indicators used in the comparison and all are assumed equal.

2.1 The Deep Statistical Ranking

Deep Statistical Comparison (*DSC*) is a recently proposed approach for making a statistical comparison of meta-heuristic stochastic optimization algorithms on a set of single-objective problems [8]. Its main contribution is its ranking scheme, which is based on the whole distribution instead of using just one statistic to describe the distribution, such as either the average or the median. A study on how to compare the performance of MOAs using quality indicators and DSC can be found in [6, 7], where DSC gave more robust results compared to a standard statistical test recommended for making a statistical comparison.

2.2 The PROMETHEE

PROMETHEE methods are used in decision making to solve a decision problem in which a set of alternatives are evaluated according to a set of criteria that are often conflicting. Without loss of generality, we can assume that these criteria have to be minimized. For the method, an evaluation matrix is constructed, in which each alternative is estimated for each criteria. The method performs pairwise comparisons between all the alternatives for each criteria to provide either a complete or partial rankings of the alternatives. Four PROMETHEE methods exist, named as I, II, III, and IV. They can be used depending on the nature of the data that is involved in the comparison and the type of ranking that is preferred.

3 The proposed methodology

The proposed methodology consists of two steps. In the first, the DSC ranking scheme is used to obtain robust statistics regarding each quality indicator separately, which are combined in the second step using the PROMETHEE II method [2].

3.1 The PROMETHEE II

Let us assume that a comparison needs to be made between m algorithms (i.e., alternatives) regarding n quality indicators (i.e., criteria) for a single problem. Let $A = \{A_1, A_2, \ldots, A_m\}$ be the set of algorithms we want to compare regarding the set of quality indicators $Q = \{q_1, q_2, \ldots, q_n\}$. The decision matrix is a $m \times n$ matrix (see Table 1) that contains the DSC rankings obtained for the algorithms for each quality indicator separately.

Table 1. Decision matrix.

	q_1	q_2	$\ldots$	q_n
A_1	$q_1(A_1)$	$q_2(A_1)$	$\ldots$	$q_n(A_1)$
A_2	$q_1(A_2)$	$q_2(A_2)$	$\ldots$	$q_n(A_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
A_m	$q_1(A_m)$	$q_2(A_m)$	$\ldots$	$q_n(A_m)$

The DSC ranking scheme always ranks the best algorithm as one, the second best as two, and so on. In our case, we are interested in minimizing the criteria since lower DSC ranking values are preferable. Before we start with the PROMETHEE, the decision matrix is transformed in such a way that the DSC rankings, which are in the same column, are transformed using a standard competition ranking scheme [6]. This should be done because for the DSC rankings it does not matter if rankings are 1.50, 3.00, and 1.50 or 1.00, 3.00, and 1.00. In both scenarios having 1.00 and 1.50 means that the algorithm is the best according to some quality indicator. Since the DSC ranking scheme can never give a 1.00, 3.00, and 1.00 when comparing three algorithms (since it follows the idea of fractional ranking), the DSC rankings for each quality indicator are transformed using the standard competition ranking scheme.

The appropriate method in our case is PROMETHEE II. It is based on pairwise comparisons that need to be made between all algorithms for each quality indicator. The differences between DSC rankings for each pair of algorithms according to a specified quality indicator are taken into consideration. For larger differences the decision maker might consider larger preferences. The preference function of a quality indicator for two algorithms is defined as the degree of

preference of algorithm A_1 over algorithm A_2 as seen in the following equation:

$$P_j(A_1, A_2) = \begin{cases} p_j(d_j(A_1, A_2)), & if\ maximizing\ the\ quality\ indicator \\ p_j(-d_j(A_1, A_2)), & if\ minimizing\ the\ quality\ indicator \end{cases}, \quad (1)$$

where $d_j(A_1, A_2) = q_j(A_1) - q_j(A_2)$ is the difference between the DSC rankings of the algorithms for the quality indicator q_j and $p_j(\cdot)$ is a generalized preference function assigned to the quality indicator. There exist six types of generalized preference functions [2]. In our case, usual preference function is used for each quality indicator because of the importance of any differences between the rankings, which is presented in Equation 2.

$$p(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}, \quad (2)$$

After selecting the preference function for each quality indicator, the next step is to define the average preference index and outranking (preference and net) flows. The average preference index for each pair of algorithms gives information of global comparison between them using all quality indicators. The average preference index can be calculated as:

$$\pi(A_1, A_2) = \frac{1}{n} \sum_{j=1}^{n} w_j P_j(A_1, A_2), \quad (3)$$

where w_j represents the relative significance (weight) of the j^{th} quality indicator. The higher the weight value of a given quality indicator the higher its relative significance. The selection of the weights is a crucial step in the PROMETHEE II method because it defines the priorities used by the decision-maker. In our case, we used the Shannon entropy weight method, which will be explained in the next subsection. For the average preference index, we need to point out that it is not a symmetric function, so $\pi(A_1, A_2) \neq \pi(A_2, A_1)$.

To rank the algorithms, the net flow for each algorithm needs to be calculated. It is the difference between the positive preference flow, $\phi(A_i^+)$, and the negative preference flow of the algorithm, $\phi(A_i^-)$. The positive preference flow gives information how a given algorithm is globally better than the other algorithms, while the negative preference flow gives the information about how a given algorithm is outranked by all the other algorithms. The positive and the negative preference flows are defined as:

$$\phi(A_i^+) = \frac{1}{(n-1)} \sum_{x \in A} \pi(A_i, x),$$
$$\phi(A_i^-) = \frac{1}{(n-1)} \sum_{x \in A} \pi(x, A_i). \quad (4)$$

The net flow of an algorithm is defined as:

$$\phi(A_i) = \phi(A_i^+) - \phi(A_i^-). \quad (5)$$

The PROMETHEE II method ranks the algorithms by ordering them according to decreasing values of net flows.

3.2 The Shannon entropy weighted method

To find the quality indicator weights, we use the Shannon entropy weighted method [1]. For this reason, the decision matrix presented in Table 1 needs to be normalized. Because the smaller value is preferred, the matrix is normalized using the following equation:

$$q_j(A_i)^{'} = \frac{\max_i(q_j(A_i)) - q_j(A_i)}{\max_i(q_j(A_i)) - \min_i(q_j(A_i))}, \tag{6}$$

where $q_j(A_i)^{'}$ is the normalized value for $q_j(A_i)$.

The entropy for each quality indicator is defined as:

$$e_j = K \sum_{i=1}^{m} W\left(\frac{q_j(A_i)^{'}}{D_j}\right), \tag{7}$$

where D_j is the sum of the j^{th} quality indicator in all algorithms, $D_j = \sum_{i=1}^{m} q_j(A_i)^{'}$, K is the normalized coefficient, $K = \frac{1}{(e^{0.5}-1)m}$, and W is a function defined as $W(x) = xe^{(1-x)} + (1-x)e^x - 1$.

The weight of each quality indicator used in Equation 3 is calculated using the following equation:

$$w_j = \frac{\frac{1}{(n-E)}(1-e_j)}{\sum_{j=1}^{n}\left[\frac{1}{(n-E)}(1-e_j)\right]}, \tag{8}$$

where E is the sum of entropies, $E = \sum_{j=1}^{n} e_j$.

4 Results

4.1 Experimental setup

The data from six algorithms is available from [14]. The algorithms are compared using 16 test problems. The number of objectives is set to four. More about the parameters of the test problems and the algorithms can be found in [14]. All test problems assume minimization of all objectives. Each algorithm was run for each problem 30 times. Before calculating the quality indicators, each approximated *Pareto* front was normalized. In our experiment quality indicators are hypervolume (q_1), epsilon indicator (q_2), r_2 indicator (q_3), and generational distance (q_4). All of them are unary indicators. Since we are introducing a methodology, we are not specifically dealing which quality indicators are used. The selection is up to user to make sure that relevant quality indicators are selected (e.g., if all quality indicators should be Pareto compliant, convergence, diversity, etc.). For

calculating the hypervolume, the reference point (1,. . . , 1) is used, while for the other quality indicators, the reference set consists of all non-dominated solutions already known from all runs for each algorithm for a given problem. Because the DSC ranking scheme involves a statistical test for comparing distributions, a two-sample *Anderson-Darling (AD)* test is used and the significance level is set to 0.05. The benefits of using this test are presented in [5].

4.2 Experimental results

In the experiment, three out of six algorithms are randomly selected. The algorithms are: DEMO^{SP2}, DEMO^{NS-II}, and NSGA-II. First, for each quality indicator, the DSC ranking scheme is used to compare the quality indicator data for a single problem. Further, the DSC rankings obtained for each quality indicator and each problem are transformed using the standard competition ranking scheme (see Table 2). The highest ranked algorithm for each problem and each quality indicator has the best performance.

Table 2. Transformed DSC rankings for each quality indicator of the algorithms, A_1=DEMO^{SP2}, A_2=DEMO^{NS-II}, and A_3=NSGA-II.

Problem	Hypervolume			r_2			Epsilon			Generational distance		
	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3
DTLZ1	2.00	1.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00
DTLZ2	2.00	1.00	3.00	3.00	1.00	2.00	2.00	1.00	3.00	2.00	1.00	3.00
DTLZ3	1.00	1.00	3.00	2.00	1.00	3.00	1.00	1.00	3.00	1.00	1.00	3.00
DTLZ4	1.00	2.00	3.00	1.00	2.00	2.00	1.00	2.00	3.00	1.00	2.00	3.00
DTLZ5	2.00	2.00	1.00	1.00	1.00	3.00	1.00	1.00	1.00	1.00	3.00	2.00
DTLZ6	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00	1.00	2.00	3.00
DTLZ7	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
WFG1	1.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00	1.00	3.00	2.00
WFG2	1.00	2.00	3.00	1.00	2.00	2.00	1.00	2.00	2.00	1.00	3.00	1.00
WFG3	1.00	3.00	2.00	1.00	2.00	2.00	1.00	2.00	2.00	1.00	2.00	2.00
WFG4	1.00	2.00	3.00	2.00	1.00	2.00	2.00	1.00	3.00	3.00	2.00	1.00
WFG5	3.00	2.00	1.00	3.00	1.00	1.00	1.00	3.00	2.00	3.00	2.00	1.00
WFG6	1.00	2.00	3.00	2.00	1.00	3.00	1.00	2.00	2.00	3.00	1.00	1.00
WFG7	1.00	2.00	3.00	2.00	1.00	3.00	1.00	2.00	2.00	3.00	2.00	1.00
WFG8	1.00	2.00	2.00	1.00	2.00	3.00	1.00	2.00	2.00	1.00	3.00	2.00
WFG9	1.00	2.00	2.00	1.00	1.00	3.00	1.00	2.00	2.00	3.00	2.00	1.00

Before we find the complete ranking of the algorithms, the weights of each quality indicator are calculated for each single problem using the Shannon entropy weighted method. The weights for all problems are presented in Table 3.

Then, the PROMETHEE II method is used to rank the algorithms for each problem. If the original decision matrix is involved in the PROMETHEE II calculations, the preference function that is used is the one for minimizing the quality indicator, while if the normalized matrix is used, the preference function is the one used to maximize the quality indicator. In our case, we have a set of three algorithms $A = \{A_1, A_2, A_3\}$ that need to be compared according to

Table 3. Weights for each quality indicator.

Problem	q_1	q_2	q_3	q_4	Problem	q_1	q_2	q_3	q_4
DTLZ1	0.25	0.25	0.25	0.25	WFG2	0.14	0.37	0.37	0.12
DTLZ2	0.25	0.25	0.25	0.25	WFG3	0.13	0.29	0.29	0.29
DTLZ3	0.24	0.28	0.24	0.24	WFG4	0.18	0.46	0.18	0.18
DTLZ4	0.18	0.46	0.18	0.18	WFG5	0.26	0.22	0.26	0.26
DTLZ5	0.57	0.20	0.00	0.23	WFG6	0.19	0.19	0.47	0.15
DTLZ6	0.25	0.25	0.25	0.25	WFG7	0.18	0.18	0.46	0.18
DTLZ7	0.25	0.25	0.25	0.25	WFG8	0.36	0.14	0.36	0.14
WFG1	0.25	0.25	0.25	0.25	WFG9	0.37	0.12	0.37	0.14

a set of four quality indicators $Q = \{q_1, q_2, q_3, q_4\}$. The rankings obtained for PROMETHEE II method are presented on the left side of Table 4. They are further compared with the rankings obtained by the average ensemble with the DSC rankings (DSC ensemble I) [6], presented in the middle part of Table 4 and the hierarchical majority vote with the DSC rankings (DSC ensemble II) [6], presented on the right side of Table 4. From it, we can see that the rankings obtained using PROMETHEE II with DSC differ from the rankings obtained using the average ensemble with DSC or the hierarchical majority vote with DSC only in two bolded problems: DTLZ5 and WFG7.

Table 4. Ensemble combiner for the algorithms: A_1=DEMO$^{\mathrm{SP2}}$, A_2=DEMO$^{\mathrm{NS-II}}$, and A_3=NSGA-II.

Problem	PROMETHEE II			DSC ensemble I			DSC ensemble II		
	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3
DTLZ1	1.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00
DTLZ2	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
DTLZ3	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
DTLZ4	1.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00
DTLZ5	**2.00**	**3.00**	**1.00**	**1.00**	**2.50**	**2.50**	**1.00**	**2.50**	**2.50**
DTLZ6	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
DTLZ7	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
WFG1	1.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00
WFG2	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00	2.00
WFG3	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00	2.00
WFG4	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
WFG5	3.00	2.00	1.00	3.00	2.00	1.00	3.00	2.00	1.00
WFG6	1.00	2.00	3.00	2.00	1.00	3.00	2.00	1.00	3.00
WFG7	**1.00**	**2.00**	**3.00**	**1.50**	**1.50**	**3.00**	**1.00**	**2.00**	**3.00**
WFG8	1.00	2.50	2.50	1.00	2.50	2.50	1.00	2.50	2.50
WFG9	1.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00

To see what happens on a single problem, let us focus on the DLTZ5 problem. The decision matrix and its normalization are presented at top of Table 5. The transformed DSC rankings for the r_2 indicator and the DLTZ5 problem are 1.00, 1.00, and 1.00. Further, there is a problem in the normalization process because the normalized rankings are indeterminate forms (i.e., 0/0) [11], so the

weight or the relative significance of this quality indicator can not be calculated. However, according to this quality indicator and the obtained DSC rankings, the compared algorithms are the same and they are all winners. Let us suppose that the weight w_3 could be calculated in some way, then the part of the average preference index that is related to the q_3 indicator is a product of $w_3 P_3(A_{i_1}, A_{i_2})$, where $i_1, i_2 = 1, \ldots, m$ and $i_1 \neq i_2$. In this case, it will be zero and will not influence the average preference index, which is used for calculating the positive and negative flows. Because it can not provide any additional information, it is removed and the result will be the same as comparing the algorithms regarding the remaining quality indicators, which in our case are q_1, q_2, and q_4. By removing the r_3 indicator, the decision matrix and its normalization are presented at the bottom part of Table 5. The weights obtained using the Shannon entropy weighted method are 0.57, 0.20, and 0.23. The final rankings and the outranking

Table 5. Decision matrices for DLTZ5.

Algorithm	Decision matrix				Normalized matrix			
	q_1	q_2	q_3	q_4	q_1	q_2	q_3	q_4
DEMO$^{\text{SP2}}$	2.00	1.00	1.00	1.00	0.00	1.00	0/0	1.00
DEMO$^{\text{NS}-\text{II}}$	2.00	1.00	1.00	3.00	0.00	1.00	0/0	0.00
NSGA-II	1.00	3.00	1.00	2.00	1.00	0.00	0/0	0.50
Algorithm	Decision matrix				Normalized matrix			
	q_1	q_2	q_3	q_4	q_1	q_2	q_3	q_4
DEMO$^{\text{SP2}}$	2.00	1.00	/	1.00	0.00	1.00	/	1.00
DEMO$^{\text{NS}-\text{II}}$	2.00	1.00	/	3.00	0.00	1.00	/	0.00
NSGA-II	1.00	3.00	/	2.00	1.00	0.00	/	0.50

flows are given on the left side od Table 6. On the right part of Table 6 the average preference indices that are used for calculating the positive and negative flows for DLTZ5 are presented.

Table 6. Outranking flows, PROMOTHEE II rankings, and average indicies for DLTZ5.

Algorithm	ϕ^+	ϕ^-	ϕ	Ranking		$\pi(A_i, A_1)$	$\pi(A_i, A_2)$	$\pi(A_i, A_3)$
DEMO$^{\text{SP2}}$	0.11	0.10	0.01	2.00	$\pi(A_1, A_j)$	0.00	**0.08**	**0.14**
DEMO$^{\text{NS}-\text{II}}$	0.03	0.17	-0.14	3.00	$\pi(A_2, A_j)$	**0.00**	0.00	**0.06**
NSGA-II	0.23	0.10	0.13	1.00	$\pi(A_3, A_j)$	**0.19**	**0.27**	0.00

Using the decision matrix presented in Table 5, the rankings obtained using the average ensemble and the hierarchical majority vote are the same and are 1.00, 2.50, and 2.50. In the case of hierarchical majority vote, DEMO$^{\text{SP2}}$ is ranked as first because it wins in three out of four quality indicators, while DEMO$^{\text{NS}-\text{II}}$ and NSGA-II are ranked second (e.g., 2.5) because both are ranked first in the case of two quality indicators, then both are second in the case of one quality indicator and third in the case of one quality indicator. All quality indicators are assumed equal and the ranking is made by counting the number of wins. However,

the obtained rankings using the data-driven preference-based approach are 2.00, 3.00, and 1.00, which are completely different from the other ensembles. From the left part of Table 6, we can see that NSGA-II has the highest positive flow. The question is why it is ranked first when DEMO^{SP2} has two wins. This happens because the quality indicators that are involved have a data-driven preference for each of them, which is obtained by the Shannon entropy weighted method. The quality indicators are ordered as q_1, q_4, q_2, (e.g, hypervolume, generational distance, and epsilon indicator), starting from the most significant one to the least significant one. The average preference indices between A_1 and A_3 that are used for calculating the positive and negative flows are:

$$\begin{aligned}\pi(A_1, A_3) &= \frac{1}{3}\left[0.57 \cdot 0 + \mathbf{0.20} \cdot \mathbf{1} + \mathbf{0.23} \cdot \mathbf{1}\right] = 0.14 \\ \pi(A_3, A_1) &= \frac{1}{3}\left[\mathbf{0.57} \cdot \mathbf{1} + 0.20 \cdot 0 + 0.23 \cdot 0\right] = 0.19\end{aligned} \tag{9}$$

Using the calculations presented in Equation 9, we can see that the average preference index between NSGA-II and DEMO^{SP2} is 0.19 and it is a result of only one win regarding the quality indicator q_1, while the average preference index between DEMO^{SP2} and NSGA-II is 0.14 and it is smaller even though it is a result of two wins regarding q_2 and q_4. This happens because q_1 is the most significant and its weight is much more than the sum of the weights of q_2 and q_4. In our experiment, the proposed data-driven preference-based approach gives different rankings from the hierarchical majority vote only for DLTZ5. This happens because only on that problem the compared algorithms are the same regarding one of the used quality indicators, which is the r_3 indicator. However, if this happens for other single-problems, the rankings can also differ from the rankings obtained by a hierarchical majority vote.

Furthermore, the obtained rankings using PROMETHEE II with DSC can be used as input data for a multiple-problem scenario. The appropriate statistical test is the *Friedman test*. Using it, the obtained p-value is 0.00, so using a significance level 0.05, we can conclude that there is a statistical significant difference between the compared algorithms using a set of benchmark problems. When comparing MOAs, often more than three algorithms are involved in the comparison, or especially a new algorithm is compared with state-of-the-art algorithm as a multiple comparisons with a control algorithm. When the number of algorithms increases the DSC rankings can be affected when correcting the p-values to control the FWER. In such a scenario, it is better to use multiple *Wilcoxon tests*, one for each pairwise comparison and then combine the p-values to find the actual p-value for the scenario. More about this scenario and the DSC approach is presented in [8]. If we are interested in to compare them using a data-driven preference-based approach, we just need to use PROMETHEE II with DSC instead of the original DSC ranking scheme to find the rankings for each pairwise comparison on each problem.

4.3 Discussion

Comparing the performance of a new MOA with the performance of state-of-the-art MOAs is a crucial task in order to find its strengths and weaknesses. Different performance metrics can be used for evaluation and they are usually combined with statistical tests to ensure that the results are significant. Several previously proposed approaches are focused on comparing MOAs using a set of quality indicators. They follow the idea of ensemble learning, but all of them assume that all quality indicators are equal. The performance metric and the way how the algorithms will be compared also depend on the user preference or the concrete application. For example, in our previous work, we presented an average ensemble and a hierarchical majority vote based on counting wins according to different quality indicators, but in this paper we proposed a data-driven preference-based approach that is a combination of PROMETHEE II and DSC ranking scheme. According to the user preference all involved quality indicators are still equal, but the data-driven preference changes this by using its entropy. Organizing the DSC rankings for each quality indicator and each problem into a decision matrix, the Shannon entropy weighted method is used to find the relative significance of each quality indicator for each problem. The relative significance of each quality indicator is related to its entropy, which is the amount of information conveyed by it. The experimental results have shown that the preference-based approach performs more or less as a hierarchical majority vote. However, it can give different rankings, and the algorithm can overrank another one even if it has a lower number of wins, but it wins in most preferred quality indicator(s). Also, if there is a quality indicator for which all compared algorithms perform the same (they all win), it does not have an influence in the comparison and it can be removed from the set of quality indicators. Comparing the hierarchical majority vote and data-driven preference-based ranking, we can say that the hierarchical majority vote is more appropriate in cases where the performance is estimated by counting wins and loses such as in the case of dynamic multi-objective optimization, otherwise data-driven preference-based ranking can be used in cases when the influence of each quality indicator is required.

5 Conclusion

In this paper, we presented a data-driven preference-based approach for comparing MOAs using a set of quality indicators. The approach is a combination of PROMETHEE II, which is a method in MCDA, and a DSC ranking scheme, that gives more robust statistical results and is based on comparing distributions instead of using only one statistic to describe the data. We compared our method with previously proposed methods where all involved quality indicators are assumed equal. We have shown that our method performs similar to a hierarchical majority vote, but also can give different rankings regarding the influence of each quality indicator, which is its preference and is estimated according to its entropy.

Acknowledgments. This work was supported by the project from the Slovenian Research Agency (research core funding No. P2-0098) and from the European Union's Horizon 2020 research and innovation program under grant agreement No. 692286.

References

1. Boroushaki, S.: Entropy-based weights for multicriteria spatial decision-making. Yearbook of the Association of Pacific Coast Geographers 79, 168–187 (2017)
2. Brans, J.P., Vincke, P.: Note - a preference ranking organisation method: (the promethee method for multiple criteria decision-making). Management science 31(6), 647–656 (1985)
3. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A., et al.: Evolutionary algorithms for solving multi-objective problems, vol. 5. Springer (2007)
4. Durillo, J.J., Nebro, A.J., Alba, E.: The jmetal framework for multi-objective optimization: Design and architecture. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. pp. 1–8. IEEE (2010)
5. Eftimov, T., Korošec, P., Seljak, B.K.: The behaviour of deep statistical comparison approach for different criteria of comparing distributions. In: Proceedings of 9th International Joint Conference on Computational Intelligence. SCITEPRESS Digital Library (2017)
6. Eftimov, T., Korošec, P., Seljak, B.K.: Comparing multi-objective optimization algorithms using an ensemle of quality indicators with deep statistical comparison approach. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI) Proceedings. pp. 2801–2809. IEEE (2017)
7. Eftimov, T., Korošec, P., Seljak, B.K.: Deep statistical comparison applied on quality indicators to compare multi-objective optimization algorithms. In: Machine Learning, Optimization, and Big Data, Third International Workshop, MOD 2017, Volterra, Italy. Springer (2017)
8. Eftimov, T., Korošec, P., Seljak, B.K.: A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics. Information Sciences 417, 186–215 (2017)
9. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization. Journal of Heuristics 15(6), 617–644 (2009)
10. Glickman, M.E.: Example of the glicko-2 system. Boston University (2012)
11. Gordon, S.P.: Visualizing and understanding l'hopital's rule. International Journal of Mathematical Education in Science and Technology pp. 1–10 (2017)
12. Ravber, M., Mernik, M., Črepinšek, M.: Ranking multi-objective evolutionary algorithms using a chess rating system with quality indicator ensemble. In: Evolutionary Computation (CEC), 2017 IEEE Congress on. pp. 1503–1510. IEEE (2017)
13. Riquelme, N., Von Lücken, C., Baran, B.: Performance metrics in multi-objective optimization. In: Computing Conference (CLEI), 2015 Latin American. pp. 1–11. IEEE (2015)
14. Tušar, T., Filipič, B.: Differential evolution versus genetic algorithms in multiobjective optimization. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 257–271. Springer (2007)
15. Yen, G.G., He, Z.: Performance metric ensemble for multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation 18(1), 131–144 (2014)

A seismic isolation optimization methodology adopted with bat algorithm

Gebrail Bekdaş[1][0000−0002−7327−9810], Seda Öncü-Davas[1], Sinan Melih Nigdeli[1][0000−0002−9517−7313] and Cenk Alhan[1][0000−0002−6649−8409]

Istanbul University, Civil Engineering Department Avcılar/Istanbul 34320, Turkey,
seda.oncu@istanbul.edu.tr, bekdas@istanbul.edu.tr,
melihnig@istanbul.edu.tr, cenkalhan@istanbul.edu.tr

Abstract. By using seismic isolation systems at the base of structures, structures and equipment in structures are protected by the reduction of total acceleration and rigid-body motion of the structure. Additionally, a limit for the displacement of the base isolation system have to be considered in the design. In that case, the optimum period and damping ratio of isolation system, which take the superstructure and earthquake excitation characteristic into account, important. In order to obtain an optimum design, an optimization methodology using time history analyses is adopted with a bioinspired algorithm called bat algorithm (BA). The BA based method is numerically demonstrated on a multi-story shear building benchmark problem considering the three cases of the maximum isolation system damping ratio. Also, the results are compared with harmony search (HS) based method and it is shown that the reduction of the ratio of the peak roof acceleration and peak ground acceleration is better than HS for the proposed method.

Keywords: Seismic Isolation System, Optimization, Metaheuristic algorithms, Bat algorithm, Structural Control, Bioinspired Algorithms.

1 Introduction

In cases where the conventional earthquake resistant structural design is inadequate, new approaches have emerged in the design of the earthquake resistant structures. One of those approaches is seismic isolation which reduces the impact of seismic loads on the structure. Superstructure of a seismically isolated structure behaves as a rigid block and a significant portion of the displacement occurs in the isolation system [1]. Thus, structural safety can be improved by preventing damage to the structural system and non-structural elements. A typical seismically isolated building typically consists of two main parts as superstructure and isolation floor. The isolation system consists of isolators and/or dampers placed between the rigid base floor and the foundation.

Seismic isolation systems aim to protect structural integrity by reducing base displacements to practical and economical limits and to protect the vibration-sensitive contents placed in the building by decreasing floor accelerations. Seismically isolated buildings can generally reach the afore mentioned targets when

they are exposed to far-fault earthquakes. However, these buildings may be challenged in simultaneously reducing base displacements and floor accelerations in case of the near-fault earthquakes [2]. Furthermore, if the excessive base displacements exceed the ultimate displacement limits, it can lead to rupture or buckling of the isolator [3]. In order to maintain structural integrity, it is necessary to reduce the base displacements below the limits, which depend on the capacity of isolator displacement and seismic gap. Therefore, the need for additional damping may arise. If additional damping is used, the base displacements can be reduced while the floor acceleration and interstory drifts can be increased. For this reason, it is pointed out that the use of right amount of damping is extremely important and may be a dilemma [4–7]. Therefore, seismic performance of seismically isolated buildings exposed to near-fault earthquakes have recently been investigated with increased interest.

On the other hand, a building may be exposed to both far-fault and near-fault earthquakes due to its location. Thus, it is desired that the building achieves a successful seismic performance through alternative solutions under both near-fault and far-fault earthquakes. Such as using different types of isolators or adding viscous damping to the isolation systems. Hall [5] showed that the use of an optimum amount of additional damping can reduce base displacements without increasing floor accelerations or interstory drifts. Makris [8] investigated the effectiveness of viscous, viscoplastic and friction damping to protect the seismically isolated buildings under near-fault earthquakes. Providakis [9] numerically investigated performances of lead rubber and friction pendulum systems equiped with additional viscous dampers in the isolation systems under far-fault and near-fault earthquakes. Lu and Hsu [10] experimentally investigated variable-frequency rocking bearings with variable mechanical properties in order to avoid excessive isolator displacements due to the near-fault earthquakes. In a comprehensive parametric study [11], the seismic performance of isolation systems with different amount of damping under near-fault earthquakes was investigated. In recent years, it has been proposed to use semi-active control systems, which can adapt themselves to external loads during the lifetime of buildings as a solution to the problems that seismically isolated buildings may face under near-fault earthquakes [12, 13]. On the other hand, in order to overcome above-mentioned dilemma, it can be a good alternative to optimize the passive isolation systems, which are often preferred in terms of ease of application and design. Studies on the optimization of seismically isolated buildings using different methods are available in the literature by using metaheuristic methods such as genetic algorithm [14] and harmony search [15].

2 Seismically Isolated Benchmark Building Model

In order to ensure that the seismically isolated buildings can succeed under both near-fault and far-fault earthquakes, the optimization of seismic isolation systems is carried out by using a methodology using time history analyses adopted

with Bat Algorithm (BA) in this study. The equation of motion of the N-story superstructure is given by Eq. (1) [16].

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + K\mathbf{x} = -M\mathbf{R}(\ddot{x}_0 + \ddot{x}_g) \tag{1}$$

in which, M, C and K are the mass, damping and stiffness matrices of superstructure, respectively. R is the vector of influence, $\ddot{x_0}$ is the relative acceleration of the base and $\ddot{x_g}$ is the earthquake acceleration. The relative floor displacement vector of the building with respect to base, the velocity vector and the acceleration vector are defined as x, $\dot{x}$ and $\ddot{x}$, respectively. Superstructures of the seismically isolated buildings can be modeled as a shear building [17] as seen in Fig. 1. The equation of motion of the base given in Eq. (2) is used for creating the shear building model [18].

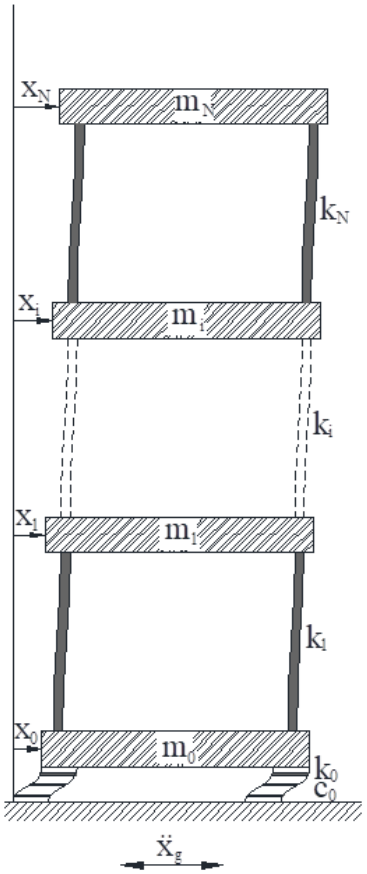


Fig. 1. Schematics of the prototype seismically isolated structure

$$m_0\ddot{x}_0 + c_1\dot{x}_1 + k_1x_1 + f_r = -M_0\ddot{u}_g \tag{2}$$

Here, m_0 is the mass of base and f_r is the restoring force generated in the isolation system. c_1 and k_1 are the damping and stiffness of the first story, respectively. The behavior of a linear seismic isolation system is defined by the Kelvin-Voight model given in Eq. (3), which is consisting of a linear spring in parallel with linear damping, where the effective viscous damping constant and the effective stiffness for the isolation system are symbolized as c_0 and k_0, respectively.

$$f_r = c_0\dot{x}_0 + k_0x_0 \tag{3}$$

The effective stiffness of the isolation system (k_0) can be calculated by Eq. (4), in which M is the total mass of the seismically isolated building and T_0 is the isolation period. Then, considering the isolation system damping ratio of ξ_0, the effective viscous damping constant of the isolation system (c_0) can be obtained by using Eq. (5).

$$k_0 = \frac{4M\pi^2}{T_0^2} \tag{4}$$

$$c_0 = \frac{4M\pi\xi_0}{T_0} \tag{5}$$

3 The Optimization Methodology

A Bat-algorithm-adopted time history analysis program has been developed for the optimization of seismic isolation systems for structures by using Matlab with Simulink [19]. In the time history analyses, the conventional Runge-Kutta method with a sensitive time step of 0.001s is used.

In the optimum design, maximum limits of peak isolation system displacement and the design variables such as the isolation period (T_0) and the isolation damping ratio (ξ_0) should be within a defined realistic range of physical application. The most important factor for seismically isolated structures in order to protect the contents of the structure and reduce earthquake induced forces is the reduction of the floor accelerations. In that case, the optimization objective is to minimize the ratio of peak roof acceleration (PRA) to the peak ground acceleration (PGA) by obtaining an optimum set of isolation system parameters defined in a chosen application range without exceeding a chosen peak isolation system displacement limit. The time history analyses of the system are done to find the responses such as x, $\dot{x}$ and $\ddot{x}$. Then, the maximum peak value of the top

story acceleration (by adding the ground acceleration for the total acceleration) is used as PRA. The maximum value of ground acceleration are used as PGA.

The bat algorithm (BA) is a bioinspired metaheuristic algorithm developed by Yang [20] and it is inspired from the echolocation behaviour of bats. The proposed methodology can be summarized as the following five steps:

Step 1: In the first step, the design constants of the problem are defined. These design constants include the properties of the main structure, ranges of design variables, the acceleration data of earthquake excitations considered in the optimization process and the algorithm-specific parameters of the bat algorithm. The algorithm-specific parameters are the bat population (n), the limits of pulse frequency (f_{min} and f_{max}), the pulse emission rate (r_i) and the loudness (A_i). The optimization process considers the analyses conducted for various earthquake excitations in order to find a general solution.

Step 2: In nature, microbats use echolocation for two reasons such as the sensing of distance and the search for prey. In this process, the bats use frequency tuning by varying the frequency, loudness and pulse emission rates when homing for preys. These characteristics are formulated in the bat algorithm and the location of a bat corresponds to the position vector. For the current study, the position vectors (d_i) from i=1 to n are generated by randomization of design variable values within the defined ranges/limits. The candidate design variables which have a peak isolation system displacement limit exceeding the allowable limit, are regenerated.

Additionally, the objective function (OF) which is minimized is also calculated and stored for all position vectors. In Eq. (6), the i^{th} position vector including the design variables such as the isolation period (T_{0i}) and the isolation damping ratio (ξ_{0i}) is shown. The objective of the optimization problem is to minimize the PRA/PGA for the excitation with the highest value and the optimization process continue until the minimization of the objective function to a desired percentage (DP) value defined by the user. It is formulated as seen in Eq. (7) for the i^{th} position vector (OF_i).

$$\mathbf{d_i} = (T_{0i}; \xi_{0i}) \; i = 1 : n \tag{6}$$

$$OF_i = \frac{PRA}{PGA} \leq DP \; i = 1 : n \tag{7}$$

Step 3: In this step, the position vectors in the bat algorithm are updated according to the equations of the bat algorithm given as Eqs. (8)-(10). First, the frequency (f_i) is adjusted and then, the velocity (v_i) is updated. The parameter; β is a random number between 0 and 1. The best position vector with the minimum objective function value is defined as d^* and the upper scripts represents the iteration number (t). If the generated design variables are out of the range, the variables are assigned with the nearest value within the range. Also, the regeneration of the design variables is done in case of the violation of peak isolation system displacement limit.

$$f_i = f_{min}(f_{max} - f_{min})\beta \tag{8}$$

$$v_i^t = v_i^{t-1} + (d_i^t - d^*)f_i \tag{9}$$

$$d_i^t = d_i^{t-1} + v_i^t \tag{10}$$

Step 4: In this step, the updated position vectors are accepted or not by using the criterion test. If the pulse rate is smaller than a randomly generated number between 0 and 1, local solutions around the best position vector are generated. If the random number is smaller than the loudness and the objective function values for the updated position vectors design variables are better than the existing best solution, then the updated position vectors are accepted. In all other cases, the values of the previous iteration are kept. Also, the value of pulse rate (r_i) is increased according to iteration number, while the value of loudness (A_i) is reduced as formulated in Eqs. (11) and (12). In these formulations, α and γ are constant values and are generally taken as 0.9.

$$A_i^{t+1} = \alpha A_i^t \tag{11}$$

$$r_i^{t+1} = r_i^0(1 - exp(-\gamma t)) \tag{12}$$

The pseudo code of the step 4 is as follows:
if (random number $\geq r_i$)
Select a position vector
Generate a local solution around the selected position vector
end if
if (random number $\leq A_i$ and $(OF_i) \leq$ f(d^*))
Accept the updated position vectors
Increase r_i and reduce A_i
end if

Step 5: The iterative process continues until the objective function of the best solution is smaller than DP. The program can lock if the user assigns a physically impossible DP value. In that case, the DP value has to be increased automatically. The program has an ability to increase the DP value after 600 attempts by 5%.

4 Case Study

As the numerical investigation, a ten-storey shear building is used with same properties for all stories. The critical period of the structure (T) is 1 s. The mass, stiffness coefficient and damping coefficient of a storey are 360 t, 650

MN/m and 6.2 MNs/m, respectively [20]. Also, the mass of the isolation level (m_0) is 360t. The optimum results are also compared with the harmony search (HS) algorithm results.

In the numerical example, the range of the isolation period is between 2 and 4 seconds, which represents a typical range for seismically isolated buildings [21]. The isolation system damping ratios were optimized for three different cases. The lower bound of the isolation system damping ratios are 0, but the upper bounds are 30%, 40% and 50% for case1, 2 and 3, respectively. The peak isolation system displacement limit was taken as 50 cm. The DP value was taken as 0 and the best possible value was searched by the increase of the value.

The algorithm specific parameters such as the minimum frequency (f_{min}), the maximum frequency (f_{max}), the initial pulse rate (r_i^0) and initial loudness (A_i^0) were taken as 0, 1, 0.5 and 1, respectively. The population of bats (n) is 5 in the numerical example. The earthquake excitation used in the example are described in Table 1. These records were downloaded from Pacific Earthquake Engineering Research Centre (PEER) [23].

In Table 1, the peak values of the ground acceleration (PGA), ground velocity (PGV) and ground displacement (PGD) are also shown. Taft and El Centro results represent far-fault motions, while other ones are the near-fault records with high PGV pulses. In Table 2, the optimum results of BA are presented with HS results.

Table 1. Earthquake records used in the optimization

Earthquake	Date	Station	Component	PGA (g)	PGV (cm/s)	PGD (cm)
Imperial Valley	1940	117 El Centro	I-ELC180	0.313	29.8	13.32
Kern Country	1952	1095 Taft	TAFT111	0.178	17.5	8.99
Tabas	1978	9101 Tabas	TAB-TR	0.852	121.4	94.58
Loma Prieta	1989	16 LGPC	LGP000	0.563	94.8	41.18
Erzincan	1992	95 Erzincan	ERZ-NS	0.515	83.9	27.35
Northridge	1994	24514 Sylmar	SYL360	0.843	129.6	32.68

Table 2. Optimum results of numerical example

Algorithm	Case	Optimum isolation period (s)	Optimum isolation damping ratio (%)	Objective function	Base isolation displacement (cm)
BA	1	3.1070	30	0.6076	50
	2	3.4732	40	0.5501	50
	3	3.6712	50	0.5439	50
HS	1	3.1009	29.96	0.6085	49.96
	2	3.4658	39.53	0.5504	50
	3	3.6638	49.75	0.5442	49.97

As seen from the results, BA is more effective than HS in minimization of the objective function. It seems that the maximum limits of isolation system damping ratios are the same for the optimum results and BA is effective in finding the exact solution. The time histories for the isolation system displacement and the total roof accelerations are plotted for case 3 obtained by BA. The total roof accelerations are also compared with the fixed based structure (Figs. 2-7).

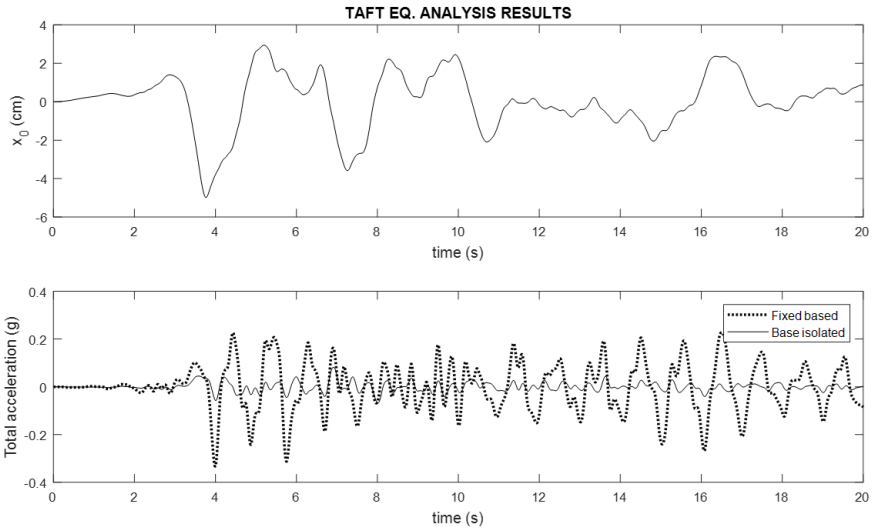


Fig. 2. The response of structure for Taft excitation

5 Discussion and Conclusion

As seen from the time history plots and the maximum responses, the base isolation system optimized by the methodology adopted with BA is effective in obtaining a good passive seismic control. As seen from the optimum results, the aim for the isolation system damping ratio is the maximum allowable value in order to maintain the maximum isolation system displacement limit. The effectiveness of the method can be also recognized from the maximum displacement of the isolation system because this value is at the limit for the optimum variables. Also, a conclusion can be derived for the optimum results of different cases. The limitation of the isolation system damping ratio prevents the increase of the optimum period of the isolation system because of the limitation of the base displacement. For that reason, the OF of Case 1 is higher than the other cases with high maximum damping ratio range and this situation is the opposite of the known increasing effect of damping on the roof acceleration of structures.

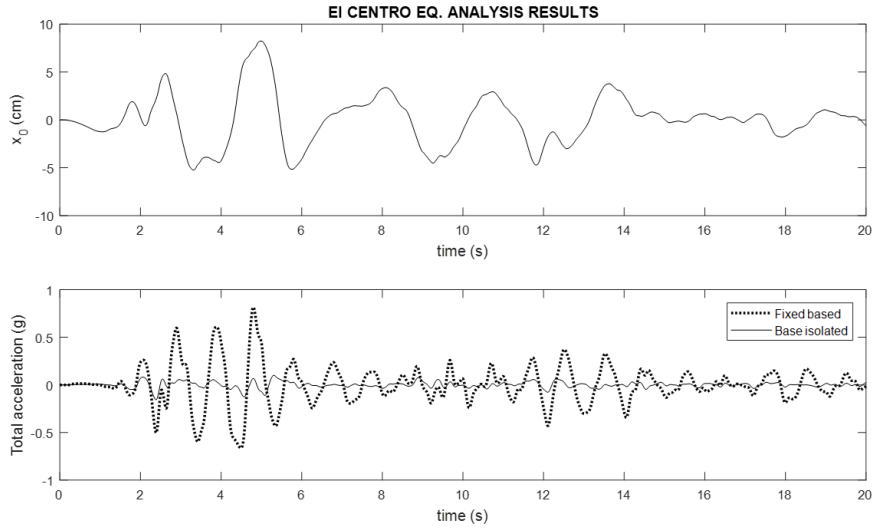


Fig. 3. The response of structure for El Centro excitation

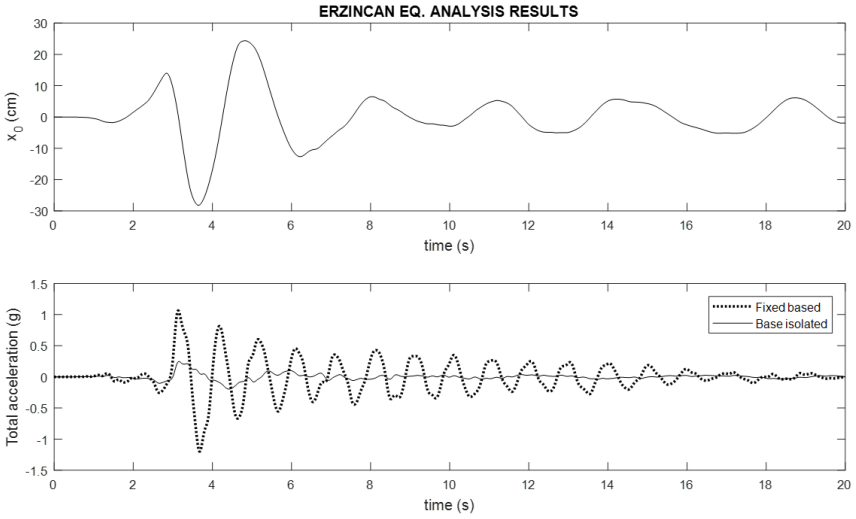


Fig. 4. The response of structure for Erzincan excitation

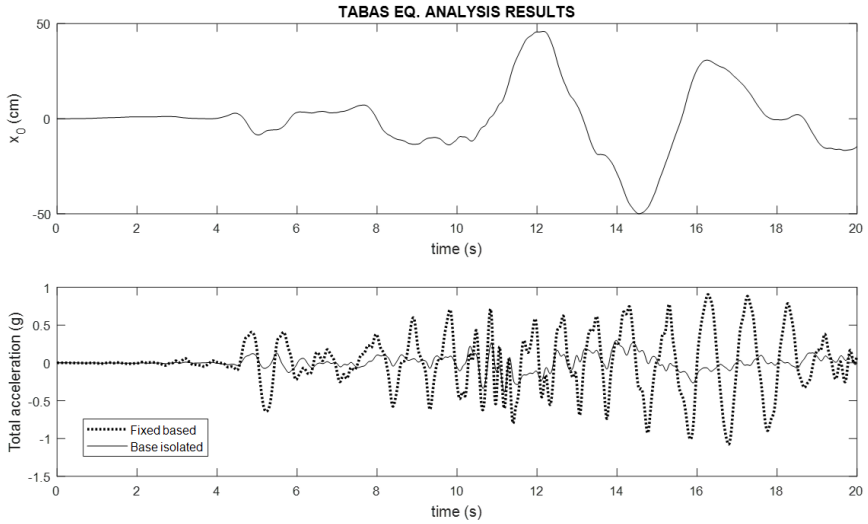


Fig. 5. The response of structure for Tabas excitation

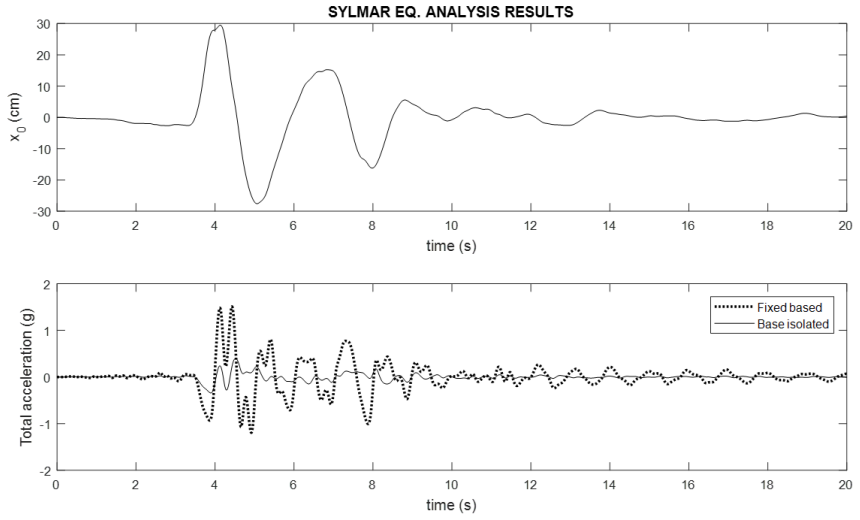


Fig. 6. The response of structure for Sylmar excitation

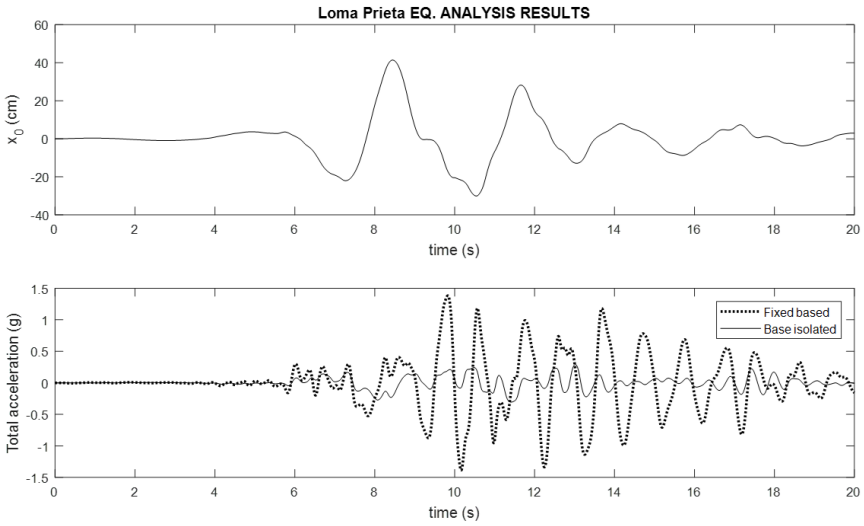


Fig. 7. The response of structure for Loma Prieta excitation

When the optimum results of BA and HS based methods are compared, BA finds more sensitive results compared to HS since the damping ratio and the isolation system displacements are exactly at the limit for BA.

In future studies, the robustness of the isolation system against different ground excitations, the sensitivity analyses of algorithm-specific parameters, different cases of the seismic isolation system displacement limits and the optimization of nonlinear isolation systems can be investigated by using the proposed methodology.

References

1. Naeim, F., Kelly, J.M., 1999, Design of seismic isolated structures: from theory to practice, mechanical characteristics and modeling of isolators, New York, Wiley.
2. Mazza, F., Vulcano, A., 2009, Nonlinear response of RC framed buildings with isolation and supplemental damping at the base subjected to near-fault earthquakes, Journal of earthquake engineering, 13(5):690-715.
3. Nagarajaiah, S., Ferrell, K., 1999, Stability of elastomeric seismic isolation bearings, Journal of Structural Engineering 125(9):946954.
4. Heaton, T.H., Hall J.F., Wald, D.J, Halling, M.W., 1995, Response of high-rise and base-isolated buildings in a hypothetical Mw 7.0 blind thrust earthquake, Science, 267:206-211.
5. Hall, J.F., 1999, Discussion: the role of damping in seismic isolation, Earthquake engineering and structural dynamics, 28:1717-1720.
6. Alhan, C., Gavin, H.P., 2004, A parametric study of linear and non-linear passively damped seismic isolation systems for buildings, Engineering structures, 26(4):485-497.

7. Mazza, F., Vulcano, A., 2009, Nonlinear Response of RC Framed Buildings with Isolation and Supplemental Damping at the Base Subjected to Near-Fault Earthquakes, Journal of earthquake engineering, 13:690-715.
8. Makris, N., Chang, S.P., 2000, Effect of viscous, viscoplastic and friction damping on the response of seismic isolated structures, Earthquake engineering and structural dynamics, 29:85-107.
9. Providakis, C.P., 2009, Effect of supplemental damping on LRB and FPS seismic isolators under near-fault ground motions, Soil dynamics and earthquake engineering, 29:80-90.
10. Lu, LY., Hsu, CC., 2013, Experimental study of variable-frequency rocking bearings for near-fault seismic isolation, Engineering structures, 46:116-129.
11. Alhan C, Öncü-Davas S., 2016, Performance limits of seismically isolated buildings under near-field earthquakes, Eng struct, 116:8394
12. Mehrparvar, B., Khoshnoudian, T., 2012, Performance-based semi-active control algorithm for protecting base isolated buildings from near-fault earthquakes, Earthquake engineering structural dynamics, 11:4355.
13. Ghaffarzadeh, H., 2013, Semi-active structural fuzzy control with MR dampers subjected to near-fault ground motions having forward directivity and fling step, Smart structures and systems, 12(6):595-617.
14. Pourzeynali, S., Zarif, M., 2008, Multi-objective optimization of seismically isolated high-rise building structures using genetic algorithms, Journal of sound and vibration, 311:11411160.
15. Nigdeli, S.M., Bekdaş, G, Alhan, C., 2013, Optimization of seismic isolation systems via harmony search, Engineering optimization, 46(11):1553-1569.
16. Naeim, F., Kelly, J.M., 1999, Design of seismic isolated structures: From theory to practice, Wiley, New York, 93121.
17. Alhan, C., Srmeli, M., 2011, Shear building representations of seismically isolated buildings. Bull earthq eng, 9:1643-1671.
18. Chopra, A.K., 2001, Dynamics of structures: Theory and applications to earthquake engineering. 2nd ed. Prentice Hall, NJ: Pearson Education, ISBN 81-7808-472-4.
19. MathWorks, M.U.S.G., 1992, the Mathworks. Inc., Natick, MA.
20. Yang, X.S. 2010, A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization (NISCO 2010) (Eds. J. R. Gonzalez et al.), Studies in computational intelligence, Springer Berlin, 284, Springer, 65-74.
21. Singh, M.P., Singh, S., Moreschi, L.M., 2002, Tuned mass dampers for response control of torsional buildings, Earthq. eng. struct. dy., 31:749769.
22. Pan, P., Zamfirescu, D., Nakashima, M., Nakayasu N., Kashiwa, H., 2005, Base-isolation design practice in Japan: Introduction to the post-Kobe approach. J Earthq Eng 9:147-171.
23. PEER, 2005, Pacific earthquake engineering resource center: NGA database. University of California, Berkeley. http://peer.berkeley.edu/nga. [Accessed November 2011].

Analyses of cable nets by using energy minimization using several metaheuristic methods

Gebrail Bekdaş[1][0000−0002−7327−9810], Aylin Ece Kayabekir[1], Sinan Melih Nigdeli[1][0000−0002−9517−7313] and Yusuf Cengiz Toklu[2]

[1] Istanbul University, Civil Engineering Department Avcılar/Istanbul 34320, Turkey, bekdas@istanbul.edu.tr, ecekayebekir@gmail.com, melihnig@istanbul.edu.tr, cengiztoklu@gmail.com

[2] Okan University, Civil Engineering Department Tuzla/Istanbul 34959, Turkey, cengiztoklu@gmail.com

Abstract. By using the minimum total potential energy of a structural systems subjected to a defined loading condition, the solution of the responses of system can be found, because systems are in the equilibrium state only in case of minimum energy. In that case, the minimum value of a system can be searched by using randomly generated design variables which are possible solution of the degrees of the system. Thus, metaheuristic methods combined with total potential energy minimization (TPO/MA) are an effective method to be an alternative to conventional methods. In this study, several metaheuristic algorithms are used in the methodology to solve the responses of cable net structures. The efficiency of the methods is tested on four cable structures which have increasing number of design variables from example 1 to 4. The investigated algorithms are flower pollination algorithm, teaching learning based optimization and Jaya algorithm. All algorithms are effective and robust on finding the design variables for minimum energy, but Jaya algorithm seems as the best one because of the simply application to the problem.

Keywords: Total potential energy method using metaheuristic algorithms, cable nets, metaheuristic methods, flower pollination algorithm, teaching learning based optimization, Jaya algorithm

1 Introduction

Cable net structures are the systems which are only generated by cable element. These structures are in the group of truss-like structures, but the nature of the cables has a special condition since the cable members cannot carry the compressive force. In that case, the cable members are in tension or unloaded. Due to this behavior, the stability of the cable net structures is provided by applying pretension forces to several or all members. In practice, the general application area of cable nets are roofs.

Several methods have been proposed for non-linear analyses of cable net structures and the following methods which are the variants clasical approaches like the finite elements and sttifness methods:

- The finite deflection theory for space structures [1]
- The direct stiffness method [2]
- An iterative approach highly nonlinear problems [3]
- A method based on discrete mathematical model for single layer cable nets [4]
- A method using a series of finite length curved elements [5]
- A finite element model for members subjected to static and dynamic loads [6]
- A search method based on minimum potential energy theory [7]
- A method using a small strain elastic catenary element [8] in a dynamic relaxation technique [9]
- A method proposing a way of generation of stiffness of cable members for fast computations [10]
- An iterative Newton-Raphson method [11]
- A method separating the complexity of nonlinear numerical algorithm from the basic principles [12]
- The investigation of limit state behavior and the effect of pre-stressing in improving the bearing capacity [13]
- An exact geometrical nonlinear equilibrium equations in non-dimensional form [14]
- A method combining analytic solution and virtual work principle for a two-node catenary cable element [15]
- An investigation of minimum complementary energy involving on stress components [16]
- A nonlinear closed form static computational model [17]
- A generalized displacement control method [18]
- A combined method of nonlinear displacement method and force density method [19]
- A formulation of the analyses of Suspen-Dome structures with multi-node sliding cable members [20]
- A point based iterative method adapting a simple convergence procedure [21]
- An incremental iterative solution based on the Newmark direct integration method and the Newton-Raphson method [22]

As an alternative solution to the mentioned classical based methods, metaheuristic algorithms employing methods can be an effective way for structural analyses. The minimization of total potential energy was used by employing the music-inspired algorithm called harmony search for the analyses of cable structures [23].

In the present study, the efficiency of three metaheuristic algorithms is evaluated for the analyses of cable net structures using total potential energy minimization. The investigated algorithms are the bio-inspired flower pollination

algorithm (FPA) and parameter free algorithms such as teaching learning based optimization (TLBO) and Jaya algorithm (JA). These algorithms are new generation algorithms and these algorithms need performance evaluation. The investigation is presented on four cable-nets structures with different sizes and number of design variables.

2 Methodology

The presented study investigates the effect of different algorithms on cable net structures for the total potential optimization using metaheuristic algorithm used for the nonlinear analyses of structural systems [23–27]. The aim of the method is to minimize the total potential energy by generating candidate solutions for the design variables of the problem which are the displacements of nodal point free in motion. For example, a 2 member structural system with a single displacement is given in Fig. 1. The deformed shape of the system can be as 1a, 1b, 1c, 1d and 1e. The form with the minimum potential energy is the stable equilibrium position. In that case, the total potential energy (Π) of the system can be found as shown in Eq. (1). This objective function includes the strain energy (u) and the work done by external forces (w) which are shown as Eq. (2) and Eq. (3), respectively.

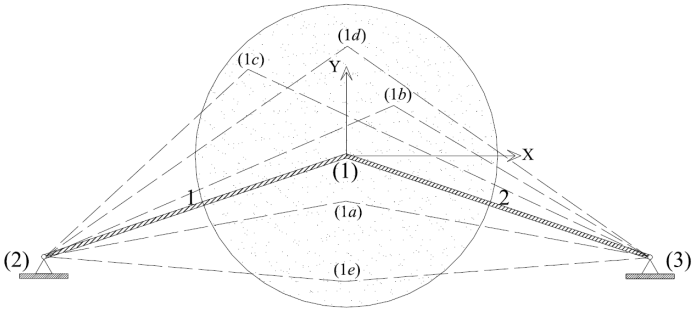


Fig. 1. An example for displacement range

$$\Pi = U - W \tag{1}$$

$$U = \frac{1}{2} \int \epsilon^T \sigma \, dV \tag{2}$$

$$W = \int (T_x u + T_y v + T_z w) \, dS \tag{3}$$

ϵ^T, σ and V represent the strain vector, stress vector and volume. T_x, T_y and T_z are the forces in unit area in the corresponding directions and the u, v, and w are the displacements at x, y and z directions, respectively. When the corresponding modifications are done for truss-like structures, the objective function (Π) takes the form given as Eq. (4).

$$\sum_{j=1}^{N_m} e_j A_j L_j - \sum_{i=1}^{N_p} P_i u_i \tag{4}$$

A_j and L_j represent the area and length of j^{th} structural member for a system with N_m member subjected to N_p loads. The forces are shown with P_i for the i^{th} force and u_i is generalized deflections coupled with the loads. e_j is the strain energy density of i^{th} member which is given as Eq. (5) if the cable is in tension or Eq. (6) if the cable is in compression. E_i and ϵ_i are the elasticity modulus and strain of i^{th} member, respectively.

$$e_i = \frac{1}{2} E_i \epsilon_i^2 \tag{5}$$

$$e_i = 0 \tag{6}$$

In the study, three algorithms which have different specific properties, are chosen. These algorithms are FPA developed by Yang [28], TLBO developed by Rao et al. [29] and JA developed by Rao [30]. In order to focus on the numerical investigations and the analyses of the problem, the general formulations of the algorithms are skipped and these formulations can found in the cited studies. Only specific differences are mentioned in the study.

FPA is generated by considering flower constancy and different types of flower pollinations such as biotic, abiotic, cross and self-pollinations. Two different phases such as global and local pollinations are created and these phases are chosen by using a switch probability (ps). Global pollination uses a Levy distribution while a linear distribution used for local one.

TLBO simulates the teaching and learning processes of a class. Two phases of education which are teacher and learner phases, are proposed to solve the problems. These two phases are consequently applied for all iterations and the only user defined parameter is the population.

JA is a single phase algorithm and contains no specific parameters. The name comes from the Sanskrit word which is victory in English. The convergence is provided by considering the best and the worst existing solutions. In the study, a single phase algorithm (JA) and two double phase algorithms using a switch probability (FPA) and consequent application of both phases (TLBO) are compared.

3 Numerical examples

3.1 Example 1: Flat cable net 1x1

In the first example, mathematical model with one freedom in the Fig. 2 is solved using total potential energy minimization (TPO/MA) method. In this model each of cables is identical. Cross sectional area of each cable and the modulus of elasticity are 0.785 mm^2, 124800 N/mm^2 respectively. The pretension force with an intensity of 200 N is applied each of cable and there is a point load on the node number 3 with intensity of 15 N. Optimum displacement of the node 3 and the total potential energy values of the system obtained with TLBO, FPA, JA are presented in Table 1. In the Tables, the displacements in x, y and z directions are shown by δ_x, δ_y and δ_z respectively.

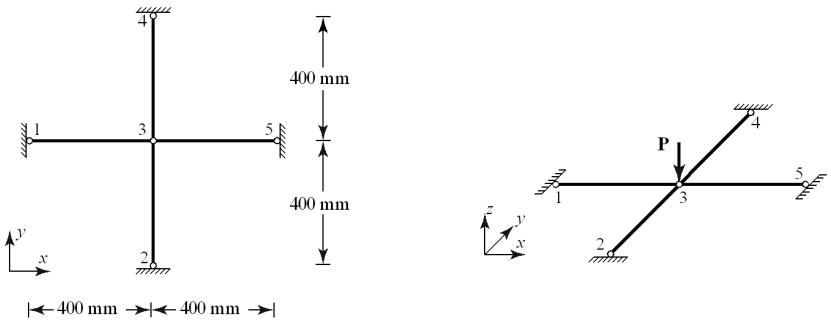


Fig. 2. Flat cable net 1x1 [23]

Table 1. TP energies (Nmm) and displacements (mm) for example 1

Node	FPA			TLBO			JA		
	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z
3	0	0	-6.98	0	0	-6.98	0	0	-6.98
Minimum TP (kNm)	272.4704			272.4704			272.4704		
Average TP (kNm)	272.4704			272.4704			272.4704		
Standard deviation	0			0			0		
Number of analyses	10024			24576			12414		

3.2 Example 2: Flat cable net 2 x 1

In the second example, a flat system with two freedoms is solved (Fig. 3). The cross sectional area of each cable is 2 mm^2 and the modulus of elasticity is 11000

N/mm^2. The pretension force is 500 N in the all cables. Point forces with the intensity of 200 N are applied to nodes 1 and 2. Optimum displacement of nodes and the total potential energy values of the system are presented in Table 2.

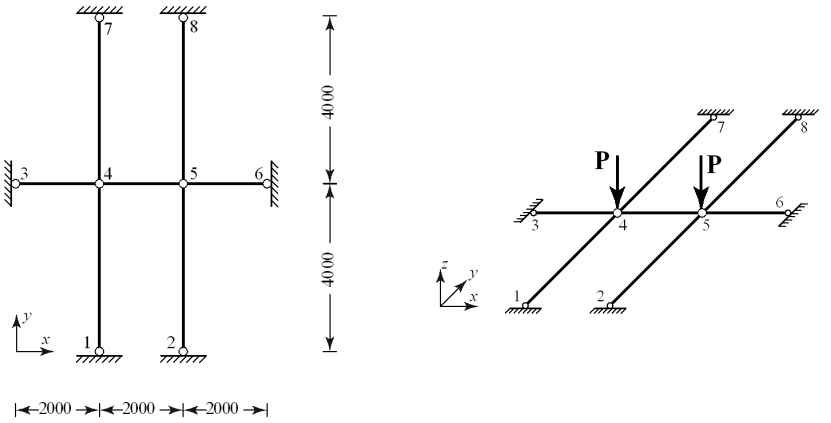


Fig. 3. System with seven cables [23]

Table 2. TP energies (Nmm) and displacements (mm) for example 2

Node	FPA			TLBO			JA		
	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z
1	0	-3.3	-199.75	0	-3.3	-199.75	0	-3.3	-199.75
2	0	-3.3	-199.75	0	-3.3	-199.75	0	-3.3	-199.75
Minimum TP (kNm)	-37442.86			-37442.86			-37442.86		
Average TP (kNm)	-37442.86			-37442.86			-37442.86		
Standard deviation	0			0			0		
Number of analyses	9832			20456			11195		

3.3 Example 3: Flat cable net 2 x 2

In the third example, a flat cable system consists of 12 cables with 4 free nodes is investigated, as shown in Fig 7. Cross-sectional area and the modulus of elasticity multiplication of each member is 97.97 kN. A pretension force with 200 kN exists in every cable and point loads are applied to nodes 4, 5 and 8. Displacement of the free nodes and the total potential energy values of the system can be seen in Table 3.

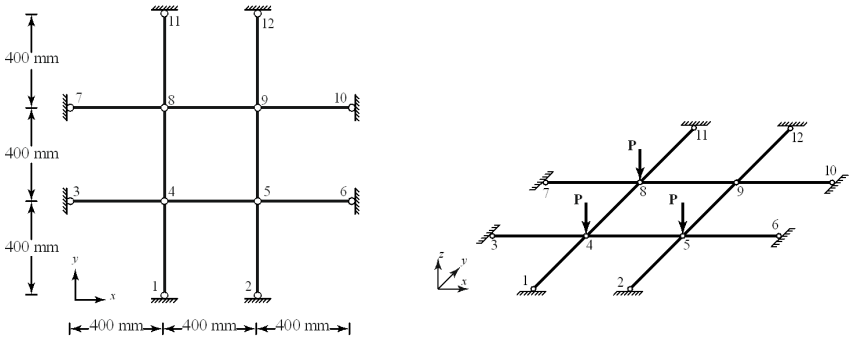


Fig. 4. Flat cable net 2 x 2 [23]

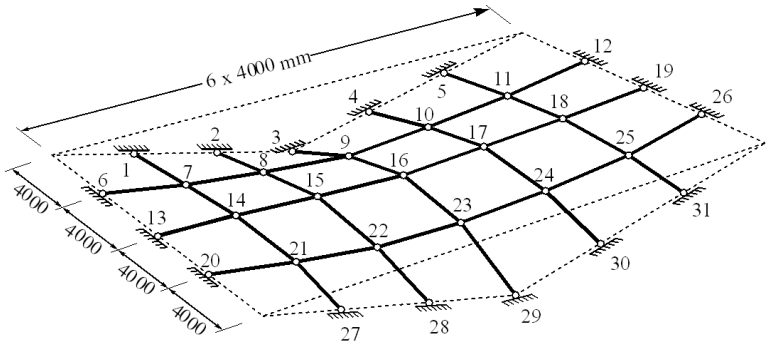


Fig. 5. Spatial cable network system [23]

Table 3. TP energies (Nmm) and displacements (mm) for example 3

Node	FPA			TLBO			JA		
	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z
4	-0.07	-0.07	-12.17	-0.07	-0.07	-12.17	-0.07	-0.07	-12.17
5	0.04	-0.07	-11.18	0.04	-0.07	-11.18	0.04	-0.07	-11.18
8	-0.07	0.04	-11.18	-0.07	0.04	-11.18	-0.07	0.04	-11.18
9	0.04	0.04	-5.59	0.04	0.04	-5.59	0.04	0.04	-5.59
Minimum TP (kNm)	706.8089			706.8089			706.8089		
Average TP (kNm)	706.8089			706.8089			706.8089		
Standard deviation	0			0			0		
Number of analyses	11365			24850			12461		

3.4 Example 4: Spatial Cable Network

In the last example, a spatial cable network system consisting of 38 cables is solved. This system has symmetry with respect to both x and y axes. The cross sectional areas and the pretension forces are 350 mm^2 and 90 kN in x-direction cables and 120 mm^2 and 30 kN in y-direction cables, respectively. The modulus of elasticity is 160 kN/mm^2 in the all cables. The concentrated loads of 6.8 kN are applied to all internal nodes. The z-coordinates of quarter system and results can be seen in Table 4.

Table 4. TP energies (Nmm) and displacements (mm) for spatial cable network

Node	z-coord	FPA			TLBO			JA		
		δ_x	δ_y	δ_z	δ_x	δ_y	δ_z	δ_x	δ_y	δ_z
1	1000.0	0	0	0	0	0	0	0	0	0
2	2000.0	0	0	0	0	0	0	0	0	0
3	3000.0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	819.5	-5.03	0.40	29.48	-5.03	0.40	29.48	-5.03	0.40	29.48
8	1409.6	-2.23	0.40	17.12	-2.23	0.40	17.12	-2.23	0.40	17.12
9	1676.9	0	-2.36	-3.19	0	-2.36	-3.19	0	-2.36	-3.19
13	0	0	0	0	0	0	0	0	0	0
14	687.0	-4.93	0	42.89	-4.93	0	42.89	-4.93	0	42.89
15	1147.8	-2.55	0	44.32	-2.55	0	44.32	-2.55	0	44.32
16	1317.6	0	0	42.13	0	0	42.13	0	0	42.13
Minimum TP (kNm)		6515666.7			6515666.7			6515666.7		
Average TP (kNm)		6515666.7			6515666.7			6515666.7		
Standard deviation		0			0			0		
Number of analyses		12270			27106			12378		

Table 5. The TP results of example 3 for different ps values

ps	Minimum TP	Analyses Number
0	706.81	5693
0.1	706.81	2220
0.2	706.81	5837
0.3	706.81	15038
0.4	706.81	4173
0.5	706.81	3157
0.6	706.81	9424
0.7	706.81	12641
0.8	706.81	14958
0.9	706.81	13995
1.0	706.81	18491

Table 6. The TP results of example 4 for different ps values

ps	Minimum TP	Analyses Number
0	6515667	16723
0.1	6515667	10567
0.2	6515667	15457
0.3	6515667	14375
0.4	6515667	17072
0.5	6515667	8235
0.6	6515667	19066
0.7	6515667	4048
0.8	6515667	14911
0.9	6681623	16302
1	6858222	19462

4 Discussion and Conclusion

For the first numerical example with three design variables, all algorithms are effective for all 30 runs of the iterative process, but TLBO needs nearly double number of analyses to reach to the optimum value. The second example has 6 design variables and the algorithm are also effective and robust. When the number design variables are increased to 12, the algorithm can also show the same efficiency. The last example is a spatial cable network with the most design variables. The efficiency and robustness of all algorithms prove the feasibility of the TPO/MA as an alternative solution method.

When the algorithms are compared, FPA and JA are better than TLBO in computational effort. JA seems as a better choice since it is simple to apply. Essentially, JA has no specific parameters. In FPA, the choosing of switch probability (ps) may be effective on the solution. In the present study, ps is taken as 0.5. It is chosen according to tests done for various ps values for examples 3 and 4. The TP values are given in Table 5 and 6 for example 3 and 4, respectively.

The result of the third example is the same for different ps values, but when ps is near to 1 (higher probability for the global pollination), the best solution cannot be found for the 20000 maximum iterative analyses for the example 4. Also, the number of analyses are generally between 0.4-0.6 for ps values. In that case, using pn as 0.5 may be the best option.

Since TLBO uses two phases of optimization, it uses 2 round of analyses in a single iteration. This situation increases the computation time and maximum number of analyses are 40000 for TLBO while 20000 maximum analyses are done for the others. All algorithms are effective to find the precise analyses results for the maximum number of analyses.

References

1. Saafan, S. A. (1970). Theoretical analysis of suspension roofs. Journal of the Structural Division, 96(2), 393-405.
2. Baron, F., Venkatesan, M. S. (1971). Nonlinear analysis of cable and truss structures. Journal of the Structural Division, 97(2), 679-710.
3. Kar, A. K., Okazaki, C. Y. (1973). Convergence in highly nonlinear cable net problems. Journal of the Structural Division, 99(3), 321-334.
4. West, H. H., Kar, A. K. (1973). Discretized initial-value analysis of cable nets. International Journal of Solids and Structures, 9(11), 1403-1420.
5. Gambhir, M. L., de V. Batchelor, B. (1977), A finite element for 3-D prestressed cablenets. Int. J. Numer. Meth. Engng., 11: 16991718.
6. Ozdemir, H. (1979). A finite element approach for cable problems. International Journal of Solids and Structures, 15(5), 427-437.
7. Monforton, G. R., El-Hakim, N. M. (1980). Analysis of truss-cable structures.Computers and Structures, 11(4), 327-335.
8. Jayaraman, H. B., Knudson, W. C. (1981). A curved element for the analysis of cable structures. Computers and Structures, 14(3), 325-333.
9. Lewis, W. J., Jones, M. S., Rushton, K. R. (1984). Dynamic relaxation analysis of the non-linear static response of pretensioned cable roofs.Computers and structures, 18(6), 989-997.
10. Desai, Y. M., Popplewell, N., Shah, A. H., Buragohain, D. N. (1988). Geometric nonlinear static analysis of cable supported structures. Computers and structures, 29(6), 1001-1009.
11. Eisenloffel, K., Adeli, H. (1994). Interactive microcomputer-aided analysis of tensile network structures. Computers and structures, 50(5), 665-675.
12. Kwan, A. S. K. (1998). A new approach to geometric nonlinearity of cable structures. Computers and structures, 67(4), 243-252.
13. Liew, J. R., Punniyakotty, N. M., and Shanmugam, N. E. (2001). Limit-state analysis and design of cable-tensioned structures. International Journal of Space Structures, 16(2), 95-110.
14. Gasparini, D., Gautam, V. (2002). Geometrically nonlinear static behavior of cable structures. Journal of Structural Engineering, 128(10), 1317-1329.
15. Wang, C., Wang, R., Dong, S., Qian, R. (2003). A new catenary cable element. International Journal of Space Structures, 18(4), 269-275.
16. Kanno, Y. and Ohsaki, M. (2005). Minimum principle of complementary energy for nonlinear elastic cable networks with geometrical nonlinearities. Journal of optimization theory and applications, 126(3), 617-641.

17. Kmet, S.and Kokorudova, Z. (2006). Nonlinear analytical solution for cable truss.Journal of engineering mechanics, 132(1), 119-123.
18. Yang, Y. B.and Tsay, J. Y. (2007). Geometric nonlinear analysis of cable structures with a two-node cable element by generalized displacement control method. International Journal of Structural Stability and Dynamics, 7(04), 571-588.
19. Such, M., Jimenez-Octavio, J. R., Carnicero, A. and Lopez-Garcia, O. (2009). An approach based on the catenary equation to deal with static analysis of three dimensional cable structures. Engineering Structures, 31(9), 2162-2170.
20. Chen, Z. H., Wu, Y. J., Yin, Y. and Shan, C. (2010). Formulation and application of multi-node sliding cable element for the analysis of Suspen-Dome structures.Finite elements in analysis and design, 46(9), 743-750.
21. Nuhoglu, A. (2011). Nonlinear analysis of cable systems with point based iterative procedure. Scientific Research and Essays, 6(6), 1186-1199.
22. Thai, H. T. and Kim, S. E. (2011). Nonlinear static and dynamic analysis of cable structures. Finite elements in analysis and design, 47(3), 237-246.
23. Toklu, Y. C., Bekdas, G. and Temur, R. (2017). Analysis of cable structures through energy minimization. Structural Engineering and Mechanics, 62(6), 749-758.
24. Toklu, Y. C. (2004). Nonlinear analysis of trusses through energy minimization.Computers and structures, 82(20), 1581-1589.
25. Toklu, Y.C., Bekda, G. andTemr, R., (2013). Analysis of Trusses by Total Potential Optimization Method Coupled with Harmony Search, Structural Engineering and Mechanics, 45(2), 183-189.
26. Toklu, Y. C., Toklu, N. E. (2013). Analysis of structures by Total Potential Optimization using Meta-heuristic Algorithms (TPO/MA). In Siarry, P. Heuristics: Theory and Applications, Nova Science. Chapter 16. pp. 345-374.
27. Temr, R., Trkan, Y. S. and Toklu, Y. C. (2014). Geometrically Nonlinear Analysis of Trusses Using Particle Swarm Optimization. In Recent Advances in Swarm Intelligence and Evolutionary Computation (Editor: Xin-She Yang) Springer Series: Study in Computational Intelligence (SCI). In press
28. Yang, X. S. (2012). Flower pollination algorithm for global optimization. In Unconventional Computation and Natural Computation 240-249.
29. Rao, R. V., Savsani, V. J., Vakharia, D. P. (2011). Teachinglearning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Design, 43(3), 303-315.
30. Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7(1), 19-34.

The Population Factor on Metaheuristic Based Analyses of Truss Structures

Aylin Ece Kayabekir[1], Gebrail Bekdaş[1 [0000−0002−7327−9810]], Sinan Melih Nigdeli[1 [0000−0002−9517−7313]] and Yusuf Cengiz Toklu[2]

[1] Istanbul University, Civil Engineering Department Avcılar/Istanbul 34320, Turkey,
ecekayebekir@gmail.com, bekdas@istanbul.edu.tr, melihnig@istanbul.edu.tr
[2] Okan University, Civil Engineering Department Tuzla/Istanbul 34959, Turkey,
cengiztoklu@gmail.com

Abstract. In this study, the population factor was investigated for the analyses of truss structures. In the methodology, the total potential energy minimization using metaheuristic algorithms (TPO/MA) was used. The bioinspired algorithms investigated in the study are Differential Evolution (DE), Particle Swarm Optimization (PSO) and Flower Pollination Algorithm (FPA). For different number of population, FPA is effective on the solution and robust comparing to the other classical algorithms.

Keywords: Truss Structures, Total potential energy minimization, Bioinspired algorithms.

1 Introduction

Metaheuristic algorithms are generally used in optimization problems. Whereas, these algorithms can be also used in the analyses instead of classical methods in engineering. For example, metaheuristic algorithms may be used in finding minimum potential energy of a structural system with several degrees of freedoms. The set of solution of degrees of freedom with the minimum potential energy will be the true solution because of the equilibrium of the system. By using total potential energy minimization using metaheuristic algorithms (TPO/MA)[1], the non-linear analyses of truss system can be automatically done while the well-known finite element method needs to be combined with several methodologies. In TPO/MA, the parameters and population (n) selection are effective on the robustness of the method.

2 Methodology

The objective of the methodology is to minimize the total potential energy of the structural system by generating possible solutions for the nodal displacements of the structure. The system is at equilibrium at the minimum total potential energy level. In that case, the solution of the structural system can be found by considering the second order effects which are the additional responses resulting

from the deflection of the system. In the methodology, n sets of randomly defined nodal displacements generate the solution matrix. According to algorithm formulations, these sets are modified according to the corresponding total potential energy level. In the study, Differential Evolution (DE) [2], Particle Swarm Optimization (PSO) [3] and Flower Pollination Algorithm (FPA) [4] are adopted with the minimization of total potential energy problem and the factor of the population number was investigated in the numerical example.

3 Numerical examples

As a numerical example, the 3-bar plane truss system shown in Fig. 1 is demonstrated for the investigation of the population number factor. The truss system is loaded from the node 4. The P loads applied to the node 4 have equal intensity in x (10 kN) and y (-10 kN) direction. The modulus of elasticity; E is 200 GPa for all the members and the equal cross-sectional area of all members is 1 mm^2. The number of population is investigated for 2, 3, 5, 10, 15, 20, 30 and 50. The DE parameters are 1 and 0.5 for the amplitude control parameter (F) and Crossover Constant (CR), respectively. The PSO learning parameters; α and β are taken as 1 and the inertia function; (t) are taken as a constant value (0.2). These parameters are the best tested parameters for the problem. The iterative analysis process was repeated for 30 independent runs. The analyses are presented in Tables 1-3 for DE, PSO and FPA, respectively. The tables contain the minimum total potential energy (TP) value of 30 runs, the average value, the standard deviation value and the number of iterations to reach the minimum TP value are shown. The optimum second order analyses results are 3.6320 mm and -1.5309 mm for the displacements; u_4 and v_4 which are the nodal displacements of node 4 in x and y coordinates, respectively. The minimum TP of the system is -25536.85 kNm.

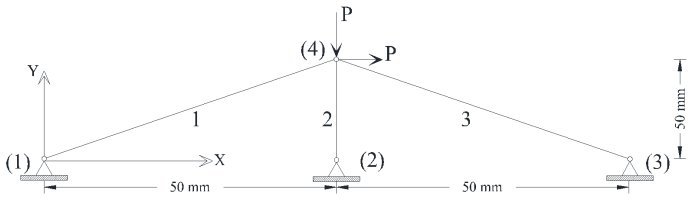


Fig. 1. 3-bar plane truss structure

4 Discussion and Conclusion

It is clearly seen that DE aims to local optimum values for low population values and the solution cannot be found if the population value is lower than 10. Also,

Table 1. Solutions for the 3-bar plane truss (DE)

Population No	2	3	5	10	15	20	30	50
Min. TP (kNm)	-17919.8	-25367.1	-25533.93	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85
Av. TP	111509.2	46237.8	-11076.60	-25442.31	-25535.48	-25536.85	-25536.85	-25536.85
Std. dev.	102919.2	72529.9	20579.29	199.05	5.32	0.0005	0	0
Iter. No.	3	4	9	33	84	157	212	249

taking the population value as 10 is not robust for DE because of the big standard deviation value. The same conclusions can be done for PSO, but it is not the worst one since the minimum solution is found if the population number is 5 or more. FPA is generally robust and effective except for the case in which the population value is 2.

Table 2. Solutions for the 3-bar plane truss (PSO)

Population No	2	3	5	10	15	20	30	50
Min. TP (kNm)	- 24614.0	-25536.3	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85
Av. TP	88083.2	17459.2	-25527.43	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85
Std. dev.	123874.6	57249.1	39.24	0	0	0	0	0
Iter. No.	34	56	831	1003	882	1099	243	202

Table 3. Solutions for the 3-bar plane truss (FPA)

Population No	2	3	5	10	15	20	30	50
Min. TP (kNm)	-25536.8	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85
Av. TP	17544.4	-25536.84	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85	-25536.85
Std. dev.	68746.8	0.02	0	0	0	0	0	0
Iter. No.	272	1345	1195	1086	1079	1019	283	283

References

1. Toklu, Y. C. (2004). Nonlinear analysis of trusses through energy minimization.Computers and structures, 82(20), 1581-1589.
2. Storn, R., Price, K. (1997). Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), 341-359.
3. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks No. IV, Perth Australia; November 27-December 1, p. 19421948 (1995).
4. Yang, X. S. (2012). Flower pollination algorithm for global optimization. In Unconventional Computation and Natural Computation 240-249.

An Efficient Cultural Algorithm for Engineering Design Optimization Problems

Mostafa Z. Ali[1], Noor H. Awad[2]
Jordan University of Science & Technology, Jordan 22110
School of Computer Information Systems[1], School of Computer Engineering[2]
mzali.pn@gmail.com, noorawad1989@gmail.com

Abstract— Many real-world problems are considered optimization problems because of their nature that one or more objectives need to be optimized. Such problems state a challenge for researchers to design efficient algorithms capable of finding the optimal solution with least computational cost. Cultural Algorithm is considered one of the most evolutionary algorithms that uses the knowledge to guide search towards promising regions. When applying to real-world optimization problems with complex landscape, it suffers from fast convergence and stagnation scenarios. This paper presents a hybrid approach of an improved version of Cultural Algorithm and an adapted Multiple Trajectory Search to solve real-world optimization problems, namely *i*CA-MTS. Three improved knowledge sources have been used which are: situational knowledge, normative knowledge, and topographic knowledge. A quality function is used to control the work between both components: Cultural Algorithm and Multiple Trajectory Search. A set of engineering design problems is used to test the performance of proposed algorithm. Comparative studies show that the new algorithm has a superior performance comparing to other several evolutionary algorithms.

Keywords—Real-world problems, Cultural Algorithm, Evolutionary Algorithm, Multiple Trajectory Search.

I. INTRODUCTION

Real-world problems have attracted substantial research interest. Most of these problems can be expressed as single objective global optimization problems. In general, optimization problems can be formulated mathematically as minimizing the cost function $f(\vec{X})(f:\Omega\subseteq\Re^D\rightarrow\Re)$ where Ω is a non-empty and bounded set that represents the domain of the decision variable space. The optimization task is to find the best parameter vector $\vec{X}^*$ from a set of available alternatives based on solution objectives and some constraints criteria [1].

Many evolutionary algorithms have been proposed in the literature such as: Genetic Algorithms (GA) [2], Particle Swarm Optimization (PSO) [3], Differential Evolution (DE) [4], Ant Colony Optimization (ACO) [5], and Cultural Algorithms (CA) [6]. When applying to real-world problems, many evolutionary algorithms can often get trapped in local optimal solution due to complex landscape of these problems. Many researchers suggest the solution of this problem by developing new enhanced algorithms based on the idea of hybridizing with other algorithms or local searches. Compare to their constituent algorithms, this mode provides a synergistic fashion to develop new algorithm with search elements which show more satisfactory performance [7-9].

Cultural Algorithm is a population-based algorithm which uses cultural evolution process to guide evolutionary search. The knowledge component is the key difference between this algorithm and all other evolutionary algorithms. Recently it has been growing interest in developing new improved Cultural Algorithms for the solution of different optimization problems and real-life applications [10-13]. The idea of hybridization is also incorporated in Cultural Algorithm to avoid premature convergence and stagnation scenarios. In [14], the Cultural Algorithm is hybridized with a recapitulated local search. A common knowledge space is used in a multi-populations scheme to integrate the generated knowledge. Among these sub-populations, a knowledge migration with less communication cost is employed to lead search in new approaches.

In Cultural Algorithm, the idea of implementing an evolutionary algorithm as a population space is well known in order to enrich search ability. In [15], the Particle Swarm Optimization is used in population space of an improved Cultural Algorithm. This combination is used for numerical optimization over continuous spaces. An improved PSO has also been merged with Cultural Algorithm in [16]. Another technique that used the Differential Evolution with Cultural Algorithm is introduced in [17]. In this algorithm, the diversity of problem solvers have been increased by incorporating operations of Differential Evolution in CA evolution. The Genetic Algorithm is also hybridized with Cultural algorithm. In [18], a constructive knowledge is being extracted from a population space of a Genetic Algorithm by using Cultural Algorithm. The job shop scheduling problems can be solved by employing this algorithm. An indistinguishable approach is implemented for real-world optimization in [19]. A novel class of hybrid niche Cultural Algorithms is introduced in [10]. In this work, three different approaches have been proposed for solving engineering applications.

While the aforementioned examples show the advantages of hybridization in Cultural Algorithm, the potential cost is also undeniable [10-12]. The first aspect is how to balance the component algorithms to satisfy the good balance between

exploration and exploitation phases. The second is the extra computational resources that may require in the optimization engine when there are two or more combined algorithms. In this paper, we keep these two aspects in consideration and develop new hybrid approach that combines Cultural Algorithm and Multiple Trajectory Search. An improved version of Cultural Algorithm has been used in this hybrid approach. The situational, topographical and normative knowledge have been used with a shared knowledge to help guiding the search toward the promising regions. In order to control synergistic between two algorithms, a quality function is applied. This algorithm is introduced for soling real-world optimization problems. It can also assist to curb the quantity of Function Evaluations (FE) that are needed to solve a problem and find an optimal solution as least as good as the best published results.

The remaining sections complete this paper as follows. Section 2 briefly introduces Cultural Algorithm and Multiple Trajectory Search. In Section 3, the proposed method is elaborated. Section 4 describes the real-world optimization problems, parameter settings and simulation results. Finally, section 5 summarizes the conclusion of this paper.

II. SCIENTIFIC BACKGROUND

A. *Cultural Algorithm*

Cultural Algorithm (CA) has been introduced by Reynolds [20]. It is derived in nature from the cultural evolution process and consists of two components: belief space and population space. In order to regulate the implementation of common knowledge for generating new individuals, the use of communication channels is essential between the two spaces. Five knowledge sources have been introduced in Classical Cultural Algorithm which are: topographic knowledge, situational knowledge, normative knowledge, domain knowledge and history knowledge. These knowledge types are residing in the belief space and responsible of guiding evolution in the search landscape by accumulating information regarding the problem domain and search space. The basic pseudo-code of Cultural Algorithm is presented in Fig. 1. After initializing the population and belief spaces, the *Obj*() function evaluates the individuals.

```
1.  Begin
2.   Set t=0
3.   Initialize belief space, B^t
4.   Initialize population space, P^t
5.   Repeat
6.     Evaluate P^t using Obj()
7.     Update B^t using Accept()
8.     Generate P^t using Influence()
9.     t=t+1
10.    Select p^t from p^(t-1)
11.  Until (termination condition met)
12. End
```

Fig. 1. Pseudo-code of the Cultural Algorithm

After that, The responsibility of *Accept*() function is to choose the finest individuals which are employed by the belief space knowledge applying the function *Update*(). Next, a new set of individuals are generated using *Influence*() function with the help of knowledge sources that are chosen using the roulette wheel selection [20].

B. *Multiple Trajectory Search*

The Multiple Trajectory Search is a kind of global optimizer algorithm that utilizes the Simulated Orthogonal Array with different step sizes to generate new good solutions [21]. The idea behind this algorithm is to move in the parameter space using distinct step sizes to change the position in each dimension to further good position. Inside Multiple Trajectory Search, various local search techniques are employed to generate new individuals capable of maneuvering search toward different directions. For each local search method, two parameters have been used and dynamically change in each generation which are Search Range (*SR*) and Test Grade (*TG*). The search range, *SR* is used and initially set to $(Ub - Lb)/2$ where Lb and Ub are the lower and upper bounds. The Test Grade, *TG* is used to select the best local search method based on a predicated value in each generation.

The algorithm starts by initializing *M* solutions uniformly distributed over the search space of the problem being solved using the Simulated Orthogonal Array, $SOA_{M\times D}$ where *D* is the dimension of problem. After that, in order to generate new individuals using local search techniques, a sequence of step sizes based on backward and forward movements are employed to the original parameters. Three local search methods have been introduced in original MTS [21]. The first local search method uses the Eq. 1 to generate new offspring. If the new offspring is not better than the current individual, Eq. 2 is used. In the case the new offspring is better than the current one, the *TG* is updated for local search 1.

$$x_{new}^{j} = x_{i}^{j} - SR \tag{1}$$

$$x_{new}^{j} = x_{i}^{j} + 0.5 \times SR \tag{2}$$

The second local search resembles the equations of the first one except it focuses on one quarter of the dimensions of the current individual. The third local search generates three different individuals in an attempt to find new good solution if the first local searches failed to update their test grades. The three solutions are generated by increasing the current dimension by 0.1 and 0.2 and by decreasing it by 0.1. The test grade is updated if one of these three solutions is better than the current one.

III. PROPOSED ALGORITHM

In our hybrid approach, an improved Cultural Algorithm is used with a modified Multiple Trajectory Search. The following sub-sections presents the algorithm in details.

A. *Improved Cultrual Algorithm*

An enhanced version of Cultural Algorithm is used in our hybrid approach. In this version, a modified belief space is used in which three improved knowledge sources have been used to guide the evolutionary search. These three knowledge sources are: Topographical, Situational and Normative knowledge sources. In our approach, these knowledge sources as described below depict the archive of the finest knowledge which will be applied during the optimization process.

1) Topographical Knowledge source: is the heart of belief space in Cultural Algorithm in which the cell-based functional patterns is used [22]. In order to split the search landscape into divergent cells, the spatial characteristics are assembled and employed. To keep track of the best individual, every cell will have the responsibility. Furthermore the Influence function will imitate the best-cell to generate new good individuals. In the original implementation of Topographical knowledge, the *k*-d tree in used to implement its structure. In our approach, to effectively manage memory and time resources when solving real-world optimization problems of complex landscape, we set *k* to 2 where each node can only have two children. This modification saves extra resources and simplify the space-partitioning data structure to utilize the spatial characteristics in an effective manner. The update process of our topographical knowledge is presented in Fig. 2 where p_cb_h is the parent cell of the best individual.

```
If search is progressing
   If curr_cell is good ➔ Y = mutate(X)
   Else
       Choose another cell from the set 1 ≤ j ≤ s
       Y = mutate(p_cb_j)
   End
   Else
       If f(X) < f(p_cb_h)
           Y = mutate(X)
       Else
           Choose C_t from the upper level cells
           Y = mutate(p_cb_t)
       End
End
```

Fig. 2. Modified topographical knowledge Source

2) Situational Knowledge source: is responsible of accumulating the efficient exemplars during the evolutionary process. These exemplars will be followed by newly produced individuals in the Situational knowledge. As a result, the Situational knowledge guides the search toward best promising regions by mimicking the best exemplars in its structure. Fig. 3 shows how the update process works where the global best solution is represented as $<gbest_1, gbest_2 \ldots gbest_D>$ and *D* is the dimension.

```
While (i<= D)
   If x_i<gbest_i
      y_i= Rnd(x_i, gbest_i)
   ElseIf x_i>gbest_i
      y_i = Rnd(gbest_i, x_i)
   Else
      y_i = mutate(x_i)
   End
   i += 1
End
```

Fig. 3. Update process of Situational knowledge

3) Normative Knowledge source: is responsible of storing the behavior of individuals in a memory structure by dealing with promising parameter settings. Using this knowledge source, it is guaranteed to move in better ranges or remain in the current situation by just storing the acceptable behavior of generated individuals. Fig. 4 presents the update process of our Situational Knowledge source where lb_i, ub_i are the lower and upper bounds, x_i is the current solution, y_i is the generated offspring, *Rnd* is a uniform random number generator and mutate(x_i) is a Gaussian distribution generator with a mean of x_i.

```
Given current solution x_i with lower and upper bounds lb, ub
While (i<= D)
   If x_i ∉ (lb_i, ub_i)
       y_i=rnd(lb_i, ub_i)
   Else
       y_i = mutate(x_i)
   End
   i += 1
End
```

Fig. 4. Update process of Normative knowledge

B. *Modified Multiple Trajectory Search*

A cultural-based scheme is used to modify the classical multiple trajectory search. In this scheme, the knowledge sources of Cultural Algorithm is used instead of simulated orthogonal arrays to generate *M* initial solutions. This represents crucial change in original multiple trajectory search by utilizing the best knowledge to generate initial search agents to better guide search instead of starting from random points. Using shared knowledge for this task enables us to provide good starter point for multiple trajectory search by benefiting from the knowledge to generate better solutions. As explained earlier, the original multiple trajectory search uses three local search methods. When hybridizing with other algorithm, these search methods absorbs extra computations to generate one good solution especially when other algorithm is strong enough and uses some knowledge. To solve such issue, a new search method is introduced and used instead of those three original methods.

After generating *M* initial individuals using knowledge sources, the best individual x_{best} is selected to generate new solution using the following equation:

$$x^{j}_{new} = x^{j}_{i,best} + \delta \quad (3)$$

Where δ is the search range or step size of our search method which is denoted as δ in our implementation. It is generated as Eq. 4 shows. It uses the Euclidean distance between two individuals x_1 and x_2 multiples by rnd_{LRF} which is a linear reducing factor chosen randomly within the range [0.02,1].

$$\delta = D(x_1, x_2) \times rnd_{LRF} \tag{4}$$

C. *Hybrid Approach*

In order to increase search ability of Cultural Algorithm, a hybrid approach that combines the capabilities of an improved Cultural Algorithm and a modified Multiple Trajectory search. In this hybrid approach, an improved belief space consists of three modified knowledge sources have been used which explained in Section A. Due to their well-known performance these aforementioned three knowledge sources were chosen [22, 23]. In order to control communication between Cultural Algorithm and Multiple Trajectory Search, a quality function is employed. This quality function is responsible of choosing the knowledge sources that will be incorporating in generating *M* initial solutions for our modified Multiple Trajectory Search. The selection for knowledge source for this task is based on its success of the previous generations. Initially, the probability of selecting a knowledge source $p^t_{ks_i}$ at *t*=0 is set to $1/N$ where *N* is the number of used knowledge sources which is 3 in our case.

For each knowledge source ks_i, the number of successful individuals are reordered in an archive which is denoted by $n_s^{t+1}(ks_i)$ while the number of other individuals which are failed in optimization process is recorded in another archive as $n_f^{t+1}(ks_i)$. Based on the sizes of used archives, the probabilities of choosing the various knowledge sources for a specific individual are updated as it is stated in the following equation shows:

$$p^t(ks_i) = \frac{n_s^{t+1}(ks_i)}{n_s^{t+1}(ks_i) + n_f^{t+1}(ks_i)} \tag{5}$$

In our hybrid approach, the Cultural Algorithm is started by initializing a set of individuals in the population space which is uniformly distributed over search range of a problem. After that, in order to generate good solutions, the adapted belief space with the improved knowledge sources are applied which also got benefited by the shared knowledge. Next, the amended multiple trajectory search is performed to help generating new search argents that help guide search toward promising regions. The *M* initial solutions are generated using the three knowledge sources based on their successes probabilities as Eq. 5 presented. Next, our local search method inside Multiple Trajectory Search is used to generate new good solutions as Eq. 4 shows. x_1 and x_2 in our case represents the best individual of population space of our Cultural Algorithm and best individual of *M* generated individuals, x_{best} respectively. If new individual x_{new} is better than the best individual of Cultural Algorithm, the replacement is occurred. Otherwise, for the next generation the existing supreme individual will be preserved.

IV. EXPERIMENTS AND RESULTS

In this section, a set of real-world optimization problems are being used to validate the efficiency of the suggested algorithm. In the experiments that is described in this section, the results of several other well-known CA-algorithms as well as other state-of-the-art algorithms are being compared with the performance of the *i*CA-MTS. An Intel (R) Core i7 2720QM processor @ 2.20 GHz, and 8 GB RAM operating on Windows 7 professional is used to perform the experiments illustrated in this paper. Java 1.8 is used to implement our proposed algorithm. For each real-world problem, a total of 50 independent runs were performed. *NP* which denotes the population size is decided at 50 individuals. The number of *M* solutions is required by our adapted Multiple Trajectory Search is set to 5. The linear reducing factor rnd_{LRF} used in Eq. 4 is randomly chosen within the interval [0.02, 1].

When dealing with real-world problems, it is crucial to know the manner of handling constraints. The Constraint handling techniques that are proposed in the literature can be either techniques which preserve feasibility or penalty-based techniques. Moreover, it can be hybrid approaches of both techniques [24, 25]. An adaptive penalty-based technique based on our previous work is implemented in this suggested algorithm [10].

A. *Tension/Compression Spring*

The tension/compression spring considers one of the challenging mechanical design problem. The optimization task is to minimize the weight of a tension/compression spring and limits its outside diameter [26]. This problem is faced with several limitations which are minimum deflection, surge frequency, shear stress and restrictions on design variables. This problem consists of three design variables which are: the wire diameter $d\,(=x_1)$, the mean coil diameter $D\,(=x_2)$, and the number of active coils $N\,(=x_3)$. The schema of the spring is given in Fig. 5. The mathematical representation of this problem can be expressed as:

$$f(x) = x_1^2 x_2 x_3 + 2x_1^2 x_2,$$

subject to,

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0, \tag{6}$$

$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \le 0,$$

$$g_3(x)=1-\frac{140.45x_1}{x_2^2x_3}\le 0,$$

$$g_4(x)=\frac{x_1+x_2}{1.5}-1\le 0,$$

$$0.05\le x_1\le 2,\ \ 0.25\le x_2\le 1.3,\ \ 2\le x_3\le 15$$

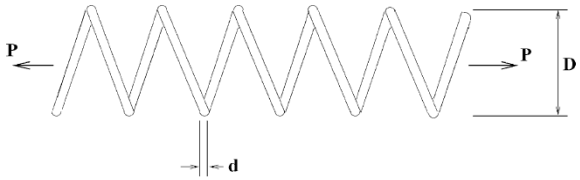


Fig. 5. Schema for the tension/compression string [26]

Table 1 presents the optimal design variables, constraints, optimal value, and required number of function evaluations (FEs) to reach the optimal value, for *i*CA-MTS and other reported values in the literature. Table 2 shows the statistical simulation results over 50 independent runs. The results show that the proposed algorithm is able to generate high quality solutions comparing to the state of the art methods. The best optimal value for *i*CA-MTS was 0.012665 using 4809 function evaluations which is better than other approaches.

Table 1

Comparison of the solution quality for tension/compression string design problem

Methods	Optimal design variables				
	x_1	x_2	x_3	f_{cost}	FEs
Coello and Montes [26]	0.051989	0.363965	10.890522	0.012681	N.A
Gao et al. [27]	0.055071	0.445656	7.913870	0.012989	10,000
Cagnina et al. [28]	0.051583	0.354190	11.438675	0.012665	24,000
Jaberipour & Khorram IPHS [29]	0.051860	0.360857	11.050339	0.012665798	200,000
Tomassetti [30]	0.051644	0.355632	11.35304	0.012665	10,000
He and Wang [31]	0.051728	0.357644	11.244543	0.0126747	N.A
Kaveh & Talatahari 2011 [32]	0.051432	0.35106	11.60979	0.0126385	4,000
Kaveh & Talatahari 2010 [33]	0.051744	0.35832	11.165704	0.126384	N.A
***i*CA-MTS (present study)**	0.051795	0.359264	11.14128	0.012665	4809

* N.A: Not Available

Table 2

Statistical analysis for the solution quality tests for the tension/compression string design problem

Methods	Best	Worst	Mean	Std
Coello and Montes [26]	0.012681	0.012973	0.0127420	5.9000E-05
Gao et al. [27]	0.012989	N.A	N.A	N.A
Cagnina et al. [28]	0.012665	N.A	N.A	N.A
Jaberipour & Khorram IPHS [29]	0.012665798	N.A	N.A	N.A
Tomassetti [30]	0.012665	N.A	N.A	N.A
He and Wang [31]	0.0126747	0.012730	0.012924	5.1985E-05
Kaveh & Talatahari 2011 [32]	0.0126385	0.0130125	0.0127504	3.948E-05
Kaveh & Talatahari 2010 [33]	0.126384	0.013626	0.012852	8.3564E-05
***i*CA-MTS (present study)**	0.012665	0.0129900	0.0127110	2.9097E-05

* N.A: Not Available

B. ***Welded Beam Design***

One of the popular practical engineering optimization design problem is the Welded/beam design [26]. In this problem, four design variables are introduced which are: $h(=x_1)$, $l(=x_2)$, $t(=x_3)$, and $b(=x_4)$. The structure of welded beam is shown in Fig. 6.

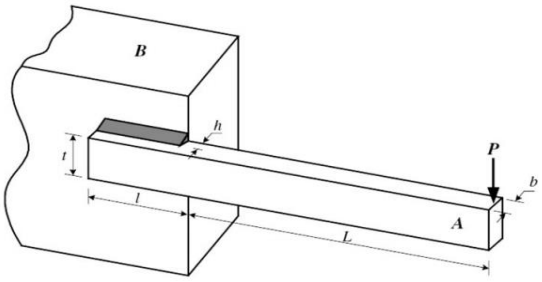


Fig. 6. Structure of the welded beam [26]

This structure consists of two beams *A* and *B* and a weld that is needed to clamp between them. The objective is to locate a feasible solution vector of dimensions *h*, *l*, *t*, and *b* to convey a certain load (*P*) while sustaining the minimum total fabrication cost. The objective function for this problem is mainly the total fabricating cost, comprised of the welding labor, set-up, and material costs. This problem can be formulated mathematically as follows:

$$f(x)=1.10471x_1^2x_2+0.04811x_3x_4(14.0+x_2)$$

subject to,

$$g_1(x)=\tau(x)-\tau_{\max}\le 0,$$

$$g_2(x)=\sigma(x)-\sigma_{\max}\le 0,$$

$$g_3(x)=x_1-x_4\le 0,$$

$$g_4(x)=0.10471x_1^2+0.04811x_3x_4(14.0+x_2)-5.0\le 0,$$

$$g_5(x)=0.125-x_1\le 0,$$

$$g_6(x)=\delta(x)-\delta_{\max}\le 0,$$

$$g_7(x)=P-P_c(x)\le 0$$

where:

$$\tau(X)=\sqrt{(\tau')^2+2\tau'\tau''\frac{x_2}{2R}+(\tau'')^2},\qquad (7)$$

$$\tau'=\frac{P}{\sqrt{2}x_1x_2},\ \tau''=\frac{MR}{J},$$

$$M=P(L+\frac{x_2}{2}),\ R=\sqrt{\frac{x_2^2}{4}+(\frac{x_1+x_3}{2})^2},$$

$$J=2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12}+(\frac{x_1+x_3}{2})^2\right]\right\},$$

$$\sigma(X)=\frac{6PL}{x_4x_3^2},\ \delta(X)=\frac{4PL^3}{Ex_3^3x_4},$$

Table 3
Comparison of the solution quality for the welded beam design problem

Methods	Optimal design variables					
	x_1	x_2	x_3	x_4	f_{cost}	*FEs*
Gao et al. [27]	0.299005	2.744191	7.502979	0.311244	2.0932	10,000
Tomassetti [30]	0.205729	3.470489	9. 036624	0.205730	1.7248	10,000
Jaberipour & Khorram IPHS [29]	0.205730	3.470490	9.036620	0.205730	1.7248	65,300
Coello &Montes [26]	0.205986	3.471328	9.020224	0.206480	1.728226	N.A
He and Wang [31]	0.202369	3.544214	9.047210	0.205723	1.728024	N.A
Kaveh & Talatahari 2011 [32]	0.207301	3.435699	9.041934	0.205714	1.723377	4,000
Kaveh & Talatahari 2010 [33]	0.205820	3.468109	9.038024	0.205723	1.724866	N.A
Gandomi et al. [28]	0.2015	3.562	9.0414	0.2057	1.7312065	50,000
*i*CA-MTS **(present study)**	0.205673	3.471676	9.036692	0.205729	1.7249	3,721

$$P_c(X)=\frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1-\frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$P=6,000\,\text{lb},\ L=14\,\text{in},\ E=30\times10^6\ \text{psi},\ G=12\times10^6\ \text{psi}$

$\tau_{max}=30{,}600\ \text{psi},\ \sigma_{max}=30{,}000,\ \delta_{max}=0.25\,\text{in}.$

The comparison of the solution quality is shown in Table 3 including the results for other state-of-the-art algorithms and the statistical simulation results over 50 independent runs are summarized in Table 4. The results show that the proposed algorithm generates a function value equals to 1.7249 using 3721 function evaluations, which is better than other approaches.

Table 4
Statistical results of differnt methods for the welded beam design problem

Methods	Best	Worst	Mean	Std
Gao et al. [27]	2.0932	N.A	N.A	N.A
Tomassetti [30]	1.7248	N.A	N.A	N.A
Jaberipour & Khorram IPHS [29]	1.7248	N.A	N.A	N.A
Coello &Montes [26]	1.728226	1.792654	1.993408	0.074713
He and Wang [31]	1.728024	1.748831	1.782143	0.012926
Kaveh & Talatahari 2011 [32]	1.723377	1.762567	1.743454	0.007356
Kaveh & Talatahari 2010 [33]	1.724866	1.759479	1.739654	0.008064
Gandomi et al. [28]	1.7312065	2.3455793	1.8786560	0.2677989
*i*CA-MTS **(present study)**	1.7249	1.7596205	1.7368892	3.328E-04

* N.A: Not Available

C. *Gear train design*

The Gear train design is a discrete engineering optimization problem introduced by Sandgran [34]. The gear ratio of a compound gear train is the main objective that can be expressed mathematically as shown below:

$$Gear\ ratio=\frac{\text{Angular velocity of the output shaft}}{\text{Angular velocity of the input shaft}}$$

The design of this problem is presented in Fig. 7 which has four integer variables denoted as T_i which defines the number of teeth in the of i^{th} gear wheel. The objective function as expressed belwo requires the teeth numbers of wheel that generate a gear ratio that reaches to 1/6.931.

$$f(T_a,T_b,T_d,T_f)=\left(\frac{1}{6.931}-\frac{T_bT_d}{T_aT_f}\right) \quad (8)$$

Where T_a,T_b,T_d,T_f are integer variables between 12 and 60.

Tables 5 and 6 show the best results for the best objective values obtained by *i*CA-MTS algorithm over 50 independent runs and along with those for other algorithms from the literature. The results show that the proposed algorithm was able to generate a gear ratio equals to 0.1442 using less number of function evaluations compared to other approaches.

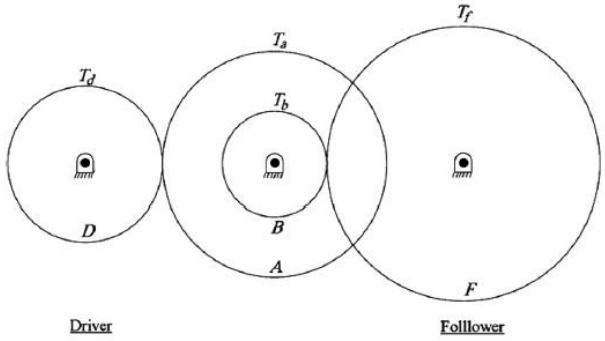


Fig. 7. Structure of the gear train [34]

Table 5
Optimal results of differnt methods for the gear train design problem

Methods	T_a	T_b	T_d	T_f
Deb & Goyal [34]	33	14	17	50
Loh & Papalambros [35]	42	16	19	50
Parsopoulos & Vrahatis [36]	43	16	19	49
Gandomi et al. [37]	43	16	19	49
Gandomi et al.[38]	49	19	16	43
***i*CA-MTS (present study)**	43	19	16	49

* N.A: Not Available

Table 6
Optimal results of differnt methods for the gear train design problem

Methods	*Gear ratio*	f_{min}	*FEs*
Deb & Goyal [34]	0.1442	1.362E-09	N.A
Loh & Papalambros [35]	0.1447	0.23E-06	N.A
Parsopoulos & Vrahatis [36]	0.1442	2.701E-12	100,000
Gandomi et al. [37]	0.1442	2.701E-12	5,000
Gandomi et al.[38]	0.1442	2.701E-12	2,000
***i*CA-MTS (present study)**	0.1442	2.701E-12	1,500

* N.A: Not Available

V. CONCLUSION

Solving real-life optimization problems is an intriguing task due to the fact that many real-life applications can be formulated as optimization problems. The evolutionary algorithms are bio-inspired algorithms which proofs its efficiency a powerful optimization tool to solve diverse set of optimization problems. Among these algorithms, the cultural algorithm is the only evolutionary algorithm which uses the knowledge source explicitly to guide the evolutionary search. This paper introduces an improved cultural algorithm with a multiple trajectory search to solve interesting real-life optimization problems. The algorithm uses three knowledge sources which are: situational knowledge, normative knowledge, and topographic knowledge with a good quality function to control the interaction and number of followers between them. The proposed algorithm namely, *i*CA-MTS, is tested on engineering problems which are: tension/compression string, welded beam and gear train design problems. The simulation results show that by comparing other state-of-the-art algorithms, the proposed algorithm is able to generate high quality solutions.

References

[1] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, New York, 2001

[2] J.H. Holland, Adaption in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975

[3] J. Kennedy, R. Eberhart, "Particle swarm optimization," Proc. IEEE ICNN, Vol. 4, pp. 1942–1948, 1995

[4] R. Storn, K. V. Price, Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI, Berkeley, CA, Tech. Rep. TR-95-012. [Online]. Available: http://http.icsi.berkeley.edu/~storn/litera.html

[5] M. Dorigoet, L.M. Gambardella, Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Trans. Evol. Comput., volume 1, numéro 1, pages 53-66, 1997

[6] Z. Kobti, R.G. Reynolds, T. Kohler, A multi-agent simulation using cultural algorithms: The effect of culture on the resilience of social systems, in: Proc. IEEE Congr. Evol. Comput., 2003, pp. 1988–1995

[7] J.A. Vrugt; B.A. Robinson; J.M. Hyman, "Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces," IEEE Trans. Evol. Comput., vol.13, no.2, pp.243,259, April 2009

[8] Y. Wang, H.-X. Li, T. Huang, L.Li ,"Differential evolution based on covariance matrix learning andbimodal distribution parameter setting," Applied Soft Computing 18 (2014) 232–247

[9] J.M. García-Nieto, E. Alba ,"Hybrid PSO6 for Hard Continuous Optimization," Soft Computing, July 2015, Volume 19, Issue 7, pp 1843-1861

[10] M.Z. Ali, N.H. Awad, A novel class of niche hybrid cultural algorithms for continuous engineering optimization, Inf. Sci. 267 (2014) 158–190

[11] M.Z. Ali, N.H. Awad, R.G. Reynolds, Balancing Search Direction in Cultural Algorithm for Enhanced Global Numerical Optimization, In Proc. IEEE Symposium on Swarm Intelligence (SIS), Dec. 2014, pp. 336-342, Orlando, Florida

[12] M.Z. Ali, N.H. Awad, R.G. Reynolds, Hybrid Niche Cultural Algorithm for Numerical Global Optimization, In Proc. IEEE Congress of Evolutionary Computations, June 2013., pp. 309-316, Cancun, Mexico

[13] Y.-N. Guo, J. Cheng, Y.-y. Cao, Y. Lin, A novel multi-population cultural algorithm adopting knowledge migration, Soft Comput., 15 (2011) 897-905

[14] T.T. Nguyen, X. Yao: An Experimental Study of Hybridizing Cultural Algorithms and Local Search. Int. J. of Neural Systems, 18 1-18 (2008)

[15] L.S. Coelho, V.C. Mariani, An efficient particle swarm optimization approach based on cultural algorithm applied to mechanical design, In: Proc. IEEE Congr. Evol. Comput., Canada, pp. 1099-1104 (2006)

[16] L. Wang, C. Cao, Z. Xu, X. Gu, "An Improved Particle Swarm Algorithm Based on Cultural Algorithm for Constrained Optimization", Adv in Intelligent and Soft Comp, 135 (2012) 453-460

[17] R.L. Becerra, C.A.C. Coello: Cultured Differential Evolution for Constrained Optimization. Comput. Meth. Appl. Mech. Eng. 195, 4303–4322 (2006)

[18] W. Wang, T. Li, "Improved cultural algorithms for job shop scheduling problem", INT. J. IND. ENG-THEORY, 18 (2011) 4

[19] A. Haikal, M. El-Hosseni, "Modified cultural-based genetic algorithm for process optimization", Ain Shams Eng. J., 2 (2011) 173-182

[20] B. Peng, R.G. Reynolds, Cultural algorithms: Knowledge learning in dynamic environments, in: Proc. IEEE Congr. Evol. Comput., 2004, pp. 1751–1758

[21] L-Y. Tseng, C.Chen, Multiple trajectory search for large global optimization, Proc. IEEE Congr. Evol. Comput. CEC, pp. 3052-3059, 2008

[22] B. Peng. Knowledge and population swarms in cultural algorithms for dynamic environments. PhD thesis, Detroit, MI, USA, 2005

[23] .G. Reynolds, B. Peng, Cultural algorithms: Computational modeling of how cultures learn to solve problems: An engineering example, Cybern. Syst., 36 (2005) 753–771

[24] C.A.C. Coello, Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art, COMPUT. METHOD APPL. M., 191 (2002) 1245-1287

[25] K. Deb, An efficient constraint handling method for genetic algorithms, COMPUT. METHOD APPL. M., 186 (2002) 311–338

[26] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advan. Eng. Inform. 16 (2002) 193–203.

[27] X.Z.Gao,X.Wang,T.Jokinen,S.J.Ovaska,A.Arkkio,K.Zenger,Ahybridoptimizationmethodforwindgeneratordesign,Int.JInnov.Comput.8(6)(2012)4347–4373.

[28] L.C. Cagnina, S.C. Esquivel, C.A.C. Coello, Solving constrained optimization problems with a hybrid particle swarm optimization algorithm, Eng. Optim. 43 (2011) 843–866.

[29] M. Jaberipour, E. Khorram, Two improved harmony search algorithms for solving engineerin goptimization problems, Commun. Nonlinear.Sci.Numer.Simulat. 15(2010)3316–3331.

[30] G. Tomassetti, A cost-effective algorithm for the solution of engineering problems with particle swarm optimization, Eng.Optimiz.42(2010)471–495.

[31] Q. He,L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng.Appl.Artif.Intell.20(2007)89–99.

[32] A. Kaveh,S. Talatahari, Hybrid charged system search and particle swarm optimization for engineering design problems, Int.J.Comput.AidedEng.Softw.28(2011)423–440.

[33] A. Kaveh, S. Talatahari, A novel heuristic optimization method: chargedsystemsearch, ActaMechanica213(2010)267–289.

[34] K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) forengineeringdesign, Comput.Sci.Inform.26(1996)30–45.

[35] H.T. Loh, P.Y. Papalambros, A sequential linearization approach for solving mixed-discretenonl inear design optimization problems, J.Mech.Des.113(1991)325–334.

[36] K.E. Parsopoulos, M.N. Vrahatis, Unified particle swarm optimization for solving constrained engineering optimization problems, Advances innaturalcom-putation,Springer-Verlag,2005,pp.582–591.

[37] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng.Comput.29(2013)17–35.

[38] A.H. Gandomi,G.J. Yun, X-S Yang, S. Talatahari, Chaos-enhanced accelerated particle swarm optimization, Commun. Nonlinear. Sci. Numer. Simulat. 18(2013)327–340.

Construction of heuristic for protein structure optimization using deep reinforcement learning

Rok Hribar[1,2], Jurij Šilc[1], and Gregor Papa[1,2]

[1] Jožef Stefan Institute, Ljubljana, Slovenia,
rok.hribar@ijs.si, jurij.silc@ijs.si, gregor.papa@ijs.si
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Abstract. Deep neural networks are constructed that are able to partially solve a protein structure optimization problem. The networks are trained using reinforcement learning approach so that free energy of predicted protein structure is minimized. Free energy of a protein structure is calculated using generalized three-dimensional AB off-lattice protein model. This methodology can be applied to other classes of optimization problems and represents a step toward automatic heuristic construction using deep neural networks. Trained networks can be used to construct better initial populations for optimization. It is shown that differential evolution applied to protein structure optimization problem converges to better solutions when initial population is constructed in this way.

Keywords: protein folding, heuristic, deep learning, differential evolution

1 Introduction

Prediction of protein structure from the sequence of its residues is a hard optimization problem. All proteins are endowed with a primary structure consisting of the chain of amino acids. Folding of this chain results into so-called 3D protein structure. The biological functional role of the protein is strictly dependent on the protein 3D structure. Knowledge of a proteins structure provides insight into how it can interact with other proteins, DNA/RNA, and small molecules. It are these interactions which define the proteins function and biological role in an organism. Thus, protein structure and structural feature prediction is a fundamental area of computational biology. Its importance is exacerbated by large amounts of sequence data coming from genomics projects and the fact that experimentally determining protein structures remains expensive and time consuming [1].

Over the last decades a lot of effort have been invested in reducing the computational cost of calculating the 3D structures of proteins. One way to decrease the computational cost is the introduction of approximate models for the calculation of protein's free energy which is minimal for appropriate 3D structure. Because the computation of free energy is less costly, optimization that finds the right structure is also less costly to preform. Examples of such approximate

models include models using a cubic lattice [2] and AB type models [3]. Another way to speed up the optimization process is to make optimization more efficient, so that less free energy evaluations are needed. This resulted in development of heuristics that are tailored specifically for this optimization problem. Since protein folding process is in nature guided only by the physical laws, optimization methods were devised that include principles from statistical physics. Heuristics from this category include annealing contour Monte Carlo [4] and conformational space annealing [5]. Another approach to developing a specialized optimization algorithm is to modify known metaheuristics such as artificial bee colony [6] or evolutionary algorithm [7].

A different approach to prediction of protein 3D structure is the use of machine learning. Here the prediction of 3D structure is based only on features directly calculated from the sequence of amino acids. There is no optimization performed during prediction. The structure is calculated simply by applying the model. Optimization is used only during the model training, when appropriate model is searched for. Currently, deep neural networks (DNNs) are the most widely used models for this problem. Properly trained DNNs are very successful at predicting protein's secondary structure ($\approx$ 80% accuracy) [8] and its disordered regions ($\approx$ 90% accuracy) [9]. However, full 3D structure prediction is much less accurate ($\approx$ 20% accuracy) [10]. DNN models are usually trained using supervised learning where experimentally acquired 3D structures of proteins are used as training examples. Advantage of this approach is that protein's free energy does not need to be calculated. But on the other hand, by using only experimental data one is limited to possibly insufficient amount of training examples to properly train DNN.

In this paper a different approach to DNN training is presented and used in which explicit training examples are not needed. Instead, the free energy of a protein is used to provide information about the quality of predicted solutions. This is possible because this problem can be interpreted as an optimization problem or a prediction problem. This allows the combination of both views to generate a new method for addressing the protein structure problem. In this regard such methodology can be applied to any optimization problem to generate DNNs able to predict a solution of an optimization problem. In other words, given a class of optimization problems one can construct a DNN that represents extremely fast heuristic specially designed for this class of optimization problems. This is a step toward automatic heuristic generation.

2 Deep neural network as an optimization algorithm

Optimization problems are often solved using approximate algorithms (heuristics) that are tailored for a specific class of problems. For example, there are specific heuristics that work well for vehicle routing [11], production scheduling [12], protein folding [13] and so on. Heuristics are especially useful if similar problems need to be solved over and over again. In such cases it is sensible to

develop specialized optimization procedures which are optimized for that specific class of problems.

In this section a methodology is presented where DNNs are trained in a way that they are able to approximately solve an optimization problem that belongs to a given class of problems. Such class of optimization problems can be represented with a fitness function f with two inputs.

$$\begin{aligned} f &: \mathcal{S}, \mathcal{X} \to \mathbb{R} \\ f_s(x) &= \min. \end{aligned} \tag{1}$$

Set $\mathcal{S}$ holds all possible optimization problems in the class, while set $\mathcal{X}$ holds all possible candidate solutions for that class of problems. For example, in case f represents a class of production scheduling problems, s encodes the orders that need to be fulfilled and x encodes the production schedule.

Given function f, it is possible to define a function g that takes a problem specification s as an input and returns the position x_s^{optimal} where function f_s has a global minimum.

$$\begin{aligned} g &: \mathcal{S} \to \mathcal{X} \\ g(s) &= \arg\min_{x \in \mathcal{X}} f_s(x) = x_s^{\text{optimal}} \end{aligned} \tag{2}$$

In other words, $g(s)$ is a solution of optimization problem $f_s(x) = \min$. Calculation of function g is in general intractable. But it might be possible to find a model that approximates g to some degree. One aim of this paper is to find out whether a trained DNN is able to approximate g. It is important to note that the input and output of DNN are traditionally floating point numbers. Therefore, s and x should be encoded as vectors of floating point numbers. Even for discrete s and x it is usually possible to find such an encoding.

It is known that a neural network can approximate arbitrary function to an arbitrary precision [14]. So g can be approximated well using DNN, however it is unknown how large such a network should be and whether it is possible to find it using known training techniques. If DNN could be trained to approximate g, such DNN can preform partial optimization extremely quickly. While DNN training is known to be resource intensive, prediction is usually not.

Training DNN to approximate g is also an optimization problem, however optimization landscape of DNN training is not similar to f_s landscape. DNN parameters encode a strategy for predicting x_s^{optimal} from s, so optimization is not performed on a single problem encoded by s, but for all possible s at once. Also DNN optimization landscape has particular properties, like the fact that saddle points are exponentially more common compared to local minima [15]. Therefore, a suitable optimization method that takes those specific properties into account should be used for training them. Currently, stochastic gradient descent (SGD) is the prevalent and very successful approach to DNN training [16].

2.1 Supervised learning approach

DNN can be trained to approximate g using supervised learning. Let $\hat{g}(s)$ be the output of DNN when s is its input. In supervised learning pairs $(s_i, x_{s_i}^{\text{optimal}})$ are provided and DNN error is minimized using SGD so that

$$J = \sum_i \left\| \hat{g}(s_i) - x_{s_i}^{\text{optimal}} \right\| = \min. \tag{3}$$

Since x_s^{optimal} is in general unknown, its best approximation has to be used which results to nonideal DNN model. So $x_{s_i}^{\text{optimal}}$ need to acquired using external optimization algorithms. In order to prevent DNN overfitting it is necessary to provide a large amount of training examples, i.e. much more than the number of DNN parameters. Therefore, such approach is extremely resource intensive.

2.2 Reinforcement learning approach

Another approach to DNN training is reinforcement learning. In this case functions f_s are used to calculate the error of DNN. DNN is trained so that

$$E = \sum_i f_{s_i}(\hat{g}(s_i)) = \min. \tag{4}$$

In reinforcement learning terminology, E can be understood as a penalty that needs to be minimized. In this case $x_{s_i}^{\text{optimal}}$ are not needed and so no external optimization is required. The downside is that SGD can not be applied as simply as with supervised learning. Error function J from equation (3) can be easily differentiated with respect to DNN parameters using backpropagation. But penalty E from equation (4) also includes application of f_s. This makes the calculation of the gradient difficult and different methods have been introduced by deep reinforcement learning community to mend this problem.

In this paper an adapted version of deterministic policy gradient method [17] is used. This method uses a differentiable model called a critic that approximates function f from equation (4). The derivative of E can then be approximately calculated using the derivative of the critic by applying the chain rule. Our adaptation of this method is to not model f with a critic but instead use f directly. In order to calculate derivative of E in this scope, the derivative $\nabla_x f_s(x)$ is required.

Fortunately, derivation of f_s can also be preformed using backpropagation principles, i.e. applying chain rule coupled with dynamic programming. There are good libraries that can preform such automatic differentiation, for example `theano`, `TensorFlow` and `CNTK`. In this paper `theano` was used to write expressions for the calculation of E. These expressions are then transformed to a computational graph for calculation of E which can be used to build computational graph for gradient calculation ∇E using the chain rule. In this respect procedure is returned for analytical gradient calculation ∇E without any assistance from the user. Computational graphs for E and ∇E calculation can be

compiled to `C++`, `CUDA` or `OpenCL` which brings multi processor support and can easily be accelerated on GPGPU or even FPGA [18]. Therefore, use of `theano` is beneficial even if just calculation of E is needed.

Therefore, by using `theano` DNN can be trained using SGD so that E from equation (4) is minimized. Great advantage of this approach is an unlimited amount of training examples s_i which can be drawn from desired distribution over s_i. This generation of training examples is extremely cheap compared to supervised learning approach where training examples need to be generated by optimization algorithm or acquired by experimental measurement.

3 Generalized three-dimensional AB off-lattice protein model

AB off-lattice model has been widely used to describe the protein secondary structure folding process for decades [3]. The off-lattice protein model was initially developed to consider 2D folding problems and was extended to deal with 3D scenarios where additional torsional energy contributions of each bond are taken into account [19]. According to the AB off-lattice model, the main driving forces that contribute to protein structure formulation are the hydrophilic and hydrophobic interactions.

The protein chain is modeled as a vector s where each component s_i specifies the hydrophilicity of amino acid at the site i. The distance of two neighboring amino acids is set to one ($d_{i,i+1}$=1). Under this model free energy G is calculated as

$$G(u,d) = \frac{1}{4}\sum_{i=1}^{n-2}(1 - u_i \cdot u_{i+1}) + 4\sum_{i=1}^{n-2}\sum_{j=i+2}^{n}\left(d_{ij}^{-12} - C(s_i,s_j)d_{ij}^{-6}\right), \quad (5)$$

where u_i is a vector from amino acid on site i to amino acid on site $i+1$ and d_{ij} is a distance between amino acids on site i and j (see Fig. 1). The interaction between two amino acids is specified by a function

$$C(s_i,s_j) = \frac{1}{8}\left(1 + s_i + s_j + 5s_is_j\right). \quad (6)$$

Structure of a protein of length n can be encoded using angles θ_i and φ_i that tell how vectors u_i are oriented in space (see Fig. 2). Therefore, a protein structure of length n is fully determined by

$$x = (\theta_2, \ldots, \theta_{n-1}, \varphi_3, \ldots, \varphi_{n-1}). \quad (7)$$

Use of this encoding reduces the dimensionality of search space and allows us to automatically fullfill the constraint $\|u_i\| = 1$. The values u_i and d_{ij} that are needed for free energy calculation can be calculated from x in the following way

$$u_i = (\cos\theta_i \sin\varphi_i, \sin\theta_i \sin\varphi_i, \cos\varphi_i) \quad (8)$$

$$r_{i+1} = r_i + u_i \quad (9)$$

$$d_{ij} = \|r_i - r_j\|. \quad (10)$$

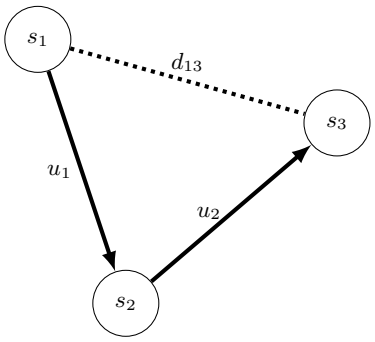


Fig. 1: Visualization of direction vectors u_i and distances d_{ij} on a protein with three amino acids.

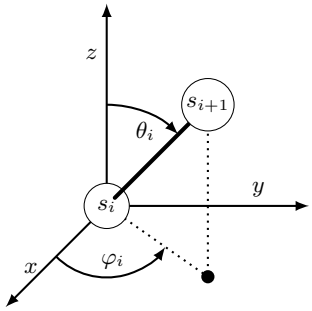


Fig. 2: Visualization of angles θ_i and φ_i from equation (7) that determine the direction of vectors u_i.

The first two amino acids in a sequence are fixed to specific coordinates and the third one is constrained to xy-plane.

$$r_1 = (0, 0, 0) \tag{11}$$

$$r_2 = (1, 0, 0) \tag{12}$$

$$\varphi_2 = \frac{\pi}{2} \tag{13}$$

By this choice, rotational symmetry of the model is eliminated.

In protein structure optimization problem we want to find a structure of a protein that minimizes the free energy G. Therefore, given a protein defined with s, we want to find x that determines the structure of that protein. So the problem class is defined by a function

$$f_s(x) = G(u(x), d(x)) = \min. \tag{14}$$

Traditionally $s_i = \pm 1$, where hydrophobic amino acid has $s_i = -1$ and hydrophilic $s_i = 1$. In this regard quantity s_i tells how hydrophilic an amino acid is. However, this paper uses a generalization of this model where $s_i \in \mathbb{R}$. One reason for this choice is the fact that amino acids in nature are not hydrophilic to the same degree [20]. Some may attract or repel water more than others. Also hydrophilicity changes with temperature [21] which allows one to use this generalized model to study how protein structure changes with temperature. Use of generalized model is also beneficial with regard to DNN training because this brings a richer set of training examples and makes the training landscape smoother.

4 Experiments

In this section DNN training procedure using reinforcement learning is presented and how solutions predicted by DNNs were used as initial population of differential evolution (DE) algorithm. A variant of SGD called Adam [22] was used

and gradient of E was used to guide the training. The calculation of ∇E was calculated using `theano` library.

$$E = \sum_{i=1}^{B} f_{s_i}(\hat{g}(s_i)) \tag{15}$$

In each step of Adam algorithm a batch of proteins s_i of random length was randomly selected. Distribution over length was uniform and over hydrophilicity a mixture of two Gaussians with mean at ± 1 and standard deviation 0.15. Based on the selected s_i calculation of E and ∇E was done by `theano`. The number of sampled proteins for E and ∇E calculation is called a batch size B. The training is more stochastic if B is low and becomes more deterministic if B is high. By experimenting with different batch sizes a good balance between speed and accuracy was found at $B = 512$.

Stated more informally, in each step, DNN tries to solve 512 random protein optimization problems and gets updated in direction that would improve its solving capabilities for those 512 proteins. Because DNN gets a different batch of random proteins in each step, it converges to a state that is able to solve all protein optimization problems equally well. Picking training batches randomly also ensures that DNN can not overfit since duplicates in the training data are extremely unlikely. Therefore, the training error of DNN is not a biased estimate of its accuracy and validation set is not needed.

A DNN structure was chosen that can take proteins with up to $n = 100$ amino acids. To allow prediction on smaller proteins zero padding was used. Example of small scale DNN is shown in Fig. 3. DNNs with different number of hidden layers was trained in order to quantify how DNN depth influences the accuracy. In all cases the width of hidden layers was chosen to be $2n = 200$. Rectified linear units were used as activation functions on hidden layers and tanh on the output layer to ensure that $\theta_i, \varphi_i \in [-\pi, \pi]$.

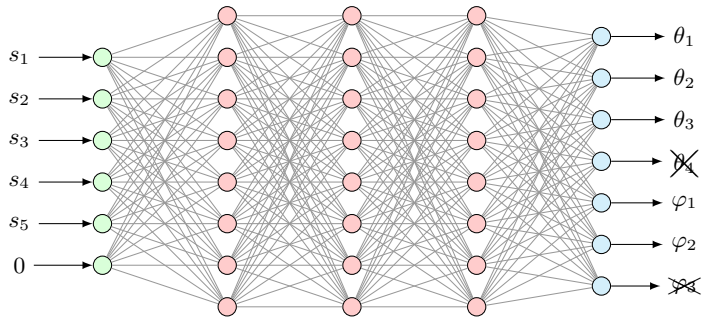


Fig. 3: Small scale example of how DNN receives the protein specification and how its output is interpreted. If a protein is shorter than DNN input layer, zero is placed on sites where there are no amino acids. In this case some angles from the output are discarded (crossed out outputs).

Before training, initial DNN weights need to be set. Initial weights were generated randomly using Glorot initialization [23]. However, if the magnitude of initial weights is to large, it can happen that initial DNN predicts very densely packed structures which causes a very large gradient due to d_{ij}^{-12} repulsive term in free energy. This causes an unstable gradient descent. Therefore, a prefactor w_i was added to initialization and DNN models with $w_1 = 0.1$ and $w_2 = 1.0$ was trained.

To avoid unstable gradient descent the predictions of initial DNN should be unfolded protein structures. This, however, produces another problem. The first summand in equation (5) forces proteins to be unfolded which means that unfolded structure is a local minimum that is common to all proteins. To avoid getting stuck in this common local minimum the first summand in equation (5) was simply not included in the calculation of free energy at the beginning of training. When DNN predicted structures began to fold, the previously ignored summand was gradually added during training. In the last stage of training the full version of free energy was used.

During DNN training it can happen that for some s_i in the batch DNN predicts a structure where two amino acids are very close to each other. A repulsive interaction causes the gradient ∇E to be very large for the entire batch. In the next step of gradient descent the DNN is thrown away from a possibly good region. Such events might be rare, but can severely disturb the progress of training. To mitigate the effect of such events, gradient norm clipping can be used. In other words, if the gradient length exceeds a given threshold, the gradient is clipped so that its length is equal to the threshold.

Solutions predicted by DNNs might be a good initial population for optimization. To test whether this is true protein structure optimization using DE was implemented. DE was shown to be the best known optimization method for this problem [7]. Specifications of implemented DE algorithm was taken from [7], but without parameter control. DE type was `best/1/bin`, population size was 100, mutation with dithering was employed with mutation constant taken between 0.1 and 1 and recombination constant was 0.9. DE was run 30 times for three proteins found in nature (1CB3, 1CRN and 2EWH) with random initial population and with initial population where 50% of candidates were predicted by DNNs.

5 Results

To measure how good a candidate solution is, we use free energy G of the protein. In case of comparing solutions gotten by DE, this is a sound measure from the point of view of statistical physics. That is, the protein is most likely to be in states with low free energy. In ideal case one could check if the solution is equivalent to the native structure, but because global minima of this protein model are unknown this is not possible. To evaluate the performance of DNN, training error is used. It is equivalent to validation error and defined as a mean of free energy values predicted by DNN for a batch of random proteins. The

variance of this error measure is very low because the batch size $B = 512$ is large.

The training error of DNNs depends of the number of layers. This dependence is usually monotonically decreasing [16], but for this problem this is not the case (see Fig. 4). This can be attributed to the sensitivity of training to selection of initial DNN weights. DNNs were initialized using random matrices. Therefore, the magnitude of DNN output is exponentially dependent of the number of layers. Since initial weights have small components ($\ll 1$), this means that initial DNNs with small number of layers predict very folded structures, while initial DNNs with high number of layers predict practically straight structures. Fig. 4 shows that initial DNN predictions should not be very folded nor very straight. Best models are somewhere in between.

The most accurate DNN models have three layers. Given that DNNs are able to partially solve an optimization problem this is a surprisingly shallow DNN architecture. Free energy of structures predicted by DNNs and by DE is shown in Table 1. It was found that gradient norm clipping is very beneficial for DNN training. Fig. 5 shows the progress of DNN training with and without the use of gradient norm clipping. Occurrence of very high gradients is rare, however they substantially alter the progress of DNN training.

Using structures predicted by DNN in initial population of DE was found to be beneficial. When using predicted initial population, DE converges to lower values of free energy (see Fig. 6). But the convergence is slightly less rapid for predicted population which could indicate that the population is actually more diverse. On the other hand, DE progresses are less dispersed among runs which means that less runs are needed to find a satisfactory solution. In case of 1CB3 protein the predicted structures are in fact so good that all DE runs converge to the best known solution in just 100 generations. For larger proteins the predicted solution are not as good, however the DE performs considerably better when using the predicted population.

6 Conclusion and future work

In this paper it is shown that deep neural networks can be trained to partially solve optimization problems belonging to a given class. The networks can be successfully trained using reinforcement learning method by knowing only the fitness function of the class of optimization problems. This is shown for the class of protein structure optimization problems. The predicted solutions were found to be good initial points for further optimization. Such trained networks can be used to acquire moderately good solutions of optimization problem when solution is needed very quickly. Therefore, the method is suitable for optimization problems that need to be solved repeatedly and is a step towards automatic heuristic construction.

For future work it should be possible to extend the method to combinatorial optimization problem where unified methodology should be further developed. The procedure of finding the best architecture of deep neural network could be

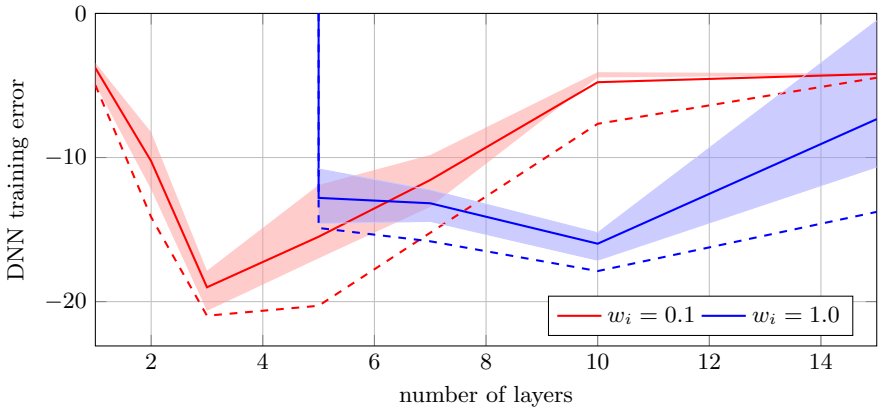


Fig. 4: Training error of DNNs with respect to the number of hidden layers for two different magnitudes of initial DNN weights. Full line is the mean error of models, shaded area shows the range of error for central 66% of the models and the dashed line is the error of the best model.

Table 1: Free energy calculated for three proteins found in nature by using structures predicted by DNN and by DE.

protein	length	best DNN	mean DE	best DE
1CB3	13	−4.6235	−2.1513	−6.7700
1CRN	46	−42.765	−64.948	−79.906
2EWH	98	−65.163	−148.32	−170.47

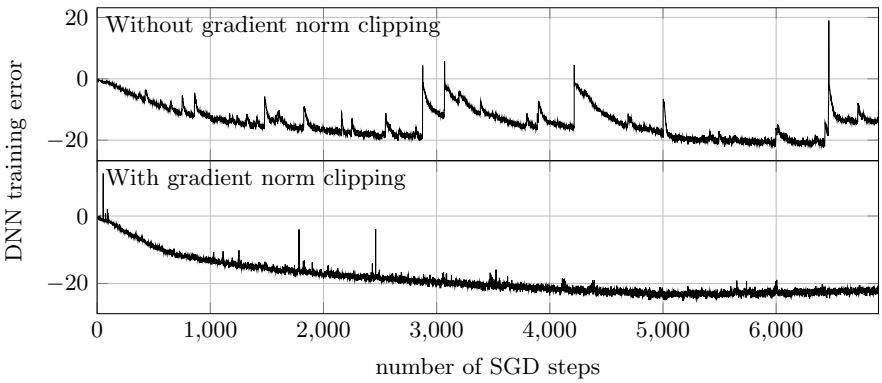


Fig. 5: DNN error during training via SGD. The upper plot shows the progress of usual SGD procedure, while the lower plot shows SGD progress when gradient norm has a predefined upper bound by using gradient norm clipping.

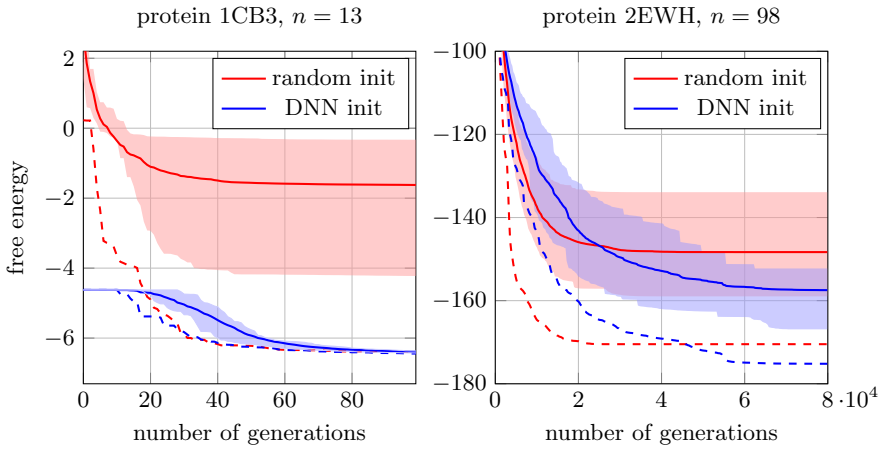


Fig. 6: Progress of 30 runs of DE by using a random initial population or initial population partially populated with solutions predicted by DNNs. Full line is the mean over all runs, shaded area shows the range of central 66% of runs and the dashed line shows the best run.

more automated so that depth and width of the network is automatically found. The same goes for the training specification. Another opportunity for future work is to combine supervised learning approach with reinforcement learning approach so that the training is guided by both approaches simultaneously.

Acknowledgments

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0098 and PR-07606).

References

1. Cheng, J., Randall, A.Z., Sweredoski, M.J., Baldi, P.: SCRATCH: a protein structure and structural feature prediction server. Nucleic acids research **33**(suppl_2) (2005) W72–W76
2. Bošković, B., Brest, J.: Genetic algorithm with advanced mechanisms applied to the protein structure prediction in a hydrophobic-polar model and cubic lattice. Applied Soft Computing **45** (2016) 61–70
3. Stillinger, F.H., Head-Gordon, T., Hirshfeld, C.L.: Toy model for protein folding. Physical review E **48**(2) (1993) 1469
4. Liang, F.: Annealing contour Monte Carlo algorithm for structure optimization in an off-lattice protein model. The Journal of chemical physics **120**(14) (2004) 6756–6763

5. Kim, S.Y., Lee, S.B., Lee, J.: Structure optimization by conformational space annealing in an off-lattice protein model. Physical review E **72**(1) (2005) 011916
6. Li, B., Lin, M., Liu, Q., Li, Y., Zhou, C.: Protein folding optimization based on 3D off-lattice model via an improved artificial bee colony algorithm. Journal of molecular modeling **21**(10) (2015) 261
7. Bošković, B., Brest, J.: Differential evolution for protein folding optimization based on a three-dimensional AB off-lattice model. Journal of molecular modeling **22**(10) (2016) 252
8. Pollastri, G., Przybylski, D., Rost, B., Baldi, P.: Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins: Structure, Function, and Bioinformatics **47**(2) (2002) 228–235
9. Cheng, J., Sweredoski, M.J., Baldi, P.: Accurate prediction of protein disordered regions by mining protein structure data. Data mining and knowledge discovery **11**(3) (2005) 213–222
10. Di Lena, P., Nagata, K., Baldi, P.: Deep architectures for protein contact map prediction. Bioinformatics **28**(19) (2012) 2449–2457
11. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European Journal of Operational Research **231**(1) (2013) 1–21
12. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Transactions on Evolutionary Computation **20**(1) (2016) 110–124
13. Perez, A., MacCallum, J., Dill, K.A.: Using physics and heuristics in protein structure prediction. Biophysical Journal **108**(2) (2015) 210a
14. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural networks **4**(2) (1991) 251–257
15. Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: Advances in neural information processing systems. (2014) 2933–2941
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016) `http://www.deeplearningbook.org`.
17. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: ICML. (2014)
18. Ling, A.C., Aydonat, U., O'Connell, S., Capalija, D., Chiu, G.R.: Creating high performance applications with Intel's FPGA OpenCL SDK. In: Proceedings of the 5th International Workshop on OpenCL, ACM (2017) 11
19. Bachmann, M., Arkın, H., Janke, W.: Multicanonical study of coarse-grained off-lattice models for folding heteropolymers. Physical review E **71**(3) (2005) 031906
20. Parker, J., Guo, D., Hodges, R.: New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: correlation of predicted surface residues with antigenicity and x-ray-derived accessible sites. Biochemistry **25**(19) (1986) 5425–5432
21. Wolfenden, R., Lewis, C.A., Yuan, Y., Carter, C.W.: Temperature dependence of amino acid hydrophobicities. Proceedings of the National Academy of Sciences **112**(24) (2015) 7484–7488
22. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
23. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. (2010) 249–256

A Bio-inspired Approach for Collaborative Exploration with Mobile Battery Recharging in Swarm Robotics

Maria Carrillo[1], Ian Gallardo[1], Javier Del Ser[1,2,3], Eneko Osaba[2], Javier Sanchez-Cubillo[2], Miren Nekane Bilbao[1], Akemi Gálvez[4,5], and Andrés Iglesias[4,5]

[1] University of the Basque Country (UPV/EHU), Bilbao, Spain
[2] TECNALIA, Derio, Spain
[3] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain
[4] Universidad de Cantabria, Santander, Spain
[5] Toho University, Funabashi, Japan
javier.delser@tecnalia.com

Abstract. Swarm Robotics are widely conceived as the development of new computationally efficient tools and techniques aimed at easing and enhancing the coordination of multiple robots towards collaboratively accomplishing a certain mission or task. Among the different criteria under which the performance of Swarm Robotics can be gauged, energy efficiency and battery lifetime have played a major role in the literature. However, technological advances favoring power transfer among robots have unleashed new paradigms related to the optimization of the battery consumption considering it as a resource shared by the entire swarm. This work focuses on this context by elaborating on a routing problem for collaborative exploration in Swarm Robotics, where a subset of robots is equipped with battery recharging functionalities. Formulated as a bi-objective optimization problem, the quality of routes is measured in terms of the Pareto trade-off between the predicted area explored by robots and the risk of battery outage in the swarm. To efficiently balance these conflicting two objectives, a bio-inspired evolutionary solver is adopted and put to practice over a realistic experimental setup implemented in the VREP simulation framework. Obtained results elucidate the practicability of the proposed scheme, and suggest future research leveraging power transfer capabilities over the swarm.

Keywords: Swarm Robotics, battery recharging, routing, NSGAII.

1 Introduction

Robotics have evolved dramatically over the years to feature unprecedented levels of intelligence, resulting in an ever-growing number of scenarios benefiting from their widespread application to accomplish complex missions, e.g. structural health monitoring, oil and gas industry, manufacturing, disaster management, precision agriculture and logistics, among many others. Providing robots with smart sensing, communication and organization functionalities allows them to capture information, operate, reason and infer knowledge from the environment in a collaborative manner. Research aimed at enhancing such functionalities by embracing elements from Artificial Intelligence and

Distributed Computing has coined the so-called Swarm Robotics concept, which refers to the deployment of a set of robots that collaborate with each other so as to collectively perform a mission in a computationally efficient fashion [1, 2].

In general, Swarm Robotics may rely on several key technologies to attain higher levels of autonomy, optimized operation and self-organization. Unfortunately, it is often the limited battery lifetime of robots not only what restricts most the autonomy of the swarm, but also what puts at risk the feasibility of complex missions where robots operate without any human intervention, as in e.g. the exploration of collapsed infrastructures after a massive disaster [3] or the structural assessment of undersea drilling equipment [4]. Despite notable advances in energy efficient robot mechanics, the battery capacity poses severe operational constraints to Swarm Robotics, to the point of jeopardizing their potential use in complex endeavors.

To overcome this issue, many research efforts have been devoted towards augmenting the power capacity of robot batteries, either by proposing new materials and chemical components or by deriving new mechanical improvements that extend further their lifetime by virtue of a lower power consumption [5]. For this same purpose, the community has also focused its attention towards the consideration of the aggregate battery power of the entire robotic swam as a whole, an unique resource whose management is to be optimized globally over all robots rather than locally. This approach grounds on advances in wireless/mobile robotic charging [6] and the deployment of mobile charging stations in the swarm [7], which can be exploited as a resource to actively locate and replenish the battery of other robots. This research topic has been very active in this regard, as evinced by the short literature review provided in what follows.

1.1 Related Work

A remarkable amount of interesting studies has been published in the last decade focused in power charging and battery consumption of swarm robots. Haek et al. discussed in [8] the importance of swarm robustness, defining this concept as the ability of the robotic swarm to perform a complex task avoiding the total drainage of their batteries. In this work authors present a solution to allow robots to robustly achieve their assigned tasks, which mainly consists of the use of power stations or power banks. In [9], a collective energy distribution system is proposed for a dust cleaning swarm of robots. Authors of this study explore the concept of *trophallaxis*, previously introduced in [10], which refers to individual robots donating an amount of their own energy reserve to other robots of the swarm. This same concept of altruistic behavior is explored in [11], materializing the idea in a specific robot architecture called CISSBot. Apart from battery charging, sharing and consumption, several additional features are also considered and studied in this contribution, such as a collision-free proximity motion control. Additional research on this topic can be found in [12].

Another interesting approach to energy consumption is the one recently proposed in [13], where an Energy-Aware Particle Swarm Optimization (EAPSO) bioinspired solver is designed to optimize the movement strategy of aerial micro-robots. Interestingly, the optimization process considers the energy levels of the robots for their efficient movement. Although authors do not propose any charging mechanism, the designed method renders a considerable reduction of the total energy consumption, making the robotic swarm more reliable and robust. Another bioinspired scheme sensitive to the

consumed energy is the Honey Bee inspired swarm in [14], which improves the energy efficiency and is proven to be effective for foraging environments, such as the collection of crops of materials.

Also interesting to mention is the preliminary research presented by [15], in which an immune system response is studied for the development of energy sharing strategies. In that case, the granuloma formation is explored, which is a process in which undesired components are removed by immune systems. This behavioral concept is mapped to the components of a Swarm Robotics system, enhancing the fault tolerance of the deployed robots. A further step was taken in [16], where another immune system mechanism is proposed based on the use of contact-less energy charging areas and their simulation-based comparison to other energy charging mechanisms. A similar technique was proposed in [17] to add self-healing capabilities to robotic swarms.

As stated in [18], an usual trend in the literature for dynamic energy charging of robots is based on the deployment of power banks or removable chargers. Despite being quite effective, this approach has its own disadvantages, such as the resulting weight increase of the robot equipment, often crucial in critical missions. With the intention of overcoming these issues, [18] describes initial research on the implementation of an energy-sharing scheme using a two-way communication mechanism. Finally, in [19] an energy-encrypted contact-less system is described for improving the charging performance and the energy transmission mechanism of swarm robots. To this end wireless power transfer is used, enabling robots to charge their batteries even in moving situations. Other contributions related to dynamic energy charging include [20], which elaborates on a novel tree-based schedule for mobile charger robots, which minimizes the travel distance without causing energy depletion; and [21], which presents a versatile mobile charging station capable of actively locating and replenishing the battery of inactive robots.

1.2 Contribution

Even though the literature has been profitable in what regards to Swarm Robotics with mobile battery recharging nodes, to the best of the authors' knowledge routing for exploration missions in Swarm Robotics has so far been addressed without considering such nodes as assets whose routes over the scenario at hand can be jointly optimized with those of exploring robots. Furthermore, when dealing with overly complex scenarios to be explored, the total area sensed by exploring robots can be intuitively thought of as a conflicting objective with the remaining battery margin; in this sense, enforcing the swarm to explore the entire area spanned by the scenario could create a risk of any robot to run out of battery on site, and be left dead and unrecoverable. This work aims at addressing this research niche by modeling and solving a bi-objective routing problem for mobile swarm robotics considering the minimization of this risk as a second fitness metric that quantifies the quality of a generated route plan. The problem formulation also includes the search for optimal routing plans for mobile battery recharging nodes along with the routes of exploring robots. Both are solved efficiently by means of a bi-objective bio-inspired solver driven by the aforementioned objectives. Results obtained from a realistic simulation framework implemented in VREP [22] are shown to be promising, with several future research lines stemming therefrom.

The rest of this paper is structured as follows: first, Section 2 formulates mathematically the optimization problem under study, including the conflicting objectives to be maximized. Next, Section 3 delves into the utilized bi-objective solver, followed by Sections 4 and 5 detailing the simulation setup and discussing the obtained results, respectively. Section 6 concludes the paper.

2 Problem Statement

Following the schematic diagram depicted in Fig. 1, we assume a swarm $\mathcal{N}$ of $|\mathcal{N}| = N$ robots, with time-dependent positions $\{\mathbf{p}_n^{\triangle,t}\}_{n=1}^N \doteq \{(x_n^{\triangle,t}, y_n^{\triangle,t})\}_{n=1}^N$ (with t denoting time) over a square area $S^{\square} \doteq [X_{min}, X_{max}] \times [Y_{min}, Y_{max}]$. Each of such robots is equipped with sensors that allow them to explore an area $\{S_n^{\triangle,t}\}_{n=1}^N$ around its location at time t, e.g. if the area is circular with radius $R_n^{\triangle}$, then $S_n = \{(x,y) \in S^{\square} : (x - x_n^{\triangle,t})^2 + (x - x_n^{\triangle,t})^2 \leq R_n^2\}$ (areas shaded in ■, ■ and ■ in Fig. 1). The total area $S^T(t)$ explored by the robotic swarm at time t' will be then given by

$$S^T(t') = \bigcup_{t=1}^{t'} \bigcup_{n=1}^{N} S_n^{\triangle,t}. \tag{1}$$

Another set of $M \leq N$ robots $\mathcal{M}$ with battery recharging capabilities is deployed in the same location jointly with $\mathcal{N}$, with coordinates $\{\mathbf{p}_m^{\odot,t}\}_{m=1}^M \doteq \{(x_m^{\odot,t}, y_m^{\odot,t})\}_{m=1}^M$. A robot $m \in \mathcal{M}$ will recharge the battery of a robot $n \in \mathcal{N}$ whenever 1) their distance $d_{m,n}^t$ falls below a certain threshold D_{max} (area in ■ in Fig. 1), i.e.

$$d_{m,n} = \sqrt{\left(x_m^{\odot,t} - x_n^{\triangle,t}\right)^2 + \left(y_m^{\odot,t} - y_n^{\triangle,t}\right)^2} \leq D_{max}, \tag{2}$$

and 2) the above condition holds for a minimum of T_{min} seconds, comprising the power plug coupling/uncoupling along with physical maneuvers to align connectors. If both conditions hold, energy is transferred from robot $m \in \mathcal{M}$ to $n \in \mathcal{N}$ at a rate of β units of energy per second (measured in e.g. Watts). Furthermore, the movement of the robot itself involves a battery consumption of γ units of power per unit of distance, so that in a certain time gap ΔT measured from time t the remaining amount of battery $B_n^{\triangle,t+\Delta T}$ in robot n can be mathematically expressed as

$$B_n^{\triangle,t+\Delta T} = \min\left\{[1 + I_D \cdot I_T \cdot \beta] \cdot B_n^{\triangle,t} - \gamma V_n^{\triangle} \Delta T, B_{max}\right\}, \tag{3}$$

where $V_n^{\triangle}$ denotes the cruise speed of the robot (in units of distance per unit of time), and I_D and I_T are binary indicator functions such that $I_D = 1$ if $d_{m,n}^{t'} \leq D_{max}\ \forall t' \in [t, t + \Delta T]$, and $I_T = 1$ if $\Delta T \geq T_{min}$ (0 otherwise in both cases). In the above expression B_{max} stands for the nominal maximum battery load (in units of power) of the robot model, which without loss of generality is assumed to be equal throughout the entire robotic swarm.

With this definition in mind, the goal of the routing optimization problem is essentially the determination of an optimal set of routes composed by $N + M$ waypoints $\mathbf{W}^{\triangle,t,\curvearrowright} \doteq \{\mathbf{w}_n^{\triangle,t,\curvearrowright}\}_{n=1}^N = \{(x_n^{\triangle,t,\curvearrowright}, y_n^{\triangle,t,\curvearrowright})\}_{n=1}^N$ and $\mathbf{W}^{\odot,t,\curvearrowright} \doteq \{\mathbf{w}_m^{\odot,t,\curvearrowright}\}_{m=1}^M = \{(x_m^{\odot,t,\curvearrowright}, y_m^{\odot,t,\curvearrowright})\}_{m=1}^M$ for all robots in the swarm (both explorers and battery chargers). Here optimality of the set of discovered routes refers to the Pareto relationship

between the explored area and a quantitative measure of the *risk of no return* taken when the entire swarm is commanded to follow a certain route. Intuitively, the more area the swarm explores, the more likely is the chance that any of the robots in the swarm lacks enough battery to return to the point $\{(x^{\Delta,0}, y^{\Delta,0})\}$ where robots had been initially located. This risk is crucial in many practical situation, e.g. disaster events where the topological characteristics of the facility to be explored remain unknown to the command center before and while the mission is performed by the robotic swarm.

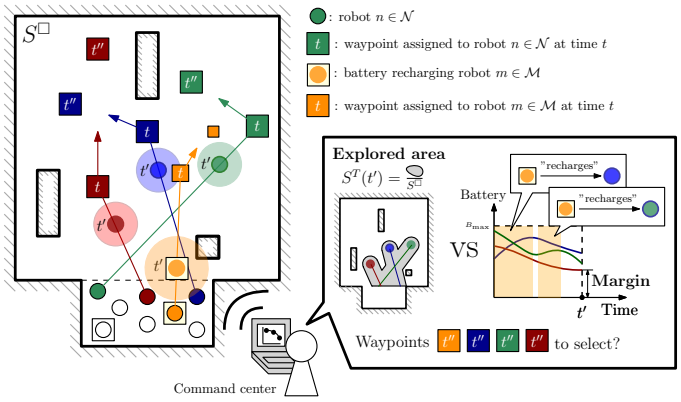


Fig. 1. Schematic diagram of the scenario tackled in this paper.

Mathematically this risk can be modeled by accounting, over the whole robotic swarm, for battery margin $B_n^{\blacktriangle,t}$ expected to be left for every robot should it proceed and move to the assigned waypoint and return safely to $\{(x^{\Delta,0}, y^{\Delta,0})\}$. Assuming that the route optimization is performed at time t, the value of the battery margin $B_n^{\blacktriangle,t}$ for robot $n \in \mathcal{N}$ when commanded to go to waypoint $\mathbf{w}_n^{\Delta,t,\looparrowright} = (x_n^{\Delta,t,\looparrowright}, y_n^{\Delta,t,\looparrowright})$ can be estimated as

$$B_n^{\blacktriangle,t}(\mathbf{p}_n^{\Delta,t}, \mathbf{w}_n^{\Delta,t,\looparrowright}, \{\mathbf{p}_m^{\Delta,t}\}_{m=1}^{M}, \{\mathbf{w}_m^{\Delta,t}\}_{m=1}^{M}) = B_n^{\Delta,t} - B_n^{\Delta,t+\Delta T_{\mathbf{p},\mathbf{w}}+\Delta T_{\mathbf{w},\mathbf{p}_0}}, \quad (4)$$

where $\Delta T_{\mathbf{p},\mathbf{w}}$ and $\Delta T_{\mathbf{w},\mathbf{p}_0}$ are the times taken for robot $n \in \mathcal{N}$ to travel from its current point $\mathbf{p}_n^{\Delta,t}$ to the assigned waypoint $\mathbf{w}_n^{\Delta,t,\looparrowright}$ and therefrom to its initial position $\{(x^{\Delta,0}, y^{\Delta,0})\}$. This estimation is made by assuming that the robot goes straight without colliding with any object nor any other robot along its path. It should be remarked that as per (3), the battery expenditure reflected in $B_n^{\Delta,t+\Delta T_{\mathbf{p},\mathbf{w}}+\Delta T_{\mathbf{w},\mathbf{p}_0}}$ takes into account not only the power consumed by the robot dynamics (which depends on its speed V_n and the traversed distances), but also time periods along the path during which the relative position between battery recharging robots and robot $n \in \mathcal{N}$ fulfill conditions I_D and I_T required to recharge the battery of robot n on the move. The total duration of such recharging periods can be computed as $\sum_{(t_s,t_e)\in\mathcal{T}_n^{\Delta,t}}(t_e - t_s)$ over the set of periods $\mathcal{T}_n^{\Delta,t}$, defined as

$$\mathcal{T}_n^{\Delta,t} \doteq \{(t_s, t_e) \in [t, t + \Delta T_{\mathbf{p},\mathbf{w}} + \Delta T_{\mathbf{w},\mathbf{p}_0}] \text{ such that :}$$
$$1)\ t_e > t_s;\ 2)\ \exists m \in \mathcal{M} : d_{mn}^{t'} \leq D_{max} \forall t' \in [t_s, t_e];\ \text{and } 3)\ t_e - t_s \geq T_{min}\}, \quad (5)$$

with $[t'_s, t'_e] \cap [t''_s, t''_e] = \emptyset \; \forall (t'_s, t'_e), (t''_s, t''_e) \in \mathcal{T}_n^{\triangle,t}$. Therefore, the swarm-wide battery margin $B^T(t)$ to be maximized at time t so as to keep the aforementioned risk to its minimum is given by

$$B^T(t) = \min_{n \in \mathcal{N}} \left\{ \max \left\{ 0, B_n^{\blacktriangle,t}(\mathbf{p}_n^{\triangle,t}, \mathbf{w}_n^{\triangle,t,\looparrowright}, \{\mathbf{p}_m^{\triangle,t}\}_{m=1}^M, \{\mathbf{w}_m^{\triangle,t}\}_{m=1}^M) \right\} \right\}, \tag{6}$$

from where the formal statement of the problem tackled in this work follows:

$$\underset{\mathbf{W}^{\odot,t,\looparrowright}, \mathbf{W}^{\triangle,t,\looparrowright}}{\text{maximize}} \quad \{S^T(t), B^T(t)\}, \tag{7a}$$

namely, as the simultaneous maximization of two conflicting objectives: the surface explored by the robotic swarm and the minimum expected battery margin over the robots should it be commanded to return to the initial deployment point after reaching the enforced waypoint. $\mathbf{W}^{\triangle,t,\looparrowright} \in S^{\square}$ and $\mathbf{W}^{\odot,t,\looparrowright} \in S^{\square}$.

3 Proposed Solver

In order to efficiently tackle the above problem, we propose to apply a centralized metaheuristic solver capable of optimally balancing the two objective functions considered in its formulation. The optimizer relies on the renowned Non-dominated Sorting Genetic Algorithm (NSGA-II, [23]), a bio-inspired approach that hinges on the concepts of non-dominance ranking and crowding distance to guide a multi-objective search over a set of potential candidate solutions (in this case, waypoints defining routes). In essence NSGA-II sorts a population of candidates according to 1) whether each solution within the population *dominates*, in terms of Pareto optimality, other solutions in the pool (yielding the so-called dominance rank of the Pareto front to which the solution at hand belongs); and 2) the closest distance from every individual to the rest of solutions (corr. crowding distance). By applying this dual selection procedure along with genetically inspired crossover and mutation operators (with probabilities P_c and P_m, respectively), the Pareto optimality of solutions contained in the population becomes improved iteration after iteration, to eventually yield a Pareto front estimation after a number of iterations of this search procedure.

An algorithmic description of the NSGA-II approach designed in this work is provided in Algorithm 1. Individuals are encoded directly as $N + M$ vectors $\mathbf{w}_i^p$ denoting the waypoints of all robots in the scenario, where $i \in \{1, \ldots, N, N+1, \ldots, N+M\}$, $p \in \{1, \ldots, P\}$, P denoting the population size and $\mathbf{w}_i^p \in \mathcal{S}^{\square} \; \forall i, \; p$. A uniform crossover operator and a Gaussian mutation with standard deviation σ have been selected as heuristic operators. The iterative application of these operators and the NSGA-II selection scheme outlined above is stopped after $\mathbb{I}$ iterations. It is important to remark at this point that the solver must be run incrementally at certain time instants, e.g. the solver is not run constantly along time but rather triggered at time ticks embedded in the set $\mathcal{T} \in \mathbb{R}[t_{ini}, t_{end}]$, where t_{ini} is the time at which the robotic swarm is first deployed and t_{end} is the time at which the battery margin $B^T(t_{end})$ in the estimated Pareto front falls below a fraction λ of the maximum battery capacity B_{max}. For the sake of simplicity, the NSGA-II solver will be executed once all robots have reached their commanded waypoints $\mathbf{W}^{\triangle,t,\looparrowright}$ and $\mathbf{W}^{\odot,t,\looparrowright}$ optimized previously, which yields the time instants contained in $\mathcal{T}$. To match this incremental nature of the proposed optimization schedule, the population of individuals is accordingly initialized by including

the best front found in the previous NSGA-II execution, randomly setting the remaining individuals until filling the population.

Algorithm 1: NSGA-II solver applied to the problem under study.

Data: Number of exploration robots N; number of battery recharging robots M; dimensions of the scenario $X_{min}, X_{max}, Y_{min}, Y_{max}$; sensing radii $\{R_n\}_{n=1}^{N}$; maximum distance D_{max} and minimum time T_{min} for battery recharge; nominal robot speeds $\{V_n^{\triangle}\}_{n=1}^{N}$ and $\{V_m^{\odot}\}_{m=1}^{M}$; maximum battery capacity B_{max}; battery charging rate β; battery consumption rate γ; crossover and mutation probabilities P_c and P_m; population size P; maximum number of iterations $\mathbb{I}$; proportion of the minimum battery margin to the maximum battery capacity λ.

Deploy all robots on the initial location $(x^{\triangle,0}, y^{\triangle,0})$, and set waypoints $\mathbf{w}_n^{\triangle,t_{ini},\looparrowright}$ and $\mathbf{w}_m^{\odot,t_{ini},\looparrowright}$ equal to $(x^{\triangle,0}, y^{\triangle,0})$ $\forall n \in \mathcal{N}$ and $\forall m \in \mathcal{M}$

Set $t' = t_{ini}$ and $\mathcal{T} = \{t_{ini}\}$

while $B^T(t) \geq \lambda B_{max}$ **do**

  while $\mathbf{p}_n^{\triangle,t} \neq \mathbf{w}_n^{\triangle,t',\looparrowright}$ and $\mathbf{p}_m^{\odot,t} \neq \mathbf{w}_m^{\odot,t',\looparrowright}$ $\forall n, m$ **do**

    Let robots move to their assigned waypoints $\mathbf{w}_n^{\triangle,t_{ini},\looparrowright}$ and $\mathbf{w}_m^{\odot,t_{ini},\looparrowright}$

    Update remaining battery $\{B_n^{\blacktriangle,t}\}_{n=1}^{N}$ as per (4) and (5)

  if $t' = t_{ini}$ **then**

    Initialize P individuals in the population uniformly at random from $\mathcal{S}^{\square}$

  else

    Retrieve the estimated Pareto from the previous run, introduce it in the population. and fill the remaining individuals randomly over $\mathcal{S}^{\square}$

  for $it \leftarrow 1$ **to** $\mathbb{I}$ **do**

    Select parents, recombine them (w.p. P_c) and mutate (w.p. P_m) the produced new offspring that represent a new set of P waypoints

    Evaluate explored area and battery margin of offspring as per (1), (6)

    Sort previous and new waypoints by rank and crowding distance

    Discard the worst P individuals in the sorted, concatenated population

  The estimated Pareto is given by the P individuals remaining in population

  Select the set of waypoints in the estimated front that best suits the commanding policy (e.g. maintain a battery margin above 10%), and assign them to robots

  Set $t' = t$, and $\mathcal{T} = \mathcal{T} + \{t\}$

All robots to initial position by $\mathbf{w}_n^{\triangle,t_{ini},\looparrowright} = \mathbf{w}_m^{\odot,t_{ini},\looparrowright} = (x^{\triangle,0}, y^{\triangle,0})$ $\forall n, m$

4 Simulation Setup

In order to assess the performance of the proposed bi-objective routing approach, a simulation setup has been constructed by resorting to VREP, a renowned software platform that permits to realistically model and perform experimental studies with swarms of robots. In order to extract valuable insights, we have kept the dimensions of the experimental scenario reduced to $N = 5$ exploring robots and a single battery recharging node ($M = 1$) deployed on a 10×10 m^2 square area. The maximum distance and minimum time to recharge batteries are set to $D_{max} = 1$ meters and $T_{min} = 3$ seconds, respectively. Robots with six mechanical legs (also referred to as *hexapods*) and

diameter size equal to 0.5 m are utilized, with speeds equal to $V_n^{\triangle} = 3.5$ cm/s $\forall n \in \mathcal{N}$ and $V_m^{\odot} = 2.6$ cm/s. Battery recharging is done at a rate of 1 % per second with respect to the nominal maximum capacity B_{max} of exploring robots, whereas the recharging node is equipped with a total battery capacity equal to $10 \cdot B_{max}$. The battery depletion rate is fixed to $\gamma = 1.5$ % of B_{max} per linear meter. As for the parameters of the NSGA-II solver, crossover and mutation rates are set to $P_c = 1$ and $P_m = 0.1$, with a population size of $P = 20$ individuals and $\mathbb{I} = 100$ iterations per run. The decision making criterion adopted to select a route among the estimated Pareto fronts was based on selecting the route whose associated battery margin is closest to 20% of B_{max}. If no route with margin greater than this threshold, the robot swarm is enforced to return to the origin position. Fig. 2 illustrates, from two different perspectives, the scenario generated in VREP and simulated to yield the results discussed in the next section[1].

5 Results and Discussion

The discussion on the results obtained by the proposed scheme starts with Fig. 3, which illustrates the set of estimated Pareto fronts along time under different assumptions. Specifically, every plot in this figure contains a three-dimensional cloud of points – each representing a given route plan (set of waypoints) – which results from the aggregation of all fronts estimated in simulation time for a single experiment. A total of 10 executions of the NSGA-II solver have sufficed for illustrating the main benefit of our proposed routing scheme: by incorporating battery recharging functionalities, the autonomy of the entire robotic swarm is enhanced, so that a larger area can be explored for a given decision making criterion imposed on the minimum admissible battery margin for the robots to return back and safe to the base.

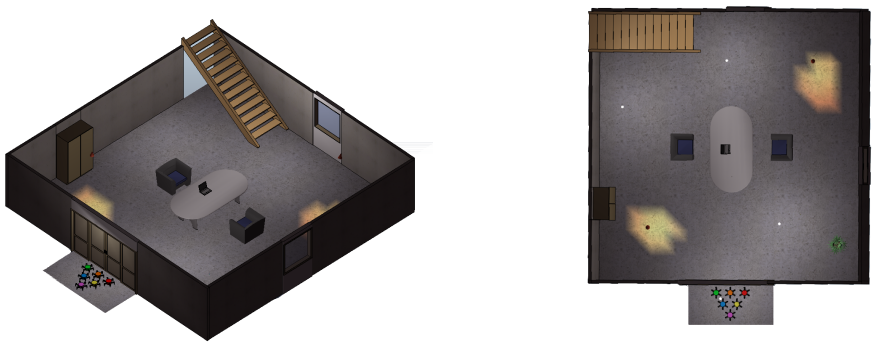


Fig. 2. Visual representation of the simulated setup yielding the results later discussed in the manuscript; (left) isometric view; (right) top-down view. The robot dynamics are provided by the VREP framework, whereas the NSGA-II routing approach has been implemented in Python and communicates with VREP via remote API functions.

To this end two different cases are assessed, depending on the exploration radii assumed for the sensing robots: 1) $R_n = 0.9$ meters, which should a priori render

[1] Videos showing how robots move over this scenario can be found at: `https://youtu.be/r31teMtWRF0` and `https://youtu.be/zewRVZQpvP8`.

minimum gains due to a more efficient area exploration; and 2) $R_n = 0.5$ meters, smaller sensing radii for which the incorporation of battery recharging functionalities in the swarm should provide higher gains. Indeed, this intuition is confirmed by the results in the plots: as evinced by the plot on the left (higher exploration radii), almost no exploration gain is obtained by including battery recharging functionalities (■) when compared to a unassisted robot swarm (■). However, when reducing the sensing radius, robots must traverse longer distances in order to explore the entire scenario, which leads to higher battery consumption levels that could be compensated efficiently by including a battery recharging node. This is precisely what the plot on the right in Fig. 3 reveals: when inspecting the evolution of the maximum battery margin in the fronts computed along time, it is straightforward to note that the margin of the unassisted swarm (■) decreases much faster than that of its assisted counterpart (■), falling below the minimum admissible threshold (20%) imposed by the mission commander. As a result, the entire swarm is commanded to return to the base once 61% of the scenario has been explored. By including the mobile recharging node, the battery margin degrades smoothly along time, and is maintained above the threshold to explore a higher area percentage (ca. 80%) even for more conservative policies. For instance, should it have been set to 60% the unassisted swarm would have explored less than 50% of the area; in the assisted case robots would have been operative for a longer time, attaining explored area ratios close to 80%.

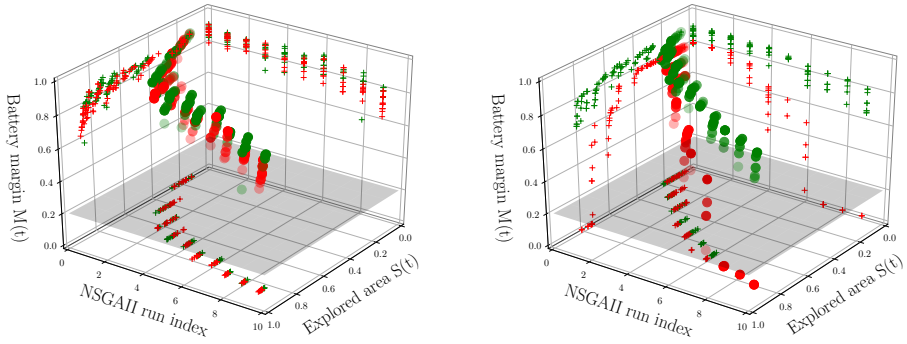


Fig. 3. Three-dimensional plot showing the Pareto trade-off between battery margin and explored area estimated by the NSGA-II solver as the simulation evolves (represented by the NSGA-II run index). The left plot corresponds to the case when $R_n = 0.9$ meters, whereas the right plot depicts the case when $R_n = 0.5$ meters, in both cases $\forall n \in \{1, \ldots, 5\}$. Also are included in the plots the two-dimensional projections of the point cloud along every axis, so that the progression of the maximum achievable value of each metric. The plane shaded in gray indicates the minimum admissible battery margin imposed by the mission commander (20%).

Besides the evidence provided by the above plots, further insights can be extracted by taking a closer look at the trajectories traced by the robots in the swarm for both cases. One should expect that for high values of the sensing radii R_n, nodes should feature relatively less dynamic mobility patterns over the scenario than those corresponding to lower values of this parameter. The plots in Fig. 4 go in line with this expected

behavior. In particular mobility traces of the robotic swarm are shown for the assisted robotic swarm with $R_n = 0.5$ meters (left) and $R_n = 0.9$ meters (right). It can be noted that the former case features rectilinear trajectories composed by long segments, whereas in the latter all robots in the swarm describe topologically tangled traces, and few cases reach the boundaries of the scenario. In summary, the sensing radii plays a crucial role in the behavior of the swarm and ultimately, in the attainable performance gain from the introduction of mobile recharging nodes in the swarm.

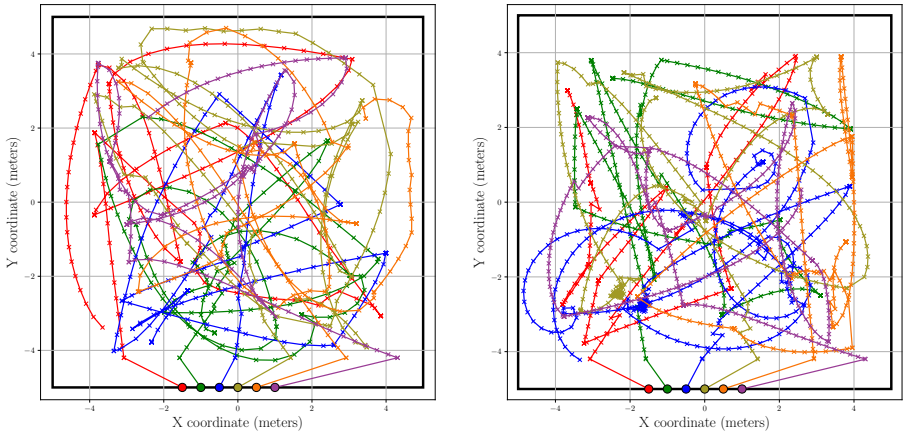


Fig. 4. Trajectories followed by the robots in the swarm for $R_n = 0.5$ meters (left) and $R_n = 0.9$ meters (right). A visual inspection permits to infer that lower values of the sensing radius make all trajectories be shorter and more complex as a result of a lower overlapping between the sensing areas of robots in the swarm. On the contrary, when the sensing radius increases robots describe cleaner, rectilinear trajectories.

6 Concluding Remarks

In this paper, a routing problem for collaborative exploration in Swarm Robotics has been presented. An analysis of the recent literature supports that one of the main issues in these systems is the energy consumption and reliability of the swarm, which jeopardizes the performance of complex missions and tasks. This identified issue is what lies behind the rationale for this research work: to include a subset of robots in the swarm endowed with battery charging capabilities. The challenge resides in how to properly route the robots in the scenario considering the existence of such nomadic battery recharging nodes, which has been formulated as a bi-objective optimization problem where a Pareto equilibrium must be met between the explored area and the risk of battery outage. In order to solve efficiently this problem, a bio-inspired approach has been designed based on the well-known NSGA-II solver. A realistic experimental setup comprising the VREP robotic simulation framework has been constructed so as to shed light on how the proposed solver performs in practice. The obtained results have proven empirically the practicality and inherent utility of the proposed routing scheme, which provides the commander of the mission with more valuable information for decision making than traditional schemes based on a single fitness function.

Several lines of research related to this work have been planned for the near future, e.g. the inclusion of other bioinspired multi-objective heuristic engines (e.g. SMPSO, MOEA/D) and their comparison to each other in terms of multi-objective indicators. Another research path that will be prospected will gravitate on relaxing and extending the assumptions and constraints defining the considered scenario towards, for instance, co-located exploration tasks (demanding different sensing equipment). Among them, the most challenging research direction to be followed focuses on distributing the intelligence among the robots in order to realize a *true* robotic swarm, namely, a swarm of robots that communicate to each other and exchange information, deciding on an optimal set of waypoints without requiring a centralized command center as the one assumed in this work.

Acknowledgments

E. Osaba and J. Del Ser would like to thank the Basque Government for its funding support through the EMAITEK program. Likewise, the involvement of A. Galvez and A. Iglesias in this work has been funded by the *Agencia Estatal de Investigación* (grant no. TIN2017-89275-R), the European Union through FEDER Funds (AEI/FEDER), and the project #JU12, jointly supported by the public body SODERCAN and European Funds FEDER (SODERCAN/FEDER).

References

1. Beni, G.: From swarm intelligence to swarm robotics. In: International Workshop on Swarm Robotics. (2004) 1–9
2. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence **7**(1) (2013) 1–41
3. Miranda, K., Molinaro, A., Razafindralambo, T.: A survey on rapidly deployable solutions for post-disaster networks. IEEE Comm. Mag. **54**(4) (2016) 117–123
4. Bogue, R., Bogue, R.: Underwater robots: a review of technologies and applications. Industrial Robot: An International Journal **42**(3) (2015) 186–191
5. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.G.: Energy-efficient motion planning for mobile robots. In: IEEE International Conference on Robotics and Automation (ICRA'04). Volume 5. (2004) 4344–4349
6. Johnson, J., Stoops, M., Schwartz, B., Masters, N., Hasan, S.: Techniques for mobile device charging using robotic devices (November 15 2016) US Patent 9,492,922.
7. Couture-Beil, A., Vaughan, R.T.: Adaptive mobile charging stations for multi-robot systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. (2009) 1363–1368
8. Haek, M., Ismail, A.R., Basalib, A., Makarim, N.: Exploring energy charging problem in swarm robotic systems using foraging simulation. Jurnal Teknologi **76**(1) (2015) 239–244
9. Melhuish, C., Kubo, M.: Collective energy distribution: Maintaining the energy balance in distributed autonomous robots using trophallaxis. Distributed Autonomous Robotic Systems 6 (2007) 275–284
10. Schmickl, T., Crailsheim, K.: Trophallaxis among swarm-robots: A biologically inspired strategy for swarm robotics. In: IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics. (2006) 377–382
11. Schioler, H., Ngo, T.D.: Trophallaxis in robotic swarms-beyond energy autonomy. In: IEEE International Conference on Control, Automation, Robotics and Vision. (2008) 1526–1533

12. Schmickl, T., Crailsheim, K.: Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm. Autonomous Robots **25**(1) (2008) 171–188
13. Mostaghim, S., Steup, C., Witt, F.: Energy aware particle swarm optimization as search mechanism for aerial micro-robots. In: IEEE Symposium Series on Computational Intelligence. (2016) 1–7
14. Lee, J.H., Ahn, C.W., An, J.: A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics. (2013) 489–493
15. Al Haek, M., Ismail, A.R., Nordin, A., Sulaiman, S., Lau, H.: Modelling immune systems responses for the development of energy sharing strategies for swarm robotic systems. In: International Conference on Computational Science and Technology. (2014) 1–6
16. Al Haek, M., Ismail, A.R.: Simulating the immune inspired energy charging mechanism for swarm robotic systems. Journal of Theoretical & Applied Information Technology **95**(20) (2017)
17. Timmis, J., Ismail, A.R., Bjerknes, J.D., Winfield, A.F.: An immune-inspired swarm aggregation algorithm for self-healing swarm robotic systems. Biosystems **146** (2016) 60–76
18. Ismail, A.R., Desia, R., Zuhri, M.F.R.: The initial investigation of the design and energy sharing algorithm using two-ways communication mechanism for swarm robotic systems. In: Computational Intelligence in Information Systems. (2015) 61–71
19. Wang, J., Liang, Z., Zhang, Z.: Energy-encrypted contactless charging for swarm robots. In: International Magnetics Conference. (2017) 1–1
20. He, L., Cheng, P., Gu, Y., Pan, J., Zhu, T., Liu, C.: Mobile-to-mobile energy replenishment in mission-critical robotic sensor networks. In: IEEE INFOCOM. (2014) 1195–1203
21. Arvin, F., Samsudin, K., Ramli, A.R.: Swarm robots long term autonomy using moveable charger. In: International Conference on Future Computer and Communication. (2009) 127–130
22. Rohmer, E., Singh, S.P., Freese, M.: V-rep: A versatile and scalable robot simulation framework. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2013) 1321–1326
23. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197

Cooperative Model for Nature-Inspired Algorithms in Solving Real-World Optimization Problems

Petr Bujok[1]

Department of Informatics and Computers
University of Ostrava
30. dubna 22, 70200 Ostrava
Czech Republic
petr.bujok@osu.cz

Abstract. A cooperative model of eight popular nature-inspired algorithms (CoNI) is proposed and compared with the original algorithms on benchmark set CEC 2011 collection of 22 real-world optimization problems. The results of experiments demonstrate the superiority of CoNI variant in the most of the real-world problems although some of original nature-inspired algorithms perform rather poorly. Proposed CoNI shares the best position in 20 out of 22 problems and achieves the best results in 8 out 22 test problems. Further fundamental points for improvement of CoNI are in selection of topology, migration policy, and migration frequency.

Keywords: global optimization, nature-inspired algorithms, real-world problems, cooperative model

1 Introduction

Many researchers have developed tens of optimization algorithms based on systems from nature [1]. Beside these methods, another scientists propose existing good-performing methods enhanced by new features [2, 3]. The goal of this paper is to reveal possibility of cooperation of nature-inspired algorithms in order to obtain more efficient optimization algorithm.

In our recent works [4, 5], we have experimentally compared the performance of nature-inspired algorithms on the collection of real-world optimization problems. It was found that none of eight nature-inspired algorithms selected to the comparison is able to provide a similar results as three recently proposed adaptive DE variants. Furthermore, some of the nature-inspired methods even perform worse than the blind random search.

In this paper, we apply and study the results of cooperative model of well-known nature-inspired optimization algorithms. A practicality of the proposed model will be achieved on the real-world problems [6] with various dimensionality. The purpose of use of these problems is simple, we want to show the

performance of compared algorithms on selected real problems to help scientists in choice of the proper optimization method. A blind random search method was not selected for this experiment.

When a problem is solved in global optimization, there is an objective function $f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$ defined on the search domain Ω limited by lower and upper boundaries, i.e. $\Omega = \prod_{j=1}^{D}[a_j,\ b_j],\ a_j < b_j,\ j = 1, 2, \ldots, D$. The global minimum point $\boldsymbol{x}^*$, which satisfies condition $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega$ is the solution of the problem.

The rest of the paper is organized as follows. Section 2 shows brief description of the nature-inspired algorithms selected for experimental comparison. A cooperation model of nature-inspired algorithms is described in Section 3. Experimental setting and methods applied to statistical assessment are described in Section 4. Experimental results on real-world optimization problems are presented in Section 5. Section 6 describes conclusion of the paper with some final remarks.

2 Selected Nature-Inspired Algorithms

The survey of bio-inspired algorithms has been presented recently in [1]. The book [7] along with mentioned survey were the main sources for the selection of nature-inspired algorithms for this experimental comparison. Based on these sources and previous studies [4, 5], the list of alphabetically sorted eight nature-inspired methods with their descriptions follows.

The artificial bee colony algorithm (ABC) was proposed by Karaboga in 2005 [8]. This algorithm models the behavior of the bees consist of three groups - employed bees, onlookers bees, and scouts. The only input parameter *limit*, usually equal to the population size, controls a number of unsuccessful new 'food positions' (position in Ω) necessary to find a new random food position. An employed ith bee jth position is updated by $y(i,j) = P(i,j) + (P(i,j) - P(r,j))\ U(-1,1)$, where j is randomly selected index from $(1, D)$ of the position to be updated (D is the dimension of the problem), r is randomly selected bee different from current ith bee and $U(-1,1)$ is a random number from the uniform distribution with parameters given in parentheses.

The bat algorithm (abbreviated Bat) simulates an echolocation behavior of real bats controlled by emission rate and loudness. The artificial representation of this phenomenon uses parameter setting that follows the original publication of Yang [9]. Maximal and minimal frequencies are set up $f_{\max} = 2,\ f_{\min} = 0$, respectively. A local-search *loudness* parameter is initialized $A_i = 1.2$ for each bat-individual and reduced if a new bat position is better than the old one using coefficient $\alpha = 0.9$. The *emission rate* parameter is initialized to each bat-individual $r_i = 0.1$ and increased by parameter $\gamma = 0.9$ in the case of a successful offspring.

The dispersive flies optimization algorithm (abbreviated DFO hereafter) was proposed in 2014 [10] by al Rifaie. The only control parameter called *disturbance*

threshold, is set to value from the recommended range $1 \times 10^{-2} < dt < 1 \times 10^{-4}$, i.e. $dt = 1 \times 10^{-3}$.

The cuckoo search algorithm (denoted Cuckoo) was introduced by Yang in 2009 [11]. This algorithm was inspired by cuckoo birds 'nest-parasitism'. Probability of the cuckoo's eggs laid in a bird-host nest is set $pa = 0.25$ and the control parameter of Lévy flight random walk is set to $\lambda = 1.5$.

The firefly algorithm (called Firefly in follows) proposed by Yang in 2008 [7] models the 'light-behavior' of fireflies when attracted another fireflies. This artificial representation of fireflies model has several control parameters that are set to recommended values – randomization parameter $\alpha = 0.5$, *light absorption* coefficient $\gamma = 1$, and *attractiveness* is updated using its initial $\beta_0 = 1$ and minimal $\beta_{\min} = 0.2$ values.

The only representative of the algorithms modeling the life of plants is Flower Pollination Algorithm for Global Optimization (denoted Flower hereafter) and was proposed by Yang in 2012 [12]. The goal of this approach is to model a process of transferring pollen grains between the flowers to their further reproduction. The main control parameter equals to probability of switching between global and local search is set to $p = 0.8$. A second parameter controlling Lévy distribution is set up $\lambda = 1.5$, as in the Cuckoo search algorithm.

The particle swarm optimization (PSO) originally proposed by Kenedy and Eberhart in 1995 belongs to very popular and ofen studied nature-inspired algorithms [13]. In this experiment, the basic variant of PSO with slightly enhanced of particles' velocities updated by the variation coefficient w and coefficient c is used. The variation control parameter w is set as a linear interpolation from maximal value $w_{\max} = 1$ to $w_{\min} = 0.3$, for each generation. Parameter controlling a local and a global part of the velocity update is set $c = 1.05$. A new velocity is computed by $\boldsymbol{v}_{i,G+1} = w_{G+1}\ \boldsymbol{v}_{i,G} + c\, U(0,1)\ (\boldsymbol{p}_{\text{best}} - \boldsymbol{x}_i) + c\, U(0,1)\ (\boldsymbol{g}_{\text{best}} - \boldsymbol{x}_i)$, where G denotes generation, $U(0,1)$ is random number generated from uniform distribution with parameters given in parentheses, $\boldsymbol{x}_i$ is current particle position, $\boldsymbol{p}_{\text{best}}$ is up-to-now best historical position of the current particle, and $\boldsymbol{g}_{\text{best}}$ is a position of the best particle in swarm history.

The self-organizing migrating algorithm (abbreviated SOMA) was proposed by Zelinka and Lampinen in 2000 as a model of a pack of predators [14]. SOMA has several control parameters and particle strategies that crucially influence the algorithm's efficiency. The best settings based on our preliminary experiments was taken for this experiment. Parameter controlling the (maximal) length of individual way toward to leader is set $PathLenght = 2$, the step size is set to $Step = 0.11$, and perturbation parameter is set $Prt = 0.1$. There are also several strategies of individual movement, the best performing strategy *all-to-one* as indicated in the preliminary experiments was applied to comparison on the CEC 2011 benchmark.

3 Cooperative Model of Nature-Inspired Algorithms

The main goal of this paper is to construct cooperative model of the aforementioned nature-inspired algorithms to achieve better efficiency. There are many possibilities how to employ selected k various algorithms to cooperation. We applied a modification of our well-performed recent cooperative model described in [15, 16]. A comprehensive review of a control parameters settings in a distributed evolutionary algorithms is in [17, 18]. The idea of the cooperative model is based on *migration model* with ring topology [16] and its pseudo-code is illustrated in Algorithm 1.

Algorithm 1 Cooperative Model of Nature-Inspired Algorithms

```
initialize nature-inspired algorithms' populations P_i, i = 1, 2, ..., k
evaluate individuals of all algorithms' populations
while stopping condition not reached do
  for i = 1, 2, ..., k do
    perform n_gen generations of ith algorithm's population
  end for
  construct a ring topology of randomly ordered algorithms' populations
  migrate selected individuals between populations by the unidirectional ring
end while
```

Proposed cooperative model has beside selected ring topology several input parameters. At first, k populations of equal size N_p is initialized and developed by k various algorithms. Then, n_{gen} generations of all nature-inspired algorithms are performed independently and several individuals are selected to exchange with other populations. This exchange is called *migration* and preliminary experiment [15] shows that combination of the best and n_{ind} randomly selected individuals is a good choice. Migration is performed between couple of populations such that selected the best individual from the donor population replaces the worst individual in the acceptor population. Randomly selected n_{ind} individuals from the donor population replaces n_{ind} randomly selected individuals in the acceptor population except the best individual.

The couples of populations to migration are given by ring topology where each population has two neighbors - preceding and following. For higher level of randomness, order of algorithms' populations in ring topology is given randomly for each migration. The populations are not communicated with the same counterparts for higher level of diversity of individuals in overall CoNI algorithm. Selected $n_{\text{ind}}+1$ individuals from the donor population (*preceding* in circle manner) replaces the selected individuals in acceptor population (*following* in circle manner).

A pseudo-parallel representation will be used to estimate efficiency of the proposed cooperative model. Physically, n_{gen} generations are performed subsequently for each algorithm on single-CPU PC (pseudo-parallelism). The quality

of the proposed cooperative model is evaluated by function value and also by number of function evaluations. The name of the proposed cooperative model of nature-inspired algorithms is abbreviated as CoNI in following text.

4 Experimental Setting

The test suite of 22 real-world problems selected for CEC 2011 competition in Special Session on Real-Parameter Numerical Optimization [6] is used as a benchmark in the experimental comparison. The functions in the benchmark differ in the computational complexity and in the dimension of the search space which varies from $D = 1$ to $D = 240$.

For each algorithm and problem, 25 independent runs were carried out. The run of the algorithm stops if the prescribed number of function evaluations $MaxFES = 150000$ is reached. The partial results of the algorithms after reaching one third and two thirds of $MaxFES$ were also recorded for further analysis. The point in the terminal population with the smallest function value is the solution of the problem found in the run.

The population size $N = 90$ was used in all the nature-inspired algorithms and CEC 2011 problems. The number of nature-inspired algorithms cooperative in CoNI is $k = 8$, the population size of each cooperative algorithm is set equally to $N_p = 15$, number of generations before migration is $n_{\text{gen}} = 10$ and $n_{\text{ind}} = 4$ individuals are randomly selected for each of migration. The other control parameters are set up according to recommendation of authors in their original papers. All the algorithms are implemented in Matlab 2010a and all computations were carried out on a standard PC with Windows 7, Intel(R) Core(TM)i7-4790 CPU 3.6 GHz, 16 GB RAM.

5 Results

A Table 1 contains the basic characteristics of CoNI algorithm at final stage of the search (FES= 150000) and the results of the Kruskal-Wallis test including significance and multiple comparison based on Dunn's method. The detailed results of the original nature-inspired algorithms on CEC 2011 real-world problems used in this experiment are presented in previous works [4, 5]. The median value for the problem (row) where CoNI algorithm achieves the best result out of all algorithms in comparison is printed bold.

Kruskal-Wallis non-parametric one-way ANOVA test was applied to each problem to obtain significant differences. It was found that the performance of the algorithms in comparison differs significantly, the null hypothesis on the same performance is rejected in all the problems at all dimensions with achieved significance level $p < 1 \times 10^{-5}$ and it means that algorithms' performance differs even in the similar medians.

The best performing algorithms significantly different from the followers and mutually with no significant differences are listed in the column "best" ordered

Table 1. The basic characteristics of function values found by the cooperative model and results of Kruskal-Wallis multiple comparison

F	D	min	max	med	mean	std	best	worst
T01	6	1.18E-06	14.779	8.41626	6.92583	5.98041	Cuckoo,CoNI	Bat,DFO
T02	30	-25.5936	-15.3846	**-22.0875**	-21.5095	3.13850	CoNI, ABC,SOMA	Bat,DFO
T03	1	1.15E-05	1.15E-05	1.15E-05	1.15E-05	5.19E-21	all	Firefly
T04	1	0	0	0	0	0	all	Firefly
T05	30	-35.7924	-31.4839	-34.1075	-33.5094	1.29947	ABC,CoNI	DFO,Bat
T06	30	-29.1627	-21.2696	-27.4277	-26.5641	2.58163	CoNI,SOMA	DFO,Bat, Firefly
T07	20	0.807308	1.3313	**1.04799**	1.0488101	1.54E-01	CoNI,SOMA, Cuckoo	DFO,Bat, Firefly
T08	7	220	220	220	220	0	all	Bat,Firefly, DFO
T09	126	13082.7	36124.4	19363.4	21467.9	6618.66	ABC,CoNI	Firefly
T10	12	-20.8443	-12.773	-19.2706	-18.8992	2.00284	Cuckoo,CoNI	Bat,Firefly, DFO
T11.1	120	62744.9	212805	71228.1	79231.1	28762.1	ABC,CoNI	DFO,Firefly
T11.2	240	1.10E+06	1.17E+06	**1.12E+06**	1.12E+06	17712.3	CoNI,ABC	Bat,Firefly, DFO
T11.3	6	15445.5	15453.4	15447.6	15448.2	2.11792	Flower	Bat,Firefly, DFO
T11.4	13	18485.2	19148.6	18820	18853.86	155.764	Flower	ABC,Bat, DFO
T11.5	15	32781.8	32984.7	32878	32872.0	51.1892	SOMA,CoNI, Flower	Bat,Firefly
T11.6	40	129038	137573	133039	133312	2570.42	Flower,CoNI, SOMA	Firefly,DFO
T11.7	140	1.92E+06	2.54E+06	**1.95E+06**	2.01E+06	139086	CoNI,Flower, SOMA	Firefly, DFO,Bat
T11.8	96	941250	1.02E+06	**946333**	951888	16253.6	CoNI,SOMA	Firefly,Bat
T11.9	96	1.00E+06	1.84E+06	1.43E+06	1.42E+06	176918	SOMA, CoNI,PSO	Firefly,Bat
T11.10	96	941689	1.14E+06	**945565**	962950	45817.2	CoNI,SOMA	Firefly,Bat
T12	26	12.4059	20.3057	**16.7507**	17.0220	1.96866	CoNI,SOMA	DFO,Bat
T13	22	11.5376	26.6287	21.4875	20.5451	3.78682	ABC,CoNI, SOMA	Bat,DFO

ascending with respect to the median function value. The worst performing algorithms significantly different from their predecessors and mutually with no significant differences are listed in the column "worst" ordered from the worst performing algorithm. Based on these columns, it is not easy to assess the superiority or inferiority of the algorithms. In the case of the proposed CoNI, the first position (column *best*) is not occupied only for the problems $T11.3$ and $T11.4$. Comparing medians of these problems it is obvious that CoNI takes at least the third position out of nine algorithms.

For better overview of the comparison of the presented algorithms' performance, the number of first, second, third, and the last positions from Kruskal-Wallis test are computed and showed in Table 2. It is clear that CoNI is able to achieve the first position in 8 out of 22 real-world problems. In the remaining problems, CoNI occupies the second or the third position without significant difference between CoNI and the best performing counterpart (based on Kruskal-Wallis test). Further promising results provide ABC, Flower, SOMA,

Table 2. Number of significant wins, second, third, and the last positions of the algorithms

Position	ABC	Bat	Cuckoo	DFO	Firefly	Flower	PSO	SOMA	CoNI
1st	4	0	2	0	0	3	0	2	8
2nd	2	0	1	0	0	1	1	5	9
3rd	0	0	3	0	0	2	5	7	2
last	1	8	0	5	8	0	0	0	0

and Cuckoo. An interesting is significant win of Flower algorithm in $T11.3$ and $T11.4$ problems. The worst performing algorithms in whole experiment are Firefly and Bat algorithm followed by DFO. Necessary to note that Firefly algorithm performs substantially better when solving an artificial problems as CEC 2014 (see results in [5]).

In Table 3 the results of Friedman test are presented. This test was carried out on medians of minimal function values at three stages of the search, namely after $FES = 50000, 100000$, and 150000. The null hypothesis on equivalent efficiency of the algorithms was rejected at the all stages of the search, p-value achieved in Friedman test was less than 5×10^{-6}.

Table 3. Mean rank of Friedman test for all algorithms

alg	1st stage	2nd stage	3rd stage	avg
CoNI	2.8	2.2	2.1	2.3
SOMA	2.7	3.0	3.0	2.9
PSO	4.3	4.2	4.4	4.3
ABC	3.9	4.5	4.6	4.3
Cuckoo	4.7	4.3	4.0	4.3
Flower	4.6	4.4	4.3	4.4
DFO	7.0	6.9	7.1	7.0
Firefly	7.6	7.6	7.7	7.6
Bat	7.6	7.8	7.9	7.8

The mean ranks from Friedman test of the algorithms in three stages are also illustrated in Fig. 1. Moreover, the mean rank values for each algorithm of three stages are joined for better conclusions. Notice that better performing algorithm over all 22 test problems achieves smaller mean rank and vice versa. Based on these results (especially a graphical representation) three groups of compared algorithms with respect to performance are arisen. The worst performing triplet is formed by DFO, Firefly, and Bat algorithm. All these nature-inspired algorithms are often used by researchers to solve the real problems. The "middle-performing" group is formed by PSO, Flower, Cuckoo, and ABC algo-

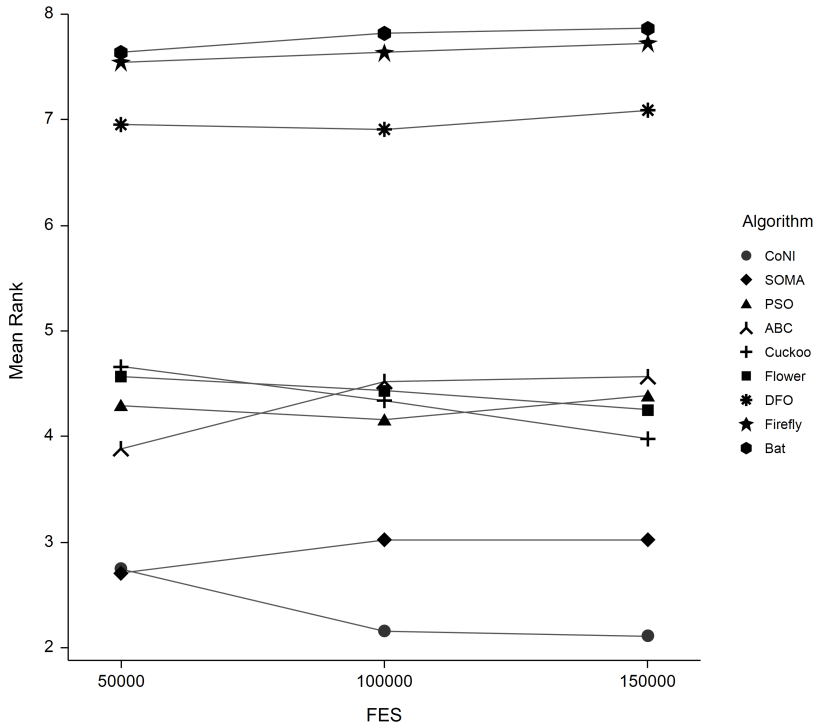


Fig. 1. Mean rank of Friedman test for all algorithms

rithm where performance of ABC and PSO is gradually in the search process surprisingly decreased. Mean ranks of these algorithms are approximately equal to the average rank. Whilst performance of SOMA in the best couple is gradually decreased with increasing *FES*, CoNI with increasing function evaluations achieved less mean rank.

Performance of new CoNI algorithm should be also compared with the performance of the winner of CEC 2011 competition, GA-MPC. Detailed results of this algorithm are provided in [19]. It is clear that CoNI is able to outperform GA-MPC in 5 problems, i.e. T04, T11.6-8, and T11.10 and in two problems (T03 and T08) both algorithms perform equally. In the remaining cases, the CEC 2011 winner performs better than cooperative model of nature-inspired algorithms.

Achieved results of this experimental study show that proposed cooperative model of nature-inspired algorithm is able to outperform the original algorithms and also partially the CEC 2011 winner. Further analysis of CoNI features should detect if some of the used nature-inspired algorithms performs better or worse. For this purpose, the number of successfully generated new individuals are computed for each of eight nature-inspired algorithms in CoNI. This characteristic denotes the number of newly generated individuals better than the old solution

in accordance with the goal function. For better comparison, a percentage success of each (ith) used algorithm is computed as a proportion of its success (suc_i) from whole success (suc_{total}):

$$psuc_i = \frac{suc_i}{suc_{\text{total}}} \cdot 100, \qquad i = 1,\ 2, \ldots k \tag{1}$$

For better comprehensibility, box-plot of the percentage successes of all algorithms is depicted in Fig. 2.

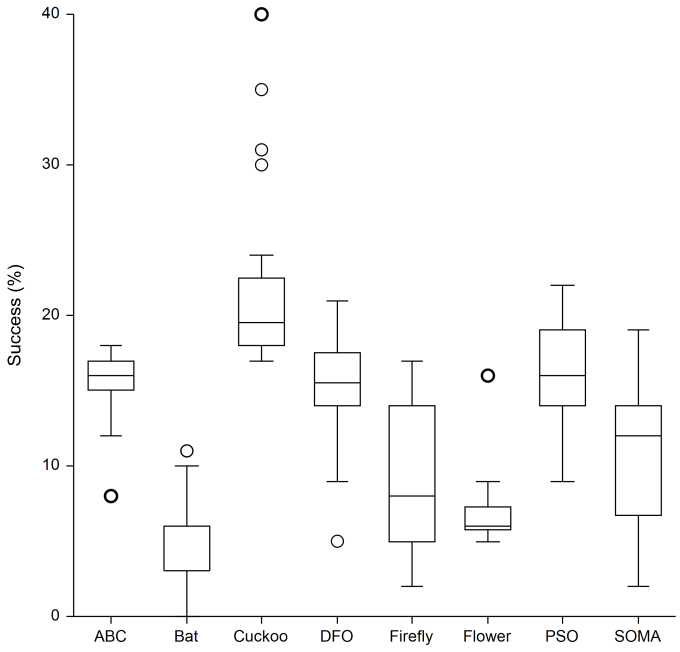


Fig. 2. Comparison of percentage successes (*psuc*) in CoNI algorithm

From the box-plot in Fig. 2, it is clear that the biggest proportion of successfully generated new-individuals provides Cuckoo algorithm with ABC and surprisingly DFO algorithm. This situation is caused by very simple reason. The worst individual of current algorithm in CoNI is replaced by the selected best individual (from another algorithm) and sometimes this new solution could be substantially better than current best solution (case of DFO). Then, some individuals of such algorithm will be more increased because of using of the new best solution. Thin boxes in the bottom of the plot for PSO and Flower suggest lower success-proportion in most of the real-world problems.

Except the percentage successes, the "name" of the nature-inspired algorithm employing the overall best solution of CoNI is stored in 17 stages of CEC 2011

problems. Because the lack of space, total number of "ownership" of the best solution in 17 stages of 22 test problems is computed: ABC (112), SOMA (79), Bat (70), Firefly (62), Cuckoo (34), Flower (11), PSO (3), and DFO (3). We can see that the most often algorithm providing the overall best solution of CoNI is ABC. This result is in contradiction with the mean rank of ABC algorithm (Fig. 1) and the success of this algorithm in CoNI (Fig. 2). The least counts of developed the best CoNI solution have DFO and PSO algorithms. When we consider presented results (Table 2 and Fig. 2), the performance of both these algorithms is rather less, especially for DFO variant. However, the percentage success of PSO algorithm in CoNI model belongs to better foursome. The phenomenon, when badly performing separately applied nature-inspired algorithm has high success in CoNI model is caused by big number of small improvements.

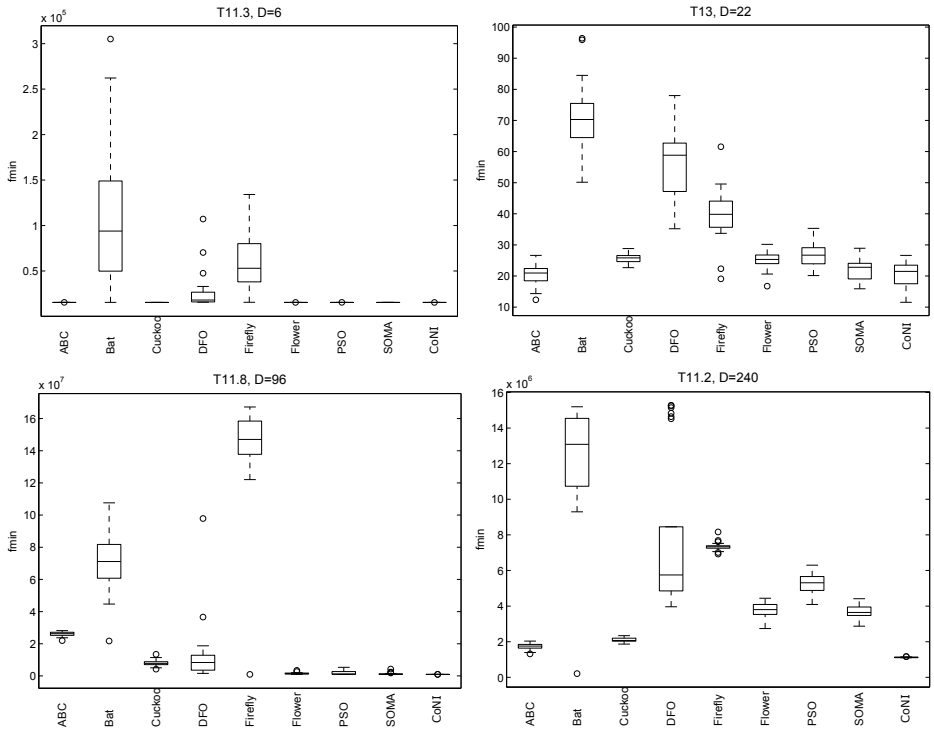


Fig. 3. Minimal function values from 25 runs in problems with various dimensionality

The various performance of nature-inspired algorithms is the main reason for using of the cooperative model because higher variability of partial solutions causes higher ability to produce more diverse individuals. The diversity of in-

dividuals is crucial feature of evolutionary algorithms in the search of the right solution of optimization problems.

The performance of CoNI algorithm in problems with various dimension level is depicted in Fig. 3. The $T11.3$ problem ($D = 6$) is solved successfully by most algorithms (boxes are flattened at the bottom of the figure). For higher dimensions (problems $T13$, $T11.8$), the better performance of CoNI is more obvious. Furthermore, for the problem with highest dimension, $T11.2$, ($D = 240$) CoNI achieves the best results among all algorithms in comparison.

6 Conclusion

In this paper the cooperation model of several algorithms based on well-known migration model is proposed to increase the efficiency of nature-inspired algorithms. The results of experimental comparison of eight popular nature-inspired algorithms with the cooperative model of these algorithms demonstrate clearly the superiority of the proposed CoNI algorithm. Good performance of CoNI is caused by exchange of the individuals between variously performing algorithms. A proper settings of the parameters of cooperative model promises better results in various real-world problems.

Although nature-inspired algorithms belong to very popular optimization methods, their efficiency is often poor as it is shown in results of [4, 5]. When we consider a No-Free-Lunch theorem [20] – each algorithm is better performing in another kind of optimization tasks – cooperative model of nature-inspired algorithms is simple idea to achieve better results. The high performance of CoNI is caused by exchange the individuals of variously successful applied algorithms in various problems. Without the migration, the results are not better than the results of the best non-parallel nature-inspired algorithm. Proposed cooperative algorithm shares the best position in 20 out of 22 problems and achieves the best results (first position) in 8 out 22 test problems.

Comparison of CoNI with the winner of CEC 2011 real-world problems test suite – (GA-MPC [19]) shows that cooperative model is competitive in most of real problems. We believe that there exist possibilities of the improvements. Especially selection of proper topology, migration policy, and migration frequency are the fundamental points for improvement of the cooperative model [17, 18].

The source code of newly proposed CoNI algorithm in Matlab is available at `www1.osu.cz/~bujok/`, the source code of some other state-of-the-art nature-inspired algorithms can be found on web site of MathWorks, `www.mathworks.com`.

References

1. Fister Jr., I., Yang, X.S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. Elektrotehniski vestnik **80**(3) (2013) 116–122
2. Wang, H., Sun, H., Li, C., Rahnamayan, S., Pan, J.S.: Diversity enhanced particle swarm optimization with neighborhood search. Information Sciences **223** (2013) 119–135

3. Yang, M., Li, C., Cai, Z., Guan, J.: Differential evolution with auto-enhanced population diversity. IEEE Transactions on Cybernetics **45**(2) (2015) 302–315
4. Bujok, P., Tvrdík, J., Poláková, R.: Nature-inspired algorithms in real-world optimization problems. MENDEL Soft Computing Journal **23** (2017) 7–14
5. Bujok, P., Tvrdík, J., Poláková, R.: Adaptive differential evolution vs. nature-inspired algorithms: An experimental comparison. In: 2017 IEEE Symposium Series on Computational Intelligence (IEEE SSCI). (2017) 2604–2611
6. Das, S., Suganthan, P.N.: Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Technical report, Jadavpur University, India and Nanyang Technological University, Singapore (2010)
7. Yang, X.S.: Nature-Inspired Optimization Algorithms. Elsevier (2014)
8. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, Kayseri, Turkey (2005)
9. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In Gonzalez, J., Pelta, D., Cruz, C., Terrazas, G., Krasnogor, N., eds.: Nicso 2010: Nature Inspired Cooperative Strategies for Optimization. Volume 284 of Studies in Computational Intelligence., Univ Laguna; Carnary Govt; Spanish Govt (2010) 65–74 International Workshop on Nature Inspired Cooperative Strategies for Optimization NICSO 2008, Tenerife, Spain, 2008.
10. al Rifaie, M.M.: Dispersive flies optimisation. In: Federated Conference on Computer Science and Information Systemss, 2014. Volume 2 of ACSIS-Annals of Computer Science and Information Systems. (2014) 529–538
11. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: 2009 World Congress on Nature Biologically Inspired Computing NaBIC. (2009) 210–214
12. Yang, X.S.: Flower pollination algorithm for global optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7445 LNCS** (2012) 240–249
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: 1995 IEEE International Conference on Neural Networks Proceedings, Vols 1-6, IEEE, Neural Networks Council (1995) 1942–1948
14. Zelinka, I., Lampinen, J.: SOMA – self organizing migrating algorithm. In Matousek, R., ed.: MENDEL, 6th International Conference On Soft Computing, Brno, Czech Republic. (2000) 177–187
15. Bujok, P., Tvrdík, J.: Parallel migration model employing various adaptive variants of differential evolution. In: Swarm and Evolutionary Computation. Volume 7269 of Lecture Notes in Computer Science., Springer-Verlag Berlin (2012) 39–47
16. Bujok, P.: Synchronous and asynchronous migration in adaptive differential evolution algorithms. Neural Network World **23**(1) (2013) 17–30
17. Laessig, J., Sudholt, D.: Design and analysis of migration in parallel evolutionary algorithms. Soft Computing **17**(7, SI) (2013) 1121–1144
18. Gong, Y.J., Chen, W.N., Zhan, Z.H., Zhang, J., Li, Y., Zhang, Q., Li, J.J.: Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Applied Soft Computing **34** (2015) 286–300
19. Elsayed, S.M., Sarker, R.A., Essam, D.L.: GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems. In: 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE (2011) 1034–1040
20. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67–82

Low-cost optimization under uncertainty through box representation of robustness measures

Mickael Rivier[1,2] and Pietro Marco Congedo[1]

[1] Team CARDAMOM, Inria Bordeaux Sud-Ouest, Talence, 33405 Cedex, France
mickael.rivier@inria.fr
[2] ArianeGroup, Le Haillan, 33185 Cedex, France
mickael.rivier@ariane.group

Abstract. This work is devoted to tackle multi-objective optimization under uncertainty problems using the SABBa framework (Surrogate-Assisted Bounding-Box approach). It is based on the coupling of the Bounding-Box approach and a surrogate-assisting strategy. It aims at efficiently dealing with robust optimization problems with approximated robustness measures. Here, some developments are introduced, which are focused on the treatment of reliability constraints and the computation of the probabilistic Hausdorff distance to analytical optima. Finally objective measures probability distribution are formulated and their performance assessed and compared with conservative boxes.

Keywords: Multi-objective optimization, Uncertainty-based optimization, Error boxes, Surrogate model

1 Introduction to SABBa framework

The basic ingredient of the SABBa framework is the Bounding-Box (BB) approach, which has been formulated in [1] and [2]. A Bounding-Box (also called here as conservative box) is defined as a multi-dimensional product of intervals centered on approximated objectives and containing the associated true values. In SABBa, it is supplemented with a surrogate-assisting strategy, which is very effective in order to reduce the overall computational cost associated to the BB approach during the last iterations of the optimization. Note that SABBa is applicable regardless of specific optimization and uncertainty quantification methods and can be then used with whatever state-of-the-art methods.

In [3], it has been shown that this framework is a robust strategy i.e. true optimal designs are never discarded and are refined within targeted accuracy. Interpolated designs from the surrogate-assisting strategy can be used through the redefinition of the objective functions and the choice of the size of error boxes.

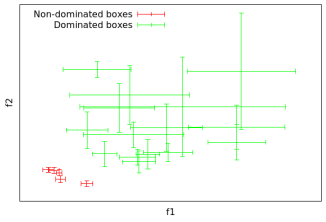


Fig. 1: Bounding-Boxes

The SABBa framework yields a strong coupling between the costly convergence of the UQ process and the quality of the associated design. Fig. 1 gives an example of such dominated and non-dominated boxes. This coupling heavily relies on the definition of Boxed Pareto dominance. Intuitively, a box $\mathcal{B}_1$ is dominated when the worst outcome of another box $\mathcal{B}_2$ dominates in the classical sense the best point of $\mathcal{B}_1$.

2 Extension to constrained optimization problem

This paper is devoted to the development of SABBa in order to tackle constrained optimization problems. This implies a redefinition of the Boxed Pareto dominance, which is formulated as follows. A box $\mathcal{B}_1$ can only be dominated by another box $\mathcal{B}_2$ if any outcome of the box $\mathcal{B}_2$ in the dimension of reliability measures satisfies the associated constraint.

The behavior of the uncertainty quantification process is studied both in separated uncertainty dimensions only) and coupled (design and uncertainty dimensions) spaces. While coupled spaces may allow fewer refinements in areas where the surrogate-assisting model is not yet converged, it increases the dimensionality of the problem and may slow the convergence. Recently, coupled spaces has received much attention, such as [4], because of its intrinsic capability of correlating quantity of interest in the design dimensions.

A preliminary result from the classical SABBa framework is presented here. The test-case studied is the following bi-objective problem :

$$\begin{aligned}
\text{minimize:} \quad & \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} \mu(u) \\ \sigma^2(u) \end{pmatrix} \\
\text{by changing:} \quad & (x_1, x_2) \in [1,2] \times [1,2] \\
\text{with:} \quad & u(\boldsymbol{x}, \xi) = \xi - x_1 \xi^5 + \cos(2\pi x_2 \xi) + 5 \\
\text{and:} \quad & \xi \sim \mathcal{U}([0,1])
\end{aligned}$$

In order to compare strategies from a quantitative point of view, the convergence of the algorithm is measured on analytical test-cases by means of Hausdorff distance to the Pareto optimal set. As the set of interest is here a set of boxes in the objective space, the modified Hausdorff distance is extended to aleatory Pareto optimal design by considering boxes as uniform distribution on the robustness measures. The indicator of interest is then the following expected modified Hausdorff $\mathbb{E}_{\mathcal{X}_{\widetilde{\mathcal{P}}}}\left[d'_H(\mathcal{X}_{\mathcal{P}}, \mathcal{X}_{\widetilde{\mathcal{P}}})\right]$ with:

$$d'_H(\mathcal{A}, \mathcal{B}) = \max\Big(\frac{1}{N_{\mathcal{A}}} \sum_{a \in \mathcal{A}} \min_{b \in \mathcal{B}} d(a,b), \frac{1}{N_{\mathcal{B}}} \sum_{b \in \mathcal{B}} \min_{a \in \mathcal{A}} d(a,b)\Big)$$

The following Fig. 2 show the convergence of the expected modified Hausdorff distance between the SABBa framework in coupled or separated spaces and the use of an a priori metamodel built in the coupled spaces. One can see here both

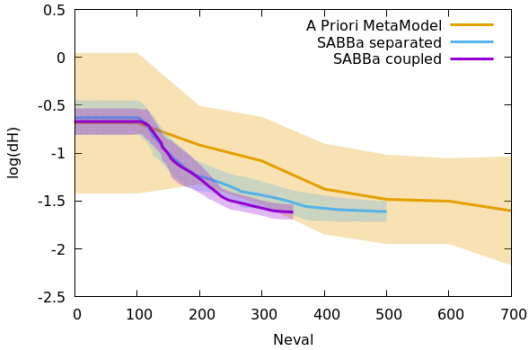


Fig. 2: Convergence curves

a quicker convergence of the SABBa framework and a significant reduction of the variability in terms of the $\mu \pm \sigma$ area.

Note that the SABBa framework is based on the computation of conservative error boxes. These boxes have a high chance of overestimating the true variability of the robustness measures. This is due to both the conservative paradigm and the assumption of uniform distribution within the boxes.

In the final paper, a cheaper but more intrusive version of the framework will be presented in details, where a Gaussian Process model in the uncertainty space may allow for the computation of the robustness measures joint distribution. These distributions could then be taken as correlated gaussian input uncertainties, as an extension of [5], in order to perform multi-objective bayesian optimization in the objective space. Refinement in this context should allow both uncertainty quantification on a new design and UQ refinement on a previous one.

This strategy will be validated first in separated spaces as explained above but may be extended to a coupled space formulation.

References

1. M. Mlakar, T. Tusar, B. Filipic, Comparing solutions under uncertainty in multi-objective optimization, Mathematical Problems in Engineering 2014 (2014) 1-10
2. F. Fusi, P.M. Congedo, An adaptive strategy on the error of the objective functions for uncertainty-based derivative-free optimization, Journal of Computational Physics 309 (2016) 241-266
3. M. Rivier, P.M. Congedo, Surrogate-assisted bounding-box approach for optimization problems with approximated objectives, RR-INRIA
4. J. Zhang, A.A. Taflanidis, J.C. Medina, Sequential approximate optimization for design under uncertainty problems utilizing Kriging metamodeling in augmented input space, In Computer Methods in Applied Mechanics and Engineering, Volume 315, 2017, Pages 369-395, ISSN 0045-7825
5. R. Le Riche, V. Picheny, Gears design with shape uncertainties using controlled monte carlo simulations and kriging, 50th AIAA/ASME.ASCE/AHS/ASC Structures, Structural Dynamics, and Material Conference, (AIAA Paper 2009-2257)

A New Binary Encoding Scheme in Genetic Algorithm for Solving the Capacitated Vehicle Routing Problem

Stanley Jefferson de A. Lima[0000−0003−0483−9535] and Sidnei Alves de Araújo[0000−0003−3970−5801]

Informatics and Knowledge Management Graduate Program,
Universidade Nove de Julho – UNINOVE, São Paulo, SP, Brazil.
stanleyjefferson@outlook.com, saraujo@uni9.pro.br,
WWW home page:http://www.uninove.br

Abstract. In the last decades the Vehicle Routing Problem (VRP) and its ramifications, including the Capacitated Vehicle Routing Problem (CVRP), have attracted the attention of researchers mainly because their presence in many practical situations. Due to the difficulties encountered in their solutions, such problems are usually solved by means of heuristic and metaheuristics algorithms, among which is the Genetic Algorithm (GA). The solution of CVRP using GA requires a solution encoding step, which demands a special care to avoid high computational cost and to ensure population diversity that is essential for the convergence of GA to global optimal or sub-optimal solutions. In this work, we investigated a new binary encoding scheme employed by GA for solving the CVRP. Conducted experiments demonstrated that the proposed binary encoding is able to provide good solutions and is suitable for practical applications that require low computational cost.

Keywords: Genetic Algorithm, Solution Encoding, Chromosome Representation, Capacitated Vehicle Routing Problem.

1 Introduction

Optimization of the logistics system has become one of the most important aspects of the supply chain during the last three decades [1]. In this context, many researchers have invested their efforts in solving various problems in this segment, among them is the Vehicle Routing Problem (VRP).

In general, the VRP consists in defining the routes that a set of vehicles must follow to supply the demand of certain customers, respecting the operational restrictions imposed by the context that they are inserted. The most common objectives of the VRP are minimize the total distance traveled, improve the transport time, minimize the number of vehicles needed and reduce the total cost of the routes [2]. One of the main ramifications of the VRP is the Capacitated Vehicle Routing Problem (CVRP), which is considered in this work and explained in detail in section 2.1.

In the literature there are several proposals to solve the VRP (and its ramifications) using different heuristic and meta-heuristics techniques, among which are: Tabu Search, Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, Particle Swarm Optimization, Variable Neighborhood Search, and Hybrid Meta-Heuristics [3]. The Genetic Algorithm (GA) stand out by its versatility of construction and the good results that it has been demonstrated in solving complex problems, including VRP, as can be seen in Lau et al. [4]; Bermudez et al. [5]; Wang and Lu [6]; Lee and Nazif [2]; Tasan and Gen [7]; Ursani et al. [8]; Lu and Yu [9]; Kuo, Zulvia and Suryadi [10]; Vidal et al. [11]; Reiter and Gutjahr [12]; Osaba, Diaz and Oniera [13] and Lima et al [14].

A trivial way of encoding solutions for the VRP using GA is through a three-dimensional binary matrix in which the rows are associated with the vehicles, the columns with the costumers and the depth with the visitation order. However, this encoding scheme demands high computational cost and may be inefficient in terms of population diversity, which is essential to promote the convergence to global optimum or sub-optimal solutions. Thus, many studies found in the literature has shown concern about how to encode VRP solutions.

In this context, and differently from all above mentioned works which explore improvements in the heuristic and meta-heuristic algorithms, this work is focused in more efficient ways to encode solutions in GA. Specifically, we are proposing a new binary encoding scheme in GA for Solving the CVRP, which constitutes the main contribution of this work.

2 Theorectical Background

2.1 Capacitated Vehicle Routing Problem (CVRP)

The CVRP is one of the most basic version of VRP. In this problem all customers have their demands previously defined which must be attend entirely by a fleet of homogeneous vehicles, all of them running from only distribution center. In the CVRP, just the vehicle capacity restriction is imposed [15], that is, the sum of the demand of all customers belonging to a route does not exceed the capacity of vehicle used to execute that route. Figure 1 illustrates an example of CVRP, which envolves two vehicles for meeting the demands of eighteen geographically dispersed customers.

Let be $G = (V, E)$ a graph in which $V = 0...n$ is the set of vertices that represent the customers and E the set of edges, representing the paths connecting the customers to each other and to the distribution center. Each edge (v_i, vj) has associated a cost C_{ij} of the path between the costumers represented by vertices i and j. When $C_{ij} = C_{ji}$, the problem is known as symmetrical, otherwise the problem is identified as asymmetrical. A set of K identical vehicles with capacity c_v is allocated to the distribution center. For each customer v is associated a demand d_v, and for the distribution center is defined $d_0 = 0$.

In summary, the CVRP consists of finding a set of routes, where each route is traveled by a vehicle, with the objective to minimize the total cost of the

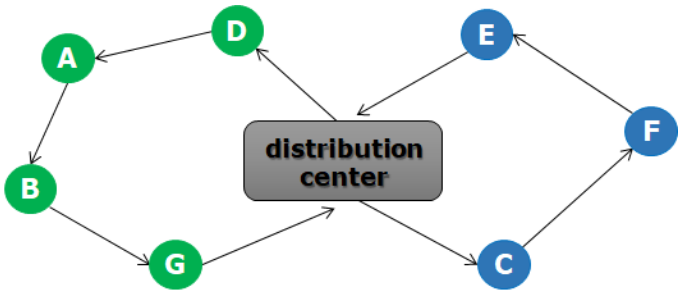


Fig. 1. Example of routing with the vehicles starting from a distribution center.

routes (TC), respecting the following restrictions: (1) each route must start and finish at the distribution center; (2) each customer must be visited just only time and (3) the sum of the customers' demands included in a route cannot exceed the vehicle's capacity. According to Vieira [3] the CVRP can be mathematically formulated as follows:

$$\text{Minimize } TC = \sum_{i=0}^{nc} \sum_{j=0, j\neq i}^{nc} \sum_{k=1}^{K} C_{ij} x_{ijk} \tag{1}$$

$$\text{Subject to } \sum_{k=1}^{K} \sum_{j=1}^{nc} x_{0jk} \leq K \tag{2}$$

$$\sum_{j=1}^{nc} x_{0jk} = \sum_{j=1}^{nc} x_{j0k} = 1, k = 1, ..., K \tag{3}$$

$$\sum_{k=1}^{K} \sum_{j=0}^{nc} x_{ijk} = 1, i = 1, ..., nc \tag{4}$$

$$\sum_{j=0}^{nc} x_{ijk} - \sum_{j=0}^{nc} x_{ijk} = 0, k = 1, ..., K i = 1, ..., nc \tag{5}$$

$$\sum_{k=1}^{K} \sum_{i\in S} \sum_{j\in S} x_{ijk} \leq |S| - v(S), \forall S \subseteq V/\{0\}, |S| \geq 2 \tag{6}$$

$$\sum_{i=1}^{nc} d_i \sum_{i=0, j\neq i} x_{ijk} \leq cv, k = 1, ..., K \tag{7}$$

$$x_{ijk} \in \{0, 1\}, i = 1, ..., nc, j = 1, ..., nc, k = 1, ..., K \tag{8}$$

where: d_i is the demand of customer i; k: vehicle; K: set of vehicles; S: set of customers; nc: Number of customers; $v(S)$: Minimum number of vehicles

to attend S; cv: Capacity of vehicles; c_{ij}: cost of the path from customer i to customer j; TC: total cost of the routes; x_{ijk}: path from customer i to customer j with vehicle k;

The Equation 2 ensures that K vehicles will be used, while the Equation 3 guarantees that each route has its beginning and ending at the distribution center. Equation 4 defines that customers must be attended exactly one time and the equation 5 keeps the flow ensuring that a vehicle arrives at a customer and out of it, preventing that the route ends prematurely. The Equation 6 prevents the formulation of routes that do not include the distribution center. In this restriction, $v(S)$ represents the minimum number of vehicles required to attend a set of customers S. To ensure that the number of vehicles used to attend the customers of set S is not less than $v(S)$, the restriction 6 establishes, indirectly, that the capacity of the vehicle is not exceeded. However, to let this explicit, the Equation 7 is used to formulate the capacity restriction.

Finally, the Equation 9 is used to evaluate the solutions generated by GA. It reflects the value of the objective function (OF) or fitness and involves the number of vehicles used in the solution, violated restrictions (Equations 2 to 7) and the total cost of routes (Equation 1).

$$OF = TC + KW_v + nrW_r \tag{9}$$

where: W_v is the weight assigned to the number of vehicles used in the solution; nr is the number of violated restrictions and W_r is the weight given to the violated restrictions.

2.2 Genetic Algorithm (GA)

The GA is an evolutionary computational technique that simulates the mechanisms of natural selection, genetics and evolution. In the last decades it has been employed in several applications to solve complex optimization problems. Its bias is how much better an individual adapts to its environment, the greater their chances of surviving and generating offspring [16]. A GA individual represents a solution to the problem being solved. Each individual is defined as a chromosome, consisting of genes, which represent variables of the problem, and each position of a gene is defined as an allele.

In GA, the crossover operation consists in recombination of genes from selected individuals, responsible to reproduce descendants more adapted to the next generation. After a certain number of generations, it is common to ocurr the loss of population diversity, which results in the premature stopping of the GA leading to local optimum solutions. To avoid this problem, the mutation is applied at a given rate of individuals (usually by randomly changing the alleles), aiming to change the characteristic of the genes [17].

Other concepts associated with GAs are:

Genotype: is related to the population in the computation space, in which the solutions are represented to be easily understood and manipulated by computers [18][19].

Phenotype: is related to the population in the real world solution space, in which the solutions are represented to be interpreted in real world situations [18][19].

Encoding and Decoding: in the most cases, the phenotype and genotype spaces are different. Encoding is an operation that transforms a solution from the phenotype to genotype space, while decoding is responsible by transforming a solution from the genotype to the phenotype space. The main ecoding schemes are: Binary, Value (integer, float, string, etc), Permutation and Tree [19] [20]. Since these operations are carried out repeatedly during the fitness value calculation (evaluation) in a GA, they need to be simple and fast.

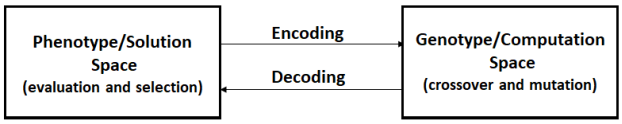


Fig. 2. Encoding/Decoding operations.

Based on the above explanation, it is observed that the encoding solution scheme is an important step in the development of GA, since it is directly related to the quality of the solutions found, as well as the computational time spent to find them.

2.3 Some Recent Encoding Schemes Used in GA for Solving the VRP and Its Ramifications

As observed in the recent literature, the most commonly used schemes for VRP solution encoding are permutation and value (integer). Such representations can be found, for example, in the works of Lu and Yu [9], Lau et al. [4], Lee and Nazif [2], Bermudez et al. [5] and Lima et al. [14].

Lu and Yu [9] proposed a simple mixed encoding scheme (permutation and integer), illustrated in Figure 3, which encapsulates a main chromosome (permutation of costumers) and a 'subchromosome' formed by integer values representing the number of customers on each route.

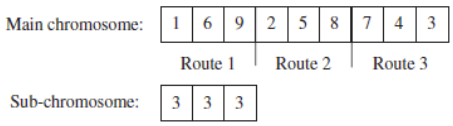


Fig. 3. Encoding scheme proposed by Lu and Yu [9].

Lau et al. [4] adopted an encoding scheme similar to that presented in [9], for the Multidepot VRP, in which the subchromosome (first chromosome positions) represents the number of costumers that each vehicle must attend, as well as the depot each vehicle will depart to make deliveries (see Figure 4).

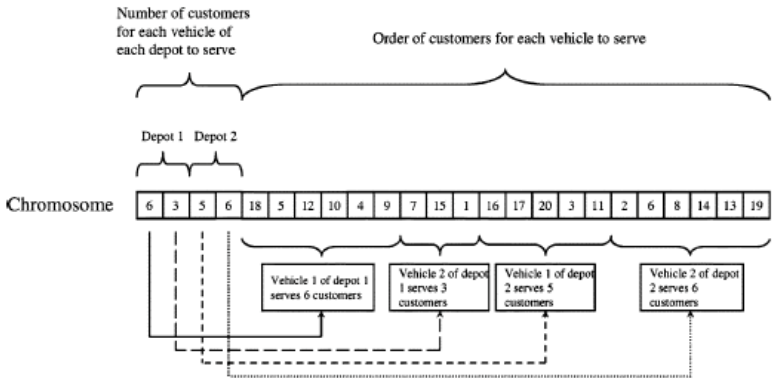


Fig. 4. Encoding scheme proposed by Lau et al. [4].

It is important to note that in these cases additional mechanisms must be used for route separation and also to bypass the problem of generating non-feasible solutions after applying the crossover and mutation operators. However, this latter mechanism is not clearly described in the above mentioned works.

Recently, Lima et al. [14] proposed a short binary encoding scheme for CVRP in which the chromosome represents the set of customers that must be attended by each vehicle, as can be seen in Figure 5, while the sequence for visiting the customers is solved by the Nearest Neighbor algorithm.

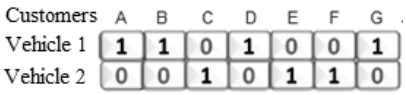


Fig. 5. Binary encoding scheme proposed by Lima et al. [14].

In this encoding scheme the alleles with value of 1 indicate the customers that will be attended by a vehicle, that is represented by each line of the binary matrix.

3 Materials and Methods

In the development of proposed encoding scheme, the programming language C/C++ and GAlib library [21] were used. The GAlib is a free library widely used for solving combinatorial optimization problems. For evaluating the proposal, experiments were performed and the results obtained were compared with the best results found in the literature for a set of instances extracted from Christofides and TSPLIB libraries, with up to 30 customers.

In the experiments we employed a desktop computer with the following configurations: Intel Celeron 2955U1.40 GHz processor; 4GB of RAM; Windows 7 Ultimate 32-bits operating system.

The following parameters (empirically defined) were employed by implemented GA:

- Population Size = 1200;
- Number of Generations (used as stop criterion) = 5000;
- Population rate of replacement = 0.8;
- Elitism rate = 0.2;
- Crossover rate = 0.8;
- Selection Method = Roulette;
- Mutation rate = 0.01;
- Type of Mutation = Flip Bit.

4 Proposed Binary Encoding Scheme

The encoding scheme proposed in this work consists of a binary matrix of $M = nc * 2 - 1$ columns by K rows. The example shown in Figure 6 illustrates the solution encoding of a CVPR instance that envolves $nc = 7$ costumers and $K = 2$ vehicles (as depicted in Figure 1). The first nc columns of each row indicate the costumers to be served by a vehicle, while the last $nc - 1$ columns consist of a vector that indicates the permutations to be made in a matrix of integers (Figure 7), called permutation matrix, representing the order that the costumers will be visited by the K vehicles.

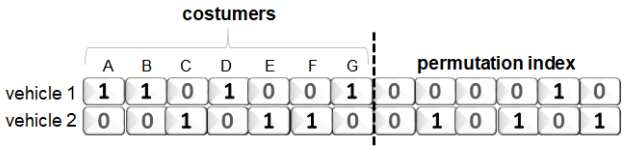


Fig. 6. Proposed binary encoding scheme for CVPR.

The permutation matrix (that can be understood as a seed) containing nc columns and K rows, shown in Figure 7, is unique and must be generated before the execution of GA using random permutations or some heuristic algorithm.

Thus, when it is combined with a GA chromosome (indicating the costumers to be visited and how the visitation order will be permuted), a solution for CVRP is generated, as shown in Figure 8.

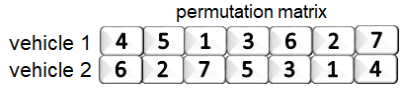


Fig. 7. Permutation matrix (seed).

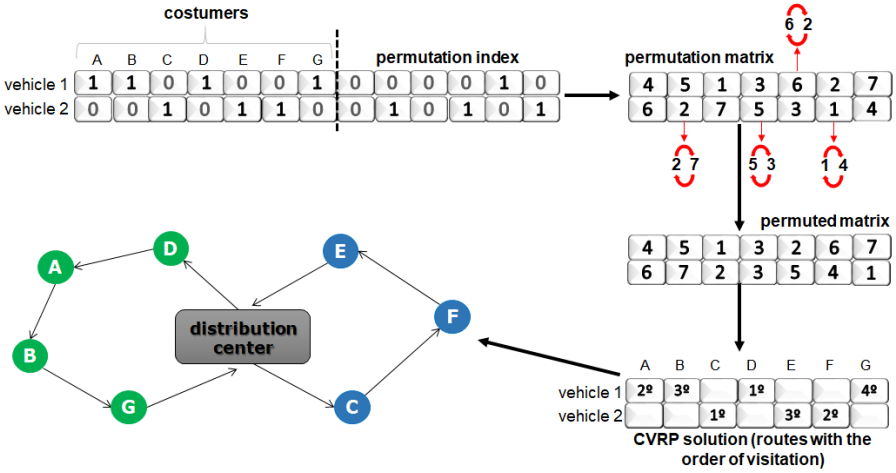


Fig. 8. Combining a permutation matrix with a cromossome to generate a solution for CVRP.

In summary, combining a unique seed with the GA chromosomes a set of different solutions for the CVRP is generated, each one providing the set of customers to be visited by each vehicle as well as the order they will be visited.

It is valid to mention that the encoding scheme described here was inspired by the work of Grassi [22], who proposed a similar way to represent solutions to the Job-shop Scheduling Problem (FJSP), and is very different from those employed in the VRP solution found in the literature, including the schemes described in section 2.3.

5 Experimental Results

To evaluate the proposed encoding scheme we executed the GA ten times for each instance. Then, the results obtained were compared with the best solutions found in the literature. To this end, we considered the optimal solutions presented by Reinelt and Wenger [23] and by Ralphs et al. [24], respectivelly, for instances extracted from Christofides and TSPLIB.

The quality of obtained solutions was evaluated by a measure known as GAP, which is widely used in the literature to express how far the result obtained for a problem is from the best result reported in the literature for that problem. In our case, $GAP = (OF - OF_{Best})/OF_{Best}$, being OF the best value of objective function (Equation 9) obtained in the 10 executions of GA and OF_{Best} the best solution found in the literature. In addition, we present in Table 1 results considering the population initiated in a random way (GA) and including in it a feasible solution generated by Gillet & Miller heuristic (GA+GM).

Table 1. Experimental results of proposed Scheme.

Instance	OF_{Best}	**GA**	**GAP%**	**GA+GM**	**GAP%**	**Time (s)**
Eil7	114	114	0.00%	114	0.00%	30
Eil13	290	316	8.97%	308	6.21%	50
Eil22	375	472	25.87%	376	0.27%	100
Eil23	875	1025	17.14%	903	3.20%	110
Eil30	545	840	54.13%	750	37.61%	150
P-n16-k8	450	520	15.56%	462	2.67%	60
P-n19-k2	212	284	33.96%	255	20.28%	70
P-n20-k2	216	270	25.00%	255	18.06%	80
P-n21-k2	211	249	18.01%	229	8.53%	90
P-n22-k2	216	338	56.48%	268	24.07%	100
P-n22-k8	590	722	22.37%	618	4.75%	100
P-n23-k8	529	675	27.60%	574	8.51%	110
E-n13-k4	247	306	26.89%	302	22.27%	50
E-n22-k4	375	462	23.20%	390	4.00%	100
E-n23-k3	569	892	56.77%	690	21.27%	110
E-n30-k3	534	880	64.76%	687	28.65%	150
Average GAP%	-	-	**29.6%**	-	**13.14%**	-

As shown in Table 1, the proposed scheme provided good performance regarding the quality of the solutions. Considering the results of GA+GM, the GAP in most cases (except for instances "Eil30" and "E-n30-k3") did not exceed 25%, being that for 56% of tested instances the GAP was less than 10%. Still analyzing the GAP, the average value did not exceed 14%, highlighting the good performance of our proposed approach.

With respect to computational cost, as can be seen in Table 1, the processing time ranges from 30 s for the smallest instance ("Eil7") to 150 s for larger

instances ("Eil30" and "E-n30-k3"). In addition, the results showed that the use of Gillett & Miller heuristic to generate a feasible solution in the initial population helps the GA to converge quickly to promising points in the search space, generating solutions with good quality. It should be noted that the average time spent in the execution of the Gillett & Miller algorithm was on average 1.7 s, which shows that it does not compromise the computational cost of GA.

Despite the good results obtained by our approach, many improvements can still be made in the proposed encoding scheme, such as: the use of more than one seed (matrix of permutation), the use of local search operators (k-opt, OR-opt, k-Point Move and Cross-Exchange) and also by incorporating some local search algorithm to refine the solutions generated by GA.

6 Conclusions

In this work we presented a new binary encoding scheme for GA to solve the CVRP. From the computational experiments carried out with instances of Christofides and TSPLIB, it was possible to conclude that the proposed scheme provided good results considering the computational cost and the quality of solutions. In addition, it was found that the chromosome representation is suitable to met the specific characteristics of the CVRP, besides it is simple to interpret and adapt. The experiments also pointed out that the use of Gillett & Miller heuristic helped the convergence of GA to promising points in the search space. In future works some improvements in the proposed scheme will be investigated, such as: i) the use of Clarke and Wright heuristic to generate feasible solutions to be injected into the initial population of the GA; ii) the use of local search operators (k-opt, OR-opt, k-Point Move and Cross-Exchange), aiming to generate different solutions from the same seed; iii) incorporate some local search algorithm to refine the solutions generated by GA and iv) apply the proposed scheme in a large number of instances found in the literature, in order to evaluate the applicability of our approach in real scenarios.

Acknowledgements

The authors would like to thank UNINOVE, FAPESP–São Paulo Research Foundation by financial support (#2017/05188-9) and CNPq–Brazilian National Research Council for the scholarship granted to S. A. Araújo (#311971/2015-6).

References

1. Dalfard, V.M., Kaveh, M., Nosratian, N.E.: Two meta-heuristic algorithms for two-echelon location-routing problem with vehicle fleet capacity and maximum route length constraints. Neural Computing and Applications **23**(7-8) (2013) 2341–2349
2. Nazif, H., Lee, L.S.: Optimised crossover genetic algorithm for capacitated vehicle routing problem. Applied Mathematical Modelling **36**(5) (2012) 2110–2117

3. Vieira, H.P.: Metaheuristic to solve vehicles routing problems with time windows. (2013)
4. Lau, H.C., Chan, T., Tsui, W., Pang, W.: Application of genetic algorithms to solve the multidepot vehicle routing problem. IEEE transactions on automation science and engineering **7**(2) (2010) 383–392
5. Bermudez, C., Graglia, P., Stark, N., Salto, C., Alfonso, H.: Comparison of recombination operators in panmictic and cellular gas to solve a vehicle routing problem. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial **14**(46) (2010)
6. Wang, C.H., Lu, J.Z.: An effective evolutionary algorithm for the practical capacitated vehicle routing problems. Journal of Intelligent Manufacturing **21**(4) (2010) 363–375
7. Tasan, A.S., Gen, M.: A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. Computers & Industrial Engineering **62**(3) (2012) 755–761
8. Ursani, Z., Essam, D., Cornforth, D., Stocker, R.: Localized genetic algorithm for vehicle routing problem with time windows. Applied Soft Computing **11**(8) (2011) 5375–5390
9. Lu, C.C., Vincent, F.Y.: Data envelopment analysis for evaluating the efficiency of genetic algorithms on solving the vehicle routing problem with soft time windows. Computers & Industrial Engineering **63**(2) (2012) 520–529
10. Kuo, R., Zulvia, F.E., Suryadi, K.: Hybrid particle swarm optimization with genetic algorithm for solving capacitated vehicle routing problem with fuzzy demand–a case study on garbage collection system. Applied Mathematics and Computation **219**(5) (2012) 2574–2588
11. Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research **60**(3) (2012) 611–624
12. Reiter, P., Gutjahr, W.J.: Exact hybrid algorithms for solving a bi-objective vehicle routing problem. Central European Journal of Operations Research **20**(1) (2012) 19–43
13. Osaba, E., Diaz, F., Onieva, E.: Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. Applied Intelligence **41**(1) (2014) 145–166
14. Lima, S.J.A., Arajo, S.A., Schimit, P.H.: A hybrid approach based on genetic algorithm and nearest neighbor heuristic for solving the capacitated vehicle routing problem. Acta Scientiarum Technology (2018)
15. Laporte, G.: The vehicle routing problem: An overview of exact and approximate algorithms. European journal of operational research **59**(3) (1992) 345–358
16. Goldberg, D.E.: E. 1989. genetic algorithms in search, optimization, and machine learning. Reading: Addison-Wesley (1989)
17. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic programming: an introduction. Volume 1. Morgan Kaufmann San Francisco (1998)
18. Brooker, R.: Concepts of genetics. McGraw-Hill Higher Education (2012)
19. Kumar, R.: Novel encoding scheme in genetic algorithms. for better fitness. International Journal of Engineering and Advanced Technology **1**(6) (2012) 214–219
20. Kumar, A.: Encoding schemes in genetic algorithm. International Journal of Advanced Research in IT and Engineering **2**(3) (2013) 1–7
21. Wall Mathew, G.: A c++ library of genetic algorithm components (1996)

22. Grassi, F.: Optimization by genetic algorithms of the sequencing of production orders in job shop environments. Master's Dissertation, Industrial Engineering Post Graduation Program – Universidade Nove de Julho (UNINOVE) (2014)
23. Reinelt, G., Wenger, K.M.: Maximally violated mod-p cuts for the capacitated vehicle-routing problem. INFORMS Journal on Computing **18**(4) (2006) 466–479
24. Ralphs, T., Pulleyblank, W., Trotter Jr, L.: On capacitated vehicle routing. Problem, Mathematical Programming (1998)

Optimization of Home Care Visits Schedule by Genetic Algorithm

Filipe Alves[1], Ana I. Pereira[1,2], Adília Fernandes[3], and Paulo Leitão[1]

[1] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal
[2] Algoritmi R&D Centre, University of Minho, Portugal
[3] Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253, Portugal
{filipealves,apereira,adilia,pleitao}@ipb.pt

Abstract. Currently, it has been verified that population is increasingly aged and it is necessary to perform home services. These services include home care visits to patients with impossibility of travel to healthcare centers, where the health professionals perform the medical treatments. Usually, this home care services are performed by nurses that need transportation for this purpose. Therefore, it is necessary to make a schedule of these home care visits that, usually, is made manually by the healthcare center. This work aims to carry out an automatic schedule of home care visits of the healthcare Center of Bragança, Portugal, in order to reduce the travel costs and optimize the time spent on trips. The Genetic Algorithm was used to solve this problem. In this paper it is presented the schedule of home care visits for three days of the healthcare center.

Keywords: optimization, schedule, home care, genetic algorithm

1 Introduction

Home Health Care (HHC) is increasingly important for the current society [1]. In Portugal, for example, there is a high number of older people that need support on theirs homes, so the home health care services are very important on these cases. For many of these people it is impossible to travel to hospitals, healthcare centers, laboratories, among other health services, due to many reasons, for example, their limited mobility, the high distance of health local, or even their homes are in isolated areas without public transportation. Thus, the Home care services are performed by the National Health System since it is economically advantageous to keep people at home instead of providing them a hospital bed [2]. So, these elderly/sick people need to perform the necessary treatments in their homes, so the health professionals need to travel to patients' residences to perform all the requested treatments [3].

To solve this issue, it is necessary to analyze the support needed for home care services to better perform the management of these services. According to studies already carried out, optimization strategies contributes to improve the Home Health Care services in many different ways [4–6]. Some reviews highlight a large

number of papers from the Operational Research community and their main subject is the optimization of the daily planning of health care services. Recently, Nickel et al. [2] propose a heuristic to address the medium-term and short-term planning problem. In the literature, the routing problem is largely tackled as a "Travelling Salesman Problem (TSP)" approach for designing the caregivers route using MILP [7] and/or heuristic [8] approaches for a static, deterministic problem. In this context, the Portuguese public health system includes two types of units: Hospitals and Healthcare Centers. The Healthcare Centers are closer to the population since they follow up their patients, continuously, and the home care services are performed by nurses teams of these units. The aim of this work consists in solving a common problem of Healthcare Centers: produce a daily vehicles schedule of a Healthcare Center where the health professionals (nurses) spent the minimum time to perform all home care visits (considering the travel and treatment patient time).

The paper is organized as follows: Sect. 2 gives a global framework, the description of the real problem and its formulation, and presents the real data collected. The Sect. 3 presents the Genetic Algorithm, the global method chosen to solve the problem. The numerical results are presented in the Sect. 4. Finally, the last scheduling presents the conclusions and future work.

2 Global Framework and Problem Definition

In this context, the global architecture of the HHC system must integrate computational support. Thus, the problem was solved sequentially using the architecture presented in Figure 1.

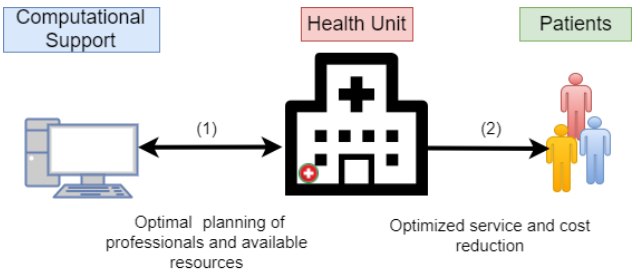


Fig. 1. Developed Architecture

The first step (1), allows to connect the computational support with the informations from Health Unit, in particularly, the list of patients, human and physical resources, among other data, allow to obtain the planning of routes for a certain day of work, that is, the optimal route, instead the manual scheduling.

The second step (2) places the optimized routes into action, allowing the reduction of time spent and reduction of costs for the service.

The Healthcare Centers have a set of vehicles that are used to perform home care visits by the nurses. The Healthcare Center of Bragança (HCB) has, at maximum, five vehicles to perform home care visits for each day. Currently, the vehicles schedule of the HCB is done manually. So, the aim of this work is to produce a vehicles schedule in order to obtain the minimum total spent time to perform all home care visits by the nurses of HCB.

To solve this problem, and considering the information given by HCB, it was considered:

- 15 minutes for the trip, in the same city or locality, to visit different patients.
- The trips duration between the different locations is known.
- The list and the duration of the treatments are known for each patient (defined and given by Health National Unit).
- The number of patients that need home care, and assigned to a working day, is known in advance and does not change during that day.
- Patients care activities cannot be performed at the same time or overlap.
- All trips begin and end up at the Healthcare Center.

2.1 Problem Formulation

Taking into account all the above information for a working day, it was also considered the following parameters for a given day: NP represents the total number of patients that need of home care and NC is the total number of vehicles used for home care visits. Other general information is needed to obtain the final formulation, such us:

- The locations of all patients.
- The time matrix that presents the travel time needed between different locations.
- Vehicles that perform home care visits is known in advance.
- The vehicle characteristics (in the same vehicle, the maximum number of travel persons is seven, hence can transport more than one team).
- Each vehicle carries nurses responsible for certain activities. Therefore, there is no interchangeability among caregivers for care activities.
- In general, each patient will be visited by a specific nurse. In some specific cases, a set of patients can be visited by the same nurse (explained later).

Consider the vector $\mathbf{x} = (p_1, ..., p_{NP}, c_1, ..., c_{NP})$ where the patient p_i will be visited by the vehicle c_i, for $i = 1, ..., NP$, and $\mathbf{x} \in \{1, \cdots, NP\}^{NP} \times \{1, \cdots, NC\}^{NP}$.

For a given $\mathbf{x}$ it is possible to define the vehicles schedule and the function $S^l(\mathbf{x})$, $l = 1, .., NC$, that represents the total time needed to perform all visits of the vehicle l, considering the vector $\mathbf{x}$. The objective function is defined as

$$f(\mathbf{x}) = \max_{l=1,...,NC} S^l(\mathbf{x}) \tag{1}$$

which represents the maximum time spent by all vehicles to perform all the visits. Then the constrained optimization problem will be defined as

$$\min f(\mathbf{x}) \tag{2}$$

where $\mathbf{x} = (p_1, ..., p_{NP}, c_1, ..., c_{NP})$ with $p_i \in \{1, ..., NP\}$ and $c_j \in \{1, ..., NC\}$; all the patients need to be treated $\cup_{i=1}^{P} p_i = \{1, ..., NP\}$ and the number of nurses in each vehicle trip is less or equal to seven.

To solve the minimization problem presented previously, Genetic Algorithm (GA) was used and it is presented in Sect. 3.

2.2 Real Data

The Healthcare Center of Bragança provided three typical working days in April 2016. The data used were available by the Healthcare Center of Bragança (chosen by the institution and simulated a normal working day). In these days, the HCB had:

- On the day 1 - 4 vehicles available to perform the home care visits, 31 patients who require home-based treatments from 12 different locations.
- On the day 2 - 5 vehicles available to perform home care visits, 25 patients who require home-based treatments from 5 different locations.
- On the day 3 - 5 vehicles available to home care visits, 22 patients who require home-based treatments from 9 different locations.

The home care services provided by the nurses, can be classified into five different treatments (or home care visits) presented in Table 1. This information was provided by HCB, where the number of treatment was assigned depending on the type of treatment, described in Table 1.

Table 1. Full characterization of the different treatments provided by the nurses

Treatment	**Description**	**Characterization**	Time (min)
1	Curative	Treatments, for example, pressure ulcer, venous ulcer, surgical wounds, traumatic wounds, ligaments, remove suture material, burns, evaluation and dressing of wound dressings	30
2	Surveillance and Rehabilitation	Evaluation, implementation and patient monitoring	60
3	Curative and Surveillance	Wound treatment, watch over bandage, frequency and tension monitoring, teach and instruct the patient of the complications and pathologies	75
4	Surveillance	Assess risk of falls, self-care, patient behaviors and still the providers knowledge. Monitor, height, tension and heart rate. Patients dietary and medical regimen	60
5	General	Evaluate, support and teach about mourning	60

Analyzing the Table 1, it is verified that the treatments are different and have different times between them. Each of these treatments will be considered for each patient according to the needs (information provided by the Health Center). It becomes necessary to know all the locations of all the patients for the vehicles scheduling.

Table 2 presents all patients locations for the three days (and the corresponding abbreviation) and the spent time between two locations (in minutes). As it was stated before, it was assigned 15 minutes to travel at the same location.

Table 2. Data about travel times between different locations (in minutes)

	A	**Bg**	**B**	**C**	**Cl**	**E**	**G**	**M**	**Ml**	**Mo**	**O**	**P**	**Pi**	**Rl**	**Rb**	**Rd**	**S**	**Sm**	**Sd**	**Sp**
Alfaião (A)	15	24	16	28	35	25	16	22	21	18	20	29	25	21	26	18	24	15	30	32
Bragada (Bg)	24	15	22	33	30	31	27	34	31	16	30	26	15	32	15	19	15	20	16	17
Bragança (B)	16	22	15	25	33	17	16	16	18	16	17	29	23	18	25	15	23	15	29	31
Carrazedo (C)	28	33	25	15	44	24	31	39	39	26	38	39	32	35	33	23	34	24	37	42
Coelhoso (Cl)	35	30	33	44	15	42	38	44	29	19	19	17	17	43	31	29	29	30	36	21
Espinhosela (E)	25	31	17	24	42	15	24	18	34	25	33	37	32	26	34	24	32	25	37	40
Gimonde (G)	16	27	16	31	38	24	15	20	18	21	22	32	29	19	29	21	27	17	33	35
Meixedo (M)	22	34	16	39	44	18	20	15	31	27	29	40	35	17	37	27	34	23	39	42
Milhão (Ml)	21	31	18	39	29	34	18	31	15	23	15	36	31	27	33	27	31	21	36	39
Mós (Mo)	18	16	16	26	19	25	21	27	23	15	24	15	16	26	15	16	19	15	18	21
Outeiro (O)	20	30	17	38	19	33	22	29	15	24	15	27	31	27	32	26	30	20	36	38
Parada (P)	29	26	29	39	17	37	32	40	36	15	27	15	19	38	27	25	25	36	31	23
Pinela (Pi)	25	15	23	32	17	32	29	35	31	16	31	19	15	34	15	20	16	21	21	19
Rabal (Rl)	31	32	18	35	43	26	19	17	27	26	27	38	34	15	34	24	32	22	38	40
Rebordaínhos (Rb)	26	15	25	33	31	34	29	37	33	15	32	27	15	34	15	22	16	20	19	20
Rebordãos (Rd)	18	19	15	23	29	24	21	27	27	16	26	25	20	24	22	15	20	15	25	28
Salsas (S)	24	15	23	34	29	32	27	34	31	19	30	25	16	32	16	20	15	20	15	15
Samil (Sm)	15	20	15	24	30	25	17	23	21	15	20	36	21	22	20	15	20	15	26	28
Sendas (Sd)	30	16	29	37	36	37	33	39	36	18	36	31	21	38	19	25	15	26	15	17
Serapicos (Sp)	32	17	31	42	21	40	35	42	39	21	38	23	19	40	20	28	15	28	17	15

The values presented in Table 2 are based on the data provided by the HCB. As mentioned previously, it is also necessary to know the treatments list of the patients for the three days in study.

The list of treatments for each patient on days 1, 2 and 3, is:

- Day 1: the patients 1, 2, 3, 4, 8, 9, 10, 11, 12, 13, 14, 17, 19, 22, 23 and 24 need treatment 1, the patients 5, 6 and 7 need treatment 2, the patient 15 and 20 requires treatment 3, the patients 16, 21, 25, 26, 27, 28, 29, 30 and 31 need treatment 4 and the patient 18 requires treatment 5. There are some patients that have the same nurse. It is the case of patient 11 and 13; patient 4 and 22; patient 17 and 23; and patient 1 and 12.
- Day 2: the patients 3, 16, 17, 21, 22, 23, 24 and 25 need treatment 1, the patients 2, 8, 9 and 11 need treatment 2, the patients 4, 5, 10, 12, 13, 14, 15 and 18 need treatment 3, the patients 1, 19 and 20 requires treatment 4 and the patients 6 and 7 need treatment 5. In this day there are two pairs of patients that have the same nurse, that is the case of patient 3 and 16; and patient 21 and 25.

– Day 3: the patients 1, 2, 3, 6, 7, 8, 9, 17, 21 and 22 need treatment 1, the patients 4 and 5 requires treatment 2, the patients 13 and 14 need treatment 3, the patients 11, 12, 15, 16, 18, 19 and 20 need treatment 4 and the patient 10 requires treatment 5. The patients 16 and 18 must be visit by the same nurse.

Based on all the presented data, the main objective is to obtain the vehicles schedule, in order to minimize the total spent time needed to perform the trips, the treatments and return to the starting point, HCB.

3 Genetic Algorithm

Initially proposed by John Holland [9], GA inspired by the natural biological evolution, uses a population of individuals to apply genetic procedures: crossover between two different individuals or/and mutation in one individual.

The values of the control parameters used in GA were adjusted to a suitable experience of the problem, i.e. it was considered a population size (Ps) and concerning the probability of the procedures (crossover and mutation), 50% rate was selected. Is expected that the following population (next generation) of individuals has a better capability. The algorithm repeats the crossover and mutation procedures in new populations until the desired diversity of solutions is performed [10, 11].

The method applied in this work is summarized by the following Algorithm.

Algorithm 1 : Genetic Algorithm

1: Generates a randomly population of individuals, $\mathcal{P}^0$, with dimension N_{pop}. Set $k = 0$.
2: **while** stopping criterion is not met **do**
3: Set $k = k + 1$.
4: $\mathcal{P}' =$ Apply crossover procedure in population $\mathcal{P}^k$.
5: $\mathcal{P}'' =$ Apply mutation procedure in population $\mathcal{P}^k$.
6: $\mathcal{P}^{k+1} = N_{pop}$ best individuals of $\{\mathcal{P}^k \cup \mathcal{P}' \cup \mathcal{P}''\}$.
7: **end while**

Details related to the algorithm implementation can be seen in [12]. The iterative procedure terminates after a maximum number of iterations (NI) or after a maximum number of function evaluations (NFE).

4 Numerical Results

The HCB also provides us the vehicles schedule, performed manually, that is, without any mathematical model or subject to computational mechanisms. Thus, for the days 1, 2 and 3, respectively, will be presented in the Tables 3, 4 and 5.

Table 3. HCB Schedule for day 1

Vehicles Scheduling in the Health Unit						
Vehicles						
1	HCB - B	P(1) - T.1	B - P	P(2) - T.1	P - B	P(3) - T.1
	P(4) - T.1	P(5) - T.2	B - Rb	P(6) - T.2	Lunch	Rb - B
	P(7) - T.2	B - M	P(22) - T.1	M - B	P(12) - T.1	P(24) - T.1
	B - Bg	P(25) - T.4	Bg - HCB			
2	HCB - C	P(8) - T.1	C - E	P(9) - T.1	E - B	P(10) - T.1
	B - Rd	P(11) - T.1	Rd - B	P(13) - T.1	B - S	P(14) - T.1
	S - HCB	Lunch				
3	HCB - B	P(15) - T.3	B - Sp	P(16) - T.4	Sp - P	P(17) - T.1
	P - B	P(18) - T.5	Lunch	B - O	P(19) - T.1	O - B
	P(20) - T.3	P(21) - T.4	P(23) - T.1	B - HCB		
4	HCB - B	P(26) - T.4	P(27) - T.4	P(28) - T.4	P(29) - T.4	Lunch
	B - MI	P(30) - T.4	P(31) - T.4	MI - HCB		

Table 4. HCB Schedule for day 2

Vehicles Scheduling in the Health Unit						
Vehicles						
1	HCB - B	P(1) - T.4	P(8) - T.2	P(19) - T.4	B - HCB	Lunch
2	HCB - B	P(2) - T.3	B - Rd	P(4) - T.3	Rd - G	P(5) - T.3
	Lunch	G - B	P(6) - T.5	P(7) - T.5	B - HCB	
3	HCB - CI	P(9) - T.2	P(10) - T.3	P(11) - T.2	P(12) - T.3	Lunch
	P(13) - T.3	P(14) - T.3	P(15) - T.3	CI - HCB		
4	HCB - B	P(16) - T.1	P(17) - T.1	P(18) - T.3	P(20) - T.4	B - G
	P(23) - T.1	Lunch	G - B	P(3) - T.1	B - HCB	
5	HCB - RI	P(21) - T.1	RI - B	P(22) - T.1	P(24) - T.1	B - Rd
	P(25) - T.1	Rd - HCB	Lunch			

Table 5. HCB Schedule for day 3

Vehicles Scheduling in the Health Unit						
Vehicles						
1	HCB - B	P(1) - T.1	P(2) - T.1	P(3) - T.1	P(19) - T.4	P(20) - T.4
	B - HCB	Lunch				
2	HCB - E	P(4) - T.2	E - B	P(5) - T.2	P(6) - T.1	P(7) - T.1
	B - Rd	P(8) - T.1	Rd - HCB	Lunch		
3	HCB - P	P(9) - T.1	P - Rd	P(10) - T.5	Rd - A	P(11) - T.4
	P(12) - T.4	A - HCB	Lunch			
4	HCB - B	P(13) - T.3	B - Sm	P(14) - T.3	Sm - B	P(15) - T.4
	P(16) - T.4	Lunch	B - Sd	P(18) - T.4	Sd - HCB	
5	HCB - Sd	P(17) - T.1	Sd - Mo	P(21) - T.1	Mo - MI	P(22) - T.1
	MI - HCB	Lunch				

It was presented the real vehicles schedule used in the reference working days to compare the improvement. Each patient is visited by one nurse. In some specific situations there are a set of patients that will be visit for the same nurse (as was described previously). It is also possible to conclude that the maximum spent time by the vehicles was 694, 651 and 448 minutes for each day (without the lunchtime), respectively, as show Table 3, 4 and 5.

Regarding the identification of patients and treatments, P(1) - T.1 represents Patient 1 who needs Treatment 1. For example, the schedule of the vehicle 1 for day 3 will be: begin the trip in HCB to Bragança to execute the home care visit of Patients 1, 2, 3, all with treatment 1, and then, still in the same locality, visits patient 19 and 20 (both need treatment 4). Finished, return to HCB for lunch.

For the computational results it was used the Matlab Software, version 2015a, running in a computer with a processor Intel (R) Core (TM) i5 2.40GHz CPU with 4.0 GB of memory RAM.

In this work, the Genetic Algorithm (GA) was used to produce the vehicles schedule with the minimum spent time (not considering the lunchtime). For the population size, it was considered $Ps = 30$ individuals, the maximum number of function evaluation was fixed at $NFE = 5000$ and the maximum number of iterations as $NI = 100$.

Since GA is a stochastic method, it was performed 100 runs to solve the problem. Table 6 presents the GA overall performance, such as: the best solution obtained in all runs (f^*_{min}), the solution average (f^*_{avg}), and finally, the average time to solve the optimization problem (Time_{avg}) in seconds.

Table 6. Summary of GA results

	f^*_{min}	f^*_{avg}	$\textbf{Time}_{avg}$
Day 1	545	573	24
Day 2	498	510	20
Day 3	333	333	15

Analyzing the numerical results presented in the previous table, it is possible to verify that the total total time found by GA for the different days, is less than the times planned manually. The first solution with the shortest total time was chosen as the final result. The average of the solutions is slightly higher in the first two days and the same on the third day, that is, optimized planning has almost always been found. In each run was always found solutions. Finally, the average time to solve the problem was always less than 24 seconds, i.e. very fast.

Consequently, the GA obtains the vehicles schedule for each working day. The Table 7 presents the vehicle schedule obtained by the algorithm for day 1. It is possible to conclude that the maximum spent time is 545 minutes.

In our problem only the effective time spent with the home care visits is considered by the Healthcare Center, excluding lunchtime. The same happens for the remaining computational results.

Table 7. Optimal vehicles schedules using GA for day 1

Vehicles Schedule using GA						
Vehicles						
1	HCB - B	P(10) - T.1	P(12) - T.1	P(1) - T.1	P(15) - T.3	B - Rb
	P(6) - T.2	Lunch	Rb - P	P(2) - T.1	P - B	P(29) - T.4
	P(26) - T.4	B - HCB				
2	HCB - B	P(21) - T.4	P(5) - T.2	B - P	P(17) - T.1	P - B
	P(23) - T.1	P(3) - T.1	Lunch	B - MI	P(31) - T.4	MI - E
	P(9) - T.1	E - C	P(8) - T.1	C -HCB		
3	HCB - B	P(28) - T.4	P(20) - T.3	B - M	P(22) - T.1	M - B
	P(4) - T.1	P(27) - T.4	Lunch	B - Bg	P(25) - T.4	Bg - B
	P(18) - T.5	B - HCB				
4	HCB - MI	P(30) - T.4	MI - B	P(7) - T.2	B - Sp	P(16) - T.4
	Sp - B	P(24) - T.1	Lunch	B - Rd	P(11) - T.1	Rd - B
	P(13) - T.1	B - S	P(14) - T.1	S - O	P(19) - T.1	O - HCB

For the day 2, it was obtained the vehicles schedule presented in Table 8, that has 498 minutes to perform all the home care visits.

Table 8. Optimal vehicles schedules using GA for day 2

Vehicles Schedule using GA						
Vehicles						
1	HCB - CI	P(14) - T.3	P(11) - T.2	P(15) - T.3	CI - B	P(19) - T.4
	Lunch	P(17) - T.1	B - HCB			
2	HCB - B	P(1) - T.4	P(7) - T.5	P(20) - T.4	B - CI	P(12) - T.3
	CI - HCB	Lunch				
3	HCB - G	P(5) - T.3	G - B	P(24) - T.1	B - CI	P(10) - T.3
	CI - B	P(6) - T.5	Lunch	B - CI	P(13) - T.3	CI - HCB
4	HCB - B	P(8) - T.2	P(3) - T.1	P(16) - T.1	B - Rd	P(4) - T.3
	Rd - B	P(18) - T.3	B - HCB	Lunch		
5	HCB - RI	P(21) - T.1	RI - Rd	P(25) - T.1	Rd - B	P(2) - T.3
	B - CI	P(9) - T.2	CI - G	P(23) - T.1	Lunch	G - B
	P(22) - T.1	B - HCB				

Finally, for the day 3 was collected a vehicles schedule with a maximum time spent by vehicles of 333 minutes, as it is possible to see in Table 9.

In order to conclude and for a better perception of the illustrated results, it will be presented the following example of home visits made by vehicle 1 on day 3. Thus, the vehicle 1 starts the route at the Healthcare Center (HCB) to Parada, provides care to Patient 9 (Treatment 1), then travels of Parada to Bragança, to provide care to Patient 19 (Treatment 4). After travel to Alfaião, takes care of Patient 11 (Treatment 4) and then travels again to Sendas to provide care to Patient 17 (Treatment 1). Finally, returns of Sendas to the point of origin (HCB) to end the home visits and have the lunchtime.

Table 9. Optimal vehicles schedules using GA for day 3

Vehicles	Vehicles Schedule using GA					
1	HCB - P	P(9) - T.1	P - B	P(19) - T.4	B - A	P(11) - T.4
	A - Sd	P(17) - T.1	Sd - HCB	Lunch		
2	HCB - B	P(20) - T.4	B - Rd	P(10) - T.5	Rd - A	P(12) - T.4
	A - B	P(6) - T.1	P(7) - T.1	B - HCB	Lunch	
3	HCB - B	P(2) - T.1	B - Rd	P(8) - T.1	Rd - B	P(1) - T.1
	B - MI	P(22) - T.1	MI - B	P(13) - T.3	B - HCB	Lunch
4	HCB - Mo	P(21) - T.1	Mo - E	P(4) - T.2	E - Sm	P(14) - T.3
	Sm - B	P(5) - T.2	B - HCB	Lunch		
5	HCB - B	P(15) - T.4	B - Sd	P(18) - T.4	Sd - B	P(16) - T.4
	P(3) - T.1	B - HCB	Lunch			

Table 10 presents a comparison of the the maximum time spent by each vehicle, using GA, and the time provided by the HCB.

Table 10. Maximum time spent (minutes) by each vehicle on home care visits

	Vehicles		
	Day 1	Day 2	Day 3
HCB	694	651	448
GA	545	498	333

Comparing the results with the vehicles schedule provided by the HCB it is possible to conclude that the GA obtained vehicles schedule with a reduction (approximately 30%) of the maximum spent time to perform all the home visits.

The numerical results show more than an optimal solution, needing a few seconds for find them. GA, had 100% of successful rate since they found a feasible solution in all runs. While the manual process takes a long time, since they are complex cases and without optimization tests, it is possible to verify that the average time to solve the problem is quickly and allows several solutions.

5 Conclusions and Future Work

The home care visits are usually planned manually and without any computer support in HCB, this implies that the solution obtained may not be the best one, in addition to the process being complex and taking a higher time consuming. So, in an attempt to optimize the process, it is necessary to use strategies to minimize the maximum time spent by each vehicle on home care routes, without, however, worsening the quality of the provided services and, always, looking for the best schedules organization. In this paper, the scheduling problem of HCB was solved successfully using GA method, needing few seconds to find the problem solution.

This approach represents a gain for all entities involved, as health professionals and patients.

For future work, it is possible to reformulate the problem and take into account the multi-objective approach to minimize the total time spent by all vehicles. Another approach may be the use of multi-agent for real-time simulation.

Acknowledgements

The authors thank HCB, Bragança, Portugal. This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e Tecnologia, Portugal, within the Project Scope: UID/CEC/00319/2013.

References

1. Benzarti, E., Sahin, E., Dallery, Y.: Operations management applied to home care services: Analysis of the districting problem. Decision Support Systems **55**(2) (2013) 587–598
2. Nickel, S., Schröder, M., Steeg, J.: Mid-term and short-term planning support for home health care services. European Journal of Operational Research **219**(3) (2012) 574–587
3. Rest, K.D., Hirsch, P.: Supporting urban home health care in daily business and times of disasters. IFAC-PapersOnLine **48**(3) (2015) 686–691
4. Liu, R., Xie, X., Augusto, V., Rodriguez, C.: Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. European Journal of Operational Research **230**(3) (2013) 475–486
5. Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J.: The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. European Journal of Operational Research **219**(3) (2012) 598–610
6. Sahin, E., Matta, A.: A contribution to operations management-related issues and models for home care structures. International Journal of Logistics Research and Applications **18**(4) (2015) 355–385
7. Kergosien, Y., Lenté, C., Billaut, J.C.: Home health care problem: An extended multiple traveling salesman problem. In: Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009). (2009) 85–92
8. Redjem, R., Marcon, E.: Operations management in the home care services: a heuristic for the caregivers routing problem. Flexible Services and Manufacturing Journal **28**(1-2) (2016) 280–303
9. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1992)
10. Ghaheri, A., Shoar, S., Naderan, M., Hoseini, S.S.: The applications of genetic algorithms in medicine. Oman medical journal **30**(6) (2015) 406
11. Kumar, M., Husian, M., Upreti, N., Gupta, D.: Genetic algorithm: Review and application. International Journal of Information Technology and Knowledge Management **2**(2) (2010) 451–454
12. Bento, D., Pinho, D., Pereira, A.I., Lima, R.: Genetic algorithm and particle swarm optimization combined with powell method. In: AIP Conference Proceedings. Volume 1558., AIP (2013) 578–581

Path Planning Optimization Method Based on Genetic Algorithm for Mapping Toxic Environment

Luis Piardi[1,2], José Lima[2,4,5], Ana I. Pereira[2,4,6], and Paulo Costa[3,5]

[1] Federal University of Technology - Paraná, Toledo, Brazil
luis.fp.piardi@alunos.ipb.pt,
[2] Instituto Politécnico de Bragança, Bragança - Portugal
jllima@ipb.pt, apereira@ipb.pt,
[3] Faculty of Engineering of University of Porto, Porto -Portugal
paco@fe.up.pt,
[4] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Bragança - Portugal
[5] INESC-TEC, Centre for Robotics in Industry and Intelligent Systems, Porto - Portugal
[6] Algoritmi R&D Centre, University of Minho, Braga - Portugal

Abstract. The ionizing radiation is used in the nuclear medicine field during the execution of diagnosis exams. The administration of nuclear radio pharmaceutical components to the patient contaminates the environment. The main contribution of this work is to propose a path planning method for scanning the nuclear contaminated environment with a mobile robot optimizing the traveled distance. The Genetic Algorithm methodology is proposed and compared with other approaches and the final solution is validated in simulated and real environment in order to achieve a closer approximation to reality.

Keywords: Genetic Algorithm, Mobile Robot, Path Planning, Optimization

1 Introduction

Medical imaging is an area of knowledge with continuous technological innovation, that develops new techniques for the medical diagnosis in order to provide an image of the anatomy of the human body and its functions [1]. According to NUMDAB (Nuclear Medicine Database), there are 1490 nuclear medicine institutions in the world, of which 1288 are active. Actually, 0.69 million PET (Positron Emission Tomography) and PET-CT (Computed Tomography) annual examinations are registered in the world [2].

The nuclear medicine provides diagnosis tests that detect with some precision when a certain part of the body has a change in the metabolism. The administration of nuclear radio pharmaceutical components to the patient must be carefully done by specialists. Unfortunately, the patient can contaminate the environment with physiologic needs. Moreover, environment and the patient should

be isolated by a period of time regarding the decay of nuclear properties. The inspection of the clearance of the environment is mainly made by human beings that are exposured to the ionizing radiation that may cause the damage in the organs and tissues. The scanning and measurement of the radiation can be done resorting to a mobile robot that performs the acquisition based on a Geiger counter. The path planning of the robot should guarantee that the complete scan is performed and ensure the environment is clean and technicians can enter the room. The presented paper addresses a path planning method that scans the desired environment while optimizing the mobile robot travelled distance. This optimization, based on Genetic Algorithm, is implemented in simulation and real robot scenario and compared with other approaches that validates the proposed methodology.

The paper is organized as follows: After a brief introduction, Section 2 presents the related work. Then, Section 3 addresses problem formulation of path planning to scan the environment. Section 4 presents the developed Genetic algorithm and its operations. Section 5 presents the obtained numerical results and compares it with a heuristic method for path planning. Finally, last section concludes the paper and presents some future work.

2 Related Work

Path planning is crucial for autonomous mobile robots in various environments with the presence of obstacles [3]. In the literature, path planning is defined as: "Given a map and a goal location, path planning involves identifying a trajectory that will cause the robot to reach the goal location when executed. Path planning is a strategic problem-solving competence, as the robot must decide what to do over the long term to achieve its goals" [4].

This subject is widely discussed by the academic community. The task of moving the robot from a starting point to a target point avoiding obstacles and running an optimized or near optimal path is a complex computational process. Complexity increases as the environment has more known, unknown or dynamic obstacles.

Several algorithms are used for the mobile robot path planning problem, e.g, visibility chart [5], Voronoi diagrams [6, 7], cell decomposition [8], potential field [9], A* [10] and other methods found in the literature. According to [3] "Each method differs in its effectiveness depending on the type of application environment and each one of them has its own strength and weaknesses".

Another approach used in the search to optimize path planning is based on Genetic Algorithms [3, 11–14]. In [3] this methodology was used with search algorithm to carry out the path planning of starting point and end point avoiding obstacles and collisions in the environment (static or dynamic where was used an optimization in the mutation operator to optimize the path or seek a path near the optimum). Already in [12], applying crossover and mutations to search for an optimized path, using a connectivity grid to represent the plant where the robot is inserted, the objective is to find the lowest path between the start

and end points, avoiding repeating cells along the way, simplifying the fitness function by analyzing path length.

Many approaches in path planning, even using Genetic Algorithm, seek only to make the shortest, or most efficient, path between two distinct points (start and target). However, our problem has a different framework, similar to the classic travelling salesman problem (TSP). Considering a range of n cities where the purpose of this problem is to start the route in city defined, visiting the other cities only once, and them returning to the first city [15]. Considering the possibility of the existence of several cities, the TSP becomes complex with $(n-1)!$ possible routes to be calculated. The differences between our problem and the travelling salesman's problem is that the starting and ending points are different and the robot must avoid collision with obstacles.

In the present work, we will adapt a Genetic Algorithm to find the smallest path to perform a scan in the environment represented by a connectivity grid. An example with this applicability is found in [16,17] works. We will initially restrict the problem to static environments, and future work will address dynamic environments where there are unknown obstacles by the robot.

3 Problem Formulation

The challenge of path planning for robots is usually formulated as follows: given a mobile robot and a description of an environment, we need to find a route between two specified locations, the start and the end point. During the execution of the path the robot can not collide with obstacles and the optimization criterion must be satisfied (i.e., shortest path) [13].

To simplify the path planning problem, it is necessary to make some assumptions. They are as follows:

- A path will be selected, always starting from a start point to a target point, as show in Fig. 1.
- Known obstacles are mapped and represent a cell in the connectivity grid.
- The proposed algorithm acts on a connectivity grid arranged in a two dimensional space (2D) or $\mathbb{R}^2$ space.
- The robot does not perform movements in diagonal directions. It only moves between interconnected points in horizontal and vertical directions in the grid of connectivity, as show in Fig. 1.
- The robot should visit all cells, or points, that are free of obstacles in the grid of connectivity at least once.

3.1 Problem Space Representation

Many works developed in the area of path planning for mobile robots, use a graph grid of connectivity to represent the environment and obstacles. In the present work we use a similar approach as presented in the papers [3,14], where we modify the order of the values as shown in Fig. 1. The dark color represents

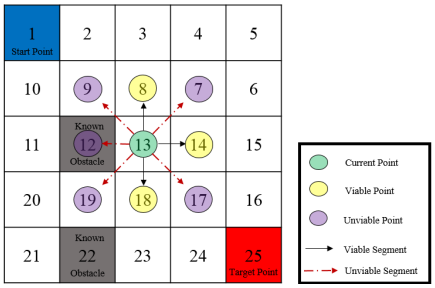


Fig. 1: Connectivity grid (5x5) and an example of possible points and segments. Without an obstacle the 12 point will be a reachable point.

obstacles, while the lighter colors represent obstacle-free cells. It is important to remind that the size of the connectivity grid can vary, according to the resolution of the desired scan.

4 Genetic Algorithm

In this section we will present the Genetic Algorithm (GA) used to solve the path planning problem described in the section above. To facilitate understanding, when referring to a gene, we are indicating a cell in the grid of connectivity. When a chromosome is pronounced, it indicates a set of cells that connects the start point to the end point.

4.1 Encoding Representation

The encoding method is one of the key steps in the GA design. The representation of the possible paths to be realized by the robot, is known as chromosome [3, 11, 14]. The path is encoded in a sequence of adjacent cells. This sequence is started with the start cell (upper left corner) and ended with the destination (bottom right corner) cell. The path consists of a variable number of segments formed between two cells or waypoints. Each segment is a straight line which can be vertical or horizontal. Diagonal segments are invalid. Figure 2 shows a possible chromosome generated from the connectivity grid of Figure 1.

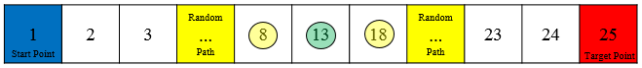


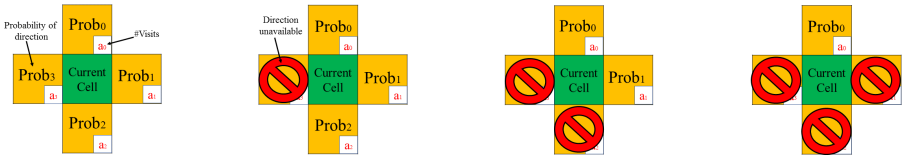
Fig. 2: Possible chromosome of the GA initial population.

4.2 Initial Population

The initial population is generated in order to respect the criteria of horizontal and vertical movement allowed. The initial population is composed by a set of chromosomes that are subjected to a random process, where each chromosome starts at the start point and ends at the target point (see Fig. 1 and Fig. 2), and each chromosome describes a path that should visit all the points of the grid of connectivity at least once.

With the intention to reducing the search time of the evolutionary algorithm, all the chromosomes generated by the initial population represent an executable path, as in the papers [3, 14, 18].

To generate a chromosome of the initial population, the algorithm applies a mask with an unitary cross as shape, where the center represents the current point and the extremity of the cross represents the directions allowed for the path. Each direction has a probability of choice according to the amount of visits already undertaken. In other words, a point that was less visited is more likely to be visited compared to its neighbors that were most visited. In this way, all points have a probability of being chosen, guaranteeing a great diversity for the initial population. The mask can be seen in Fig. 3a, where the possible configurations are also shown depending on the availability of neighboring points or in case the mask is centered at the end of the grid.



(a) All directions available. (b) Three directions available. (c) Two directions available. (d) One Directions Available.

Fig. 3: Geometry of the mask for the generation of the initial population.

The probability of each direction $Prob_i$, for $i = 0, ..., 3$, is a function of the visit cell number (a_i) and n, that represents the number of available directions. $Prob_i$, for $i = 0, ..., 3$, can be defined as:

$$Prob_i = \begin{cases} \dfrac{\sum_{k=0}^{n-1} a_k - a_i}{(n-1) \cdot \sum_{k=0}^{n-1} a_k}, & for \quad n > 1 \\ 1, & for \quad n = 1 \end{cases} \tag{1}$$

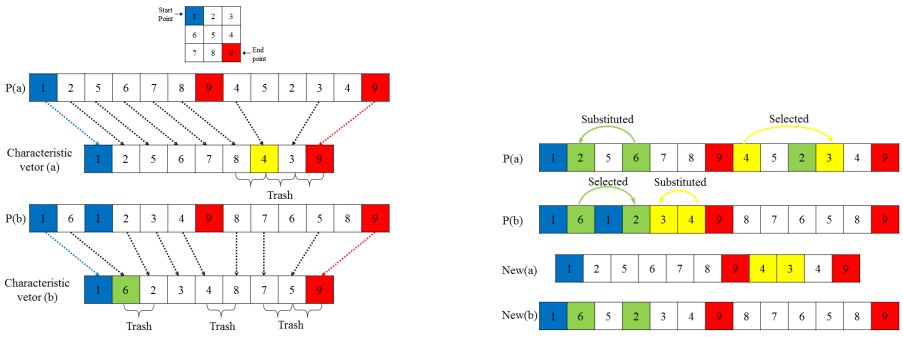
The procedure ends when all cells are visited and the path terminates at the end point of the connectivity grid.

4.3 Crossover Operation

Crossover can be defined as a process of taking two parent solutions to produce a child. After reproduction process, the population is enriched with better individuals. The goal of the operator is to find new structures that have a high probability of causing significant improvements [19].

In the developed algorithm, the crossover consider two parents, selected randomly, to produce two children. The first step of this process is to generate the characteristic path of these two selected parents. The characteristic path is a sequence of cells that are visited for the first time in the path. In this way, individual and distinct information is stored for each parent. Each characteristic path must begins at the starting point and ends at the endpoint (even if the endpoint was previously visited).

Between two consecutive cells of the characteristic path there may be several cells in the parent chromosome, which were already visited. Fig. 4a illustrates a situation where two characteristic paths are generated for two parents, where a 3×3 grid was used to facilitate the understanding of this operation.



(a) Characteristic path. (b) Generation of new chromosomes.

Fig. 4: Crossover operation details.

Only the garbage ranges can be used to generate the new chromosomes (offsprings). Therefore a random point of the characteristic path is selected (in yellow in (a) and green in (b)), where between it and its subsequent point there is garbage to be replaced by the smaller interval in the opposite parents that generated the characteristic path. Fig.4b shows in detail the process of generating the new chromosomes.

4.4 Mutation Operation

Mutation operation prevents the algorithm from being trapped in a local minimum. Mutation is an operator to maintain genetic diversity in the population. It introduces new genetic structures in the population by randomly modifying

some of its building blocks. It helps to escape from a local solution and maintains diversity in the population to find structures that improve the path planning [19]. All chromosomes are candidates to be submitted in the mutation process with a probability of $Prob_m$ [18].

In order to prevent a mutation parent producing a infeasible path, the developed algorithm for the mutation operator was established to avoid all such cases, i.e, all the paths generated after the mutation are feasible. As in the crossover operator, the first procedure performed is to generate a characteristic path. Then, a cell of the characteristic path is randomly chosen and its subsequent one where a random path between them will be inserted. The generated random path allows only horizontal and vertical movements. Fig. 5 illustrates the operation for a 3×3 size connectivity grid.

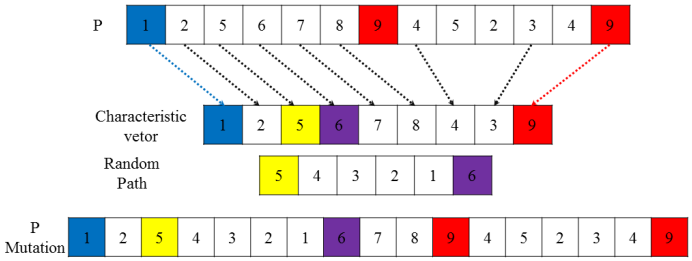


Fig. 5: Detail of mutation operator.

4.5 Selection Process

In our work the goal is to obtain an optimal path, i. e., a path with the shortest distance between the starting point and the end point by visiting all points of the connectivity grid. In order to evaluate the chromosomes, the amount of cells are analysed. The best path is the one with less cells in its chromosome. When one of the chromosomes has $(i \times j)$ cells, i.e. when all points are visited only once to accomplish all the visits, the path is fully optimized.

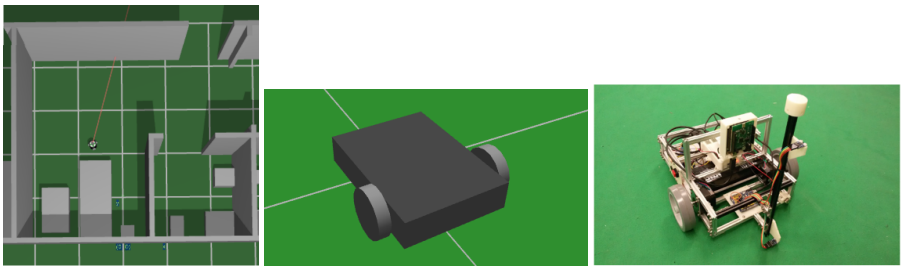
To get the best path it is necessary that the smallest paths are maintained and transferred to the next generations. A selection process is proposed to obtain the best parents and yet guarantee the diversity of the new populations. This process consists of ordering all the chromosomes obtained in the current iteration of the algorithm, considering the results of crossover and mutation operations, and classifying them in ascending order. After ordering the chromosomes the new population is selected, where 10% of the individuals are the ones with the smallest paths and the remaining 90% are selected randomly from the ordered chromosomes.

5 Numerical Results

In this section we will present the results obtained with the Genetic Algorithm for the path planning to the problem described above. In order to evaluate the obtained results, we will compare with the heuristic method proposed in [16]. This heuristic planning method is based on eight different priorities of directions for the robot, where the best priority is selected and executed by the robot.

To validate and test the results of the path planning algorithm, we used the SimTwo Simulator [20]. In the simulation environment (Fig. 6a), the robot follows the dimensions of real robot used in [17], as show Fig. 6b and Fig. 6c.

The test environment used has a dimension of 3 meters long by 3 meters wide. As explained above, the size of the connectivity grid can be changed and consequently the resolution of the scan as well. For the present work, an 8×8 grid was used.

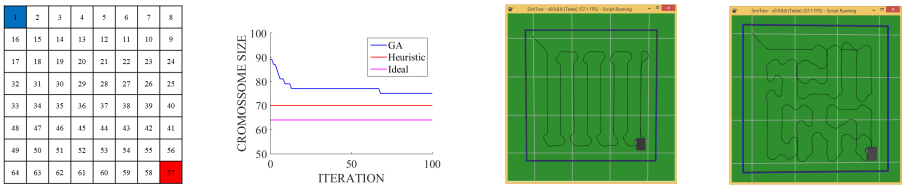


(a) SimTwo: 3D simulation environment. (b) Simulated robot. (c) Real Robot [17].

Fig. 6: Tools used to test the developed algorithm.

In this work we test three different situations. Situation A where there are no obstacles and the optimal path is trivial. Situations B and C where there are some known obstacles and it is not obvious the optimal path. The Genetic Algorithm was executed ten times for each case, and the best result for each problem is presented.

Fig. 7 presents the Situation A where connectivity grid has no obstacles. The performance of the Genetic Algorithm (GA) is presented in Fig. 7b where is analysed the evolution of the chromosomes generations during a GA run. The line referring to the heuristic method presents the number of waypoints visited when it is applied in path planning and the ideal line represents the number of obstacle-free waypoints present in the connectivity grid, i.e., the ideal size of the chromosome. Then it is possible to observe that the heuristic planning obtains better results when compared with the GA results in environments with simple layout (without obstacles). The obtained solution of the two tested procedures are illustrated in 7c and 7d.



(a) Connectivity grid 8×8.

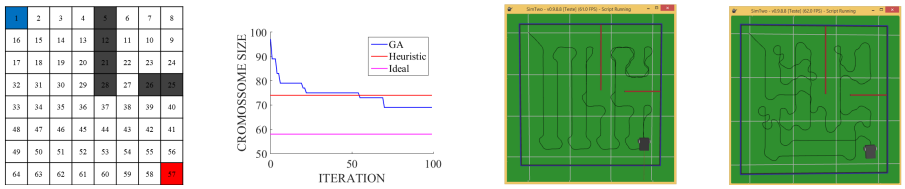
(b) Comparison of algorithms.

(c) Path planning by heuristic.

(d) Path planning by GA.

Fig. 7: Plan with a connectivity grid 8×8 without obstacles.

Fig. 8 presents the Situation B where connectivity grid has two known obstacles. The performance of the Genetic Algorithm (GA) is presented in Fig. 8b where is possible to observe that, after some iterations, the Genetic Algorithm is capable to identify a better path when compared with the heuristic planning procedure. The obtained solutions of the heuristic planning and Genetic Algorithm are illustrated in Fig. 8c and Fig. 8d, respectively.



(a) Connectivity grid 8×8.

(b) Comparison of algorithms.

(c) Path planning by heuristic.

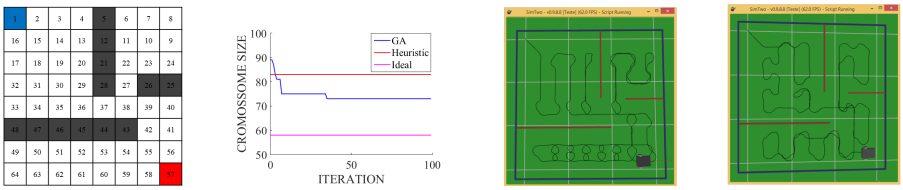
(d) Path planning by GA.

Fig. 8: Plan with a connectivity grid 8×8 with two known obstacles.

Fig. 9 presents the Situation C where the connectivity grid has three known obstacles. The performance of the Genetic Algorithm (GA) is presented in Fig. 9b where it is possible to observe that GA obtains better solutions than the heuristic planning procedure starting from the initial iterations.

The obtained solutions of the heuristic planning and Genetic Algorithm are illustrated in Fig. 9c and Fig. 9d, respectively.

The Table 1 presents the number of visited cells. So, analyzing the Genetic Algorithm behaviour in the Situations A, B and C it is possible to conclude that the GA method has better results in a more complex environment.



(a) Connectivity grid 8×8.

(b) Comparison of algorithms.

(c) Path planning by heuristic.

(d) Path planning by GA.

Fig. 9: Plan with a connectivity grid 8×8 with three known obstacles.

Table 1: Number of visited cells: Heuristic Procedure and Genetic Algorithm method comparison

Situation	HP	GA
A	70	75
B	75	71
C	83	72

6 Conclusion and Future Work

The presented paper proposes a path planning using an adapted Genetic Algorithm to perform a scan in environments with toxic substances. The path is applied to a mobile robot that moves according to the computed trajectory. It is desired to optimize the travelled distance by the robot while mapping all the desired waypoints. In the environments with known obstacles, the efficiency of the proposed Genetic Algorithm is relevant, identifying the optimal path in complex situations. Thus, a more efficient trajectory is planned.

As future work we intend to apply the algorithm in dynamic environments with also unknown obstacles and replace the SimTwo by the real robot for a more realistic approach. Moreover, a real time constraint should be addressed to solve the path during the scan.

Acknowledgment

This work is financed by Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the FCT Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within projects

POCI-01-0145-FEDER-006961, POCI-01-0145-FEDER-007043 and
UID/CEC/00319/2013.

References

1. Suetens, P.: Fundamentals of Medical Imaging. 2 edn. Cambridge University Press (2009)
2. International Atomic Energy Agency (IAEA). http://nucmedicine.iaea.org/default.asp Online, accessed on 13/12/2017.
3. Tuncer, A., Yildirim, M.: Dynamic path planning of mobile robots with improved genetic algorithm. Computers & Electrical Engineering **38**(6) (2012) 1564–1572
4. Roland Siegwart, I.R.N.: Introduction to Autonomous Mobile Robots. 1st edn. The MIT Press (2004)
5. Ma, Y., Zheng, G., Perruquetti, W.: Cooperative path planning for mobile robots based on visibility graph. In: Proceedings of the 32nd Chinese Control Conference. (July 2013) 4915–4920
6. Dong, H., Li, W., Zhu, J., Duan, S.: The path planning for mobile robot based on voronoi diagram. In: 2010 Third International Conference on Intelligent Networks and Intelligent Systems. (Nov 2010) 446–449
7. Yang, X., Zeng, Z., Xiao, J., Zheng, Z.: Trajectory planning for robocup msl mobile robots based on bézier curve and voronoi diagram. In: 2015 IEEE International Conference on Information and Automation. (Aug 2015) 2552–2557
8. Kloetzer, M., Mahulea, C., Gonzalez, R.: Optimizing cell decomposition path planning for mobile robots using different metrics. In: 2015 19th International Conference on System Theory, Control and Computing (ICSTCC). (Oct 2015) 565–570
9. Yu, Z.z., Yan, J.h., Zhao, J., Chen, Z.F., Zhu, Y.h.: Mobile robot path planning based on improved artificial potential field method. Harbin Gongye Daxue Xuebao(Journal of Harbin Institute of Technology) **43**(1) (2011) 50–55
10. Moreira, A.P., Costa, P.J., Costa, P.: Real-time path planning using a modified a* algorithm. In: Proceedings of ROBOTICA 2009-9th Conference on Mobile Robots and Competitions. (2009)
11. Hu, Y., Yang, S.X.: A knowledge based genetic algorithm for path planning of a mobile robot. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Volume 5., IEEE (2004) 4350–4355
12. Ismail, A., Sheta, A., Al-Weshah, M.: A mobile robot path planning using genetic algorithm in static environment. Journal of Computer Science **4**(4) (2008) 341–344
13. Sedighi, K.H., Ashenayi, K., Manikas, T.W., Wainwright, R.L., Tai, H.M.: Autonomous local path planning for a mobile robot using a genetic algorithm. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). Volume 2. (June 2004) 1338–1345 Vol.2
14. Alnasser, S., Bennaceur, H.: An efficient genetic algorithm for the global robot path planning problem. In: 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP). (July 2016) 97–102
15. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM) **7**(4) (1960) 326–329
16. Piardi, L., Lima, J., Costa, P., Brito, T.: Development of a dynamic path for a toxic substances mapping mobile robot in industry environment. In: Iberian Robotics conference, Springer (2017) 655–667

17. Lima, J., Costa, P.: Ultra-wideband time of flight based localization system and odometry fusion for a scanning 3 dof magnetic field autonomous robot. In: Iberian Robotics conference, Springer (2017) 879–890
18. Moharam, R., Morsy, E.: Genetic algorithms to balanced tree structures in graphs. Swarm and Evolutionary Computation **32** (2017) 132–139
19. Sivanandam, S.N., Deepa, S.N.: Introduction to Genetic Algorithms. Springer
20. Costa, P., Gonçalves, J., Lima, J., Malheiros, P.: Simtwo realistic simulator: A tool for the development and validation of robot software. Theory and Applications of Mathematics & Computer Science **1**(1) (2011) 17

A Surrogate-Assisted Approach for Expensive Equality Constrained Optimization

Rommel G. Regis
Saint Joseph's University, Philadelphia PA 19131, USA,
rregis@sju.edu

Abstract. This paper presents a surrogate-assisted approach for expensive equality constrained black-box optimization. The proposed method proceeds in two phases where the first phase finds an approximately feasible point and the second phase improves the objective function value of this point by moving close to the region determined by the active surrogate equality constraints. The proposed method is implemented using radial basis function (RBF) surrogates. The numerical results are promising on test problems with a single equality constraint in comparison to a pattern search method.

Keywords: constrained optimization, equality constraint, surrogate model, radial basis function, expensive function

1 Introduction

This paper solves equality constrained optimization problems of the form:

$$\min \{f(x) \ : \ H(x) = (h_1(x), \ldots, h_p(x)) = 0, \ \ell \leq x \leq u\} \tag{1}$$

where $f, h_1, \ldots, h_p$ are functions whose values at an input $x \in \mathbb{R}^d$ are obtained from a deterministic and expensive computer simulation. The region $[\ell, u] \subset \mathbb{R}^d$ defined by the bounds is the *search space* for problem (1). Here, one *simulation* for a given $x \in [\ell, u]$ yields the values of $f(x)$ and $H(x)$. This paper assumes that accurate gradient information for the objective and constraint functions are not available. Problem (1) is denoted by ECBOP$(f, H, [\ell, u])$, where ECBOP stands for *Equality Constrained Black-box Optimization Problem*.

While several surrogate-based methods have been developed for problems with expensive black-box inequality constraints (e.g., Regis [1]), very few have been proposed for problems with expensive equality constraints. Among them is SACOBRA (Bagheri et al. [2]), which handles equality constraints by starting with an expanded feasible region that is allowed to gradually shrink to a volume close to zero. Another possibility is to use probability of feasibility approaches (e.g., Forrester et al. [3]) where the infill criterion uses a product of the expected improvement of the objective function and the probability of feasibility. This paper proposes the *Equality Constrained Optimization using Surrogates (ECOS)* algorithm that handles expensive black-box equality constraints using a different approach. In ECOS, the next sample point is chosen from a set of randomly generated trial points that lie close to the surrogate equality constraints.

2 Handling Expensive Equality Constraints

Algorithm 1 is a pseudo-code for ECOS, which consists of two phases. Phase I finds an approximately feasible point according to some constraint tolerance ϵ, and then Phase II improves the objective function value of this approximately feasible point by moving close to the region determined by the surrogate equality constraints. In each iteration of Phase II, multiple trial points are generated using a Gaussian distribution centered at the current best point and whose covariance matrix is chosen so that the trial points tend to lie close to the intersection of the surrogate equality constraints. Next, the sample point is chosen to be best trial point according to two criteria: predicted objective function value based on the surrogate and minimum distance from previous sample points. In Algorithm 1, t is the number of simulations, $x^{(t)}$ is the sample point, and $y^{(t)}$ is the current best point with respect to f and a constraint violation function V_H. Here, $V_H(x) = \max_{1 \le i \le p}[\max(|h_i(x)| - \epsilon, 0)]$, but other choices are possible.

To describe the generation of trial points, consider a non-degenerate multivariate normal $N(\mu, \Sigma)$ distribution. Here, μ is the mean vector and Σ is the symmetric and positive definite covariance matrix. Its PDF is given by

$$f(x) = (2\pi)^{-d/2}(\det(\Sigma))^{-1/2} \exp\left[-(1/2)(x-\mu)^T \Sigma^{-1}(x-\mu)\right],$$

which is constant on the surface of ellipsoids of the form $(x-\mu)^T\Sigma^{-1}(x-\mu) \le c^2$ for some fixed $c \in \mathbb{R}$. If $c^2 = \chi^2_d(\alpha)$ (upper (100α)th percentile of the chi-squared distribution with d degrees of freedom), then

$$P[(x-\mu)^T\Sigma^{-1}(x-\mu) \le \chi^2_d(\alpha)] = 1 - \alpha.$$

Let $\lambda_1, \ldots, \lambda_d$ be the eigenvalues of Σ and let $u_1, \ldots, u_d$ be corresponding unit eigenvectors. The ith axis of the ellipsoid is defined by $\sqrt{\lambda_i \chi^2_d(\alpha)} u_i$.

To generate the trial points in Steps 4 and 8, first determine the the gradients of the active surrogate equality constraints at the current best point $y^{(t)}$. Next, find an orthonormal basis for the tangent plane to the surrogate equality constraints at $y^{(t)}$. Then, extend the orthonormal basis for the tangent plane to an orthonormal basis for $\mathbb{R}^d$. The vectors in this orthonormal basis will be the eigenvectors for the covariance matrix of the Gaussian distribution. Next, determine the eigenvalues for the covariance matrix that will produce a suitable $(1-\alpha)100\%$ prediction ellipsoid for the Gaussian distribution. Here, we want the lengths of the axes in the directions of the basis vectors for the tangent plane to be much larger than the lengths for the other directions. Using the eigenvectors and eigenvalues obtained earlier, determine the covariance matrix for the multivariate Normal distribution. Finally, generate multivariate Normal random vectors with mean $y^{(t)}$ and with the covariance matrix found.

ECOS is implemented using a cubic radial basis function (RBF) model described in Powell [4] and applied to several test problems with a single equality constraint. Here, both the objective and equality constraints are treated as black-boxes and assumed to be expensive. Preliminary results for ECOS are promising in comparison with some alternative methods such as pattern search.

Algorithm 1 Equality Constrained Optimization using Surrogates (ECOS).

Inputs: (1) ECBOP($f, H, [\ell, u]$); (2) constraint violation function $V_H(x)$; (3) number of trial points: r; (4) space-filling design: $\{x^{(1)}, \ldots, x^{(k)}\} \subseteq [\ell, u]$ with $k \geq d + 1$; (5) type of surrogate; (6) constraint tolerance: ε; (7) maximum simulations: T_{max}

Output: The best point found by the algorithm.

1. Evaluate Design. For $t = 1, \ldots, k$, run simulator to obtain $f(x^{(t)})$ and $H(x^{(t)})$.

2. Initialize Current Best Point. Set the number of simulations $t := k$ and let $y^{(t)}$ be the current best point with respect to f and V_H. If all sample points are infeasible, then proceed to Phase I. Else, proceed to Phase II.

Phase I (Find a Feasible Point):

3. Fit Surrogates. Fit surrogates for the objective and equality constraint functions using all previous sample points (all are currently infeasible).

4. Generate Trial Points. Generate random trial points from a Gaussian distribution with mean $y^{(t)}$ and whose covariance matrix is described below.

5. Select Sample Point. Use surrogate constraints to identify trial points with the minimum number of predicted constraint violations. Among these points, select $x^{(t+1)}$ as the one with the smallest predicted $V_H(x)$ based on the surrogates.

6. Evaluate Sample Point. Run simulator to obtain $f(x^{(t+1)})$ and $H(x^{(t+1)})$. Update $y^{(t+1)}$. Reset $t \leftarrow t + 1$. If $t < T_{max}$ and $y^{(t)}$ is not yet feasible, then go back to Step 3. Else, if $t = T_{max}$, then stop.

Phase II (Improve Best Feasible Point Found):

7. Fit Surrogates. Fit surrogates for the objective and constraint functions using all previous sample points (have at least one feasible point).

8. Generate Trial Points. Generate random trial points from a Gaussian distribution with mean $y^{(t)}$ and whose covariance matrix is chosen so that the trial points will tend to lie close to the intersection of the surrogate equality constraints.

9. Select Sample Point. Use surrogates for the objective and constraints to determine which trial points have the minimum number of predicted constraint violations. Among these points, select $x^{(t+1)}$ to be the best point according to a weighted combination of two criteria: surrogate objective function value and minimum distance from previous sample points.

10. Evaluate Sample Point. Run simulator to obtain $f(x^{(t+1)})$ and $H(x^{(t+1)})$. Update $y^{(t+1)}$. Reset $t \leftarrow t + 1$. If $t < T_{max}$, go back to Step 7; otherwise, stop.

References

1. Regis, R.G.: Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Computers and Operations Research **38**(5) (2011) 837–853
2. Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Applied Soft Computing **61** (2017) 377–393
3. Forrester, A.I.J., Sobester, A., Keane, A.J.: Engineering Design via Surrogate Modelling: A practical guide. John Wiley & Sons (2008)
4. Powell, M.J.D.: The theory of radial basis function approximation in 1990. In Light, W., ed.: Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions. Oxford University Press, Oxford, U.K. (1992) 105–210

An Adaptive Metaheuristic for Unconstrained Multimodal Numerical Optimization

Helder Pereira Borges[1], Omar Andres Carmona Cortes[1], and Dario Vieira[2]

[1] Instituto Federal do Maranhão (IFMA)
São Luis, MA, Brasil
{helder, omar}@ifma.edu.br
[2] Engineering School of Information and Digital Technologies (EFREI)
Villejuif, France
dario@efrei.fr

Abstract. The purpose of this paper is to show an adaptive metaheuristic based on GA, DE, and PSO. The choice of which one will be used is made based on a probability that is uniform at the beginning of the execution, and it is updated as the algorithm evolves. That algorithm producing better results tend to present higher probabilities of being selected. The metaheuristic has been tested in four multimodal benchmark functions for 1000, 2000, and 3000 iterations, managing to reach better results than the canonical GA, DE, and PSO. A comparison between our adaptive metaheuristic and an adaptive GA has shown that our approach presents better outcomes, which was proved by a t-test, as well.

Keywords: Metaheuristics, Genetic Algorithms, Differential Evolution, Particle Swarm Optimization, Adaptive, Multimodal.

1 Introduction

Different metaheuristics present unique exploration and exploitation capabilities, i.e., they possess different forms of exploring and exploiting the search space. Thus, what works for solving a specific problem might be not good for tackling another one. Moreover, each problem can demand a particular set of parameters for each algorithm.

In this context, adaptive algorithms have appeared trying to solve as many problems as possible with no changes. Mostly approaches deal with adaptation in terms of operators or parameters. When dealing with operators, the adaptation tries to identify which operator is more suitable to the problem, while in parameters, the algorithm attempts to discover the best value. Both situations happen during the execution of the algorithm, i.e., on-the-fly. For example, in [1], a Genetic Algorithm choses between four crossover and three mutations operators as the metaheuristic solves multimodal benchmarks functions. Then, the authors evolve the algorithm to a self-adaptive one in [2], in which the parameters are encoded into the genes of each solution.

In fact, there are many works dealing with adaptive algorithms such as [3], [4], [5], [6], [7], [8], etc. However, works that deal with different metaheuristics at the same time are rare. For instance, [9] came up with an algorithm that executes a GA and a PSO simultaneously; then they share information between their populations. The main drawback of this algorithm is the performance because both metaheuristics must execute at the same time. Costa's work [10] came up with the idea of upgrading the population in the SPEA2 (Strength Pareto Evolutionary Algorithm) using GA, DE and PSO by applying a stochastic approach, in which as the algorithm executes if a metaheuristic creates a population which dominates the previous one, then the probability of being chosen increases.

In this context, this paper is organized as follows: Section 2 illustrates the pseudo code and how the canonical algorithms GA, DE, and PSO work; Section 3 introduces our adaptive approach and how the algorithm chooses which metaheuristic to use in execution time; Section 4 shows how the experiments were set and explains the results; finally, Section 5 presents the conclusion and future work.

2 Metaheuristics

2.1 Genetic Algorithms

In 1962, Holland [11] proposed an adaptive system that will become the Genetic Algorithm as we know it. The pseudo code of a Genetic Algorithm is shown in Algorithm 1. Firstly, the GA creates a random set of candidate solutions. For each one, the algorithm calculates its fitness that expresses the quality of a solution. Then, individuals are chosen to form a temporary population using a selection mechanism. The temporary population undergoes genetic operators (crossover and mutation) to generate the new population. Finally, the new population is evaluated. If the algorithm is elitist and the previous population contains the best chromosome, this solution replaces the worst individual in the new one; otherwise, the old population is entirely replaced by the new one. The whole process is repeated while the stop criterion is not reached.

Algorithm 1 - Genetic Algorithm

```
Population ← generateInitialPopulation();
fitness ← Eval(Population)
while stop Criteria not reached do
    TempPopulation ← Selection(Population);
    TempPopulation ← Crossover(TempPopulation);
    Population ← Mutation(TempPopulation);
    fitness ← Eval(Population)
end while
```

2.2 Particle Swarm Optimization

The particle swarm optimization was proposed by Kennedy and Eberhart [12] in 1995. The algorithm consists of particles that are placed into a search space, and move themselves combining its own history position and the current global optimal solution. A particle position is represented in the search space as $S_i^D = (s_i^1, s_i^2, ..., s_i^D)$ and it is updated based on its velocity $V_i^D = (v_i^1, v_i^2, ..., v_i^D)$, in which D represents the problem dimension. The new position is computed by Equations 1 and 2, where w represents the inertia weight, c_l and c_2 are acceleration constants, r_l and r_2 are random number in the range $[0, 1]$, p_i^d is the best position reached by the particle p, and g^d is a vector storing the global optima of the swarm so far.

$$v_i^d = w \times v_i^d + c_1 r_1 \times (p_i^d - x_i^d) + c_2 r_2 \times (g^d - x_i^d) \tag{1}$$

$$s_i^d = s_i^d + v_i^d \tag{2}$$

The Algorithm 2 outlines how PSO works. Initially, the swarm is created at random, in which each particle has to be within the domain $[a_i^d, b_i^d]$. Then, particles are evaluated to initialize the P matrix and the g^d vector, which are the best experience of each particle and the best solution that has been found so far, respectively. Thereafter, the velocity and the position of a particle are updated within a loop that obeys some stop criterion. In the pseudo code presented in the Algorithm2, the stop criterion is a certain number of iterations.

Algorithm 2 - PSO Pseudo Code

```
S ← InitSwarm();
fitness ← Eval(S);
g ← best(fitness);
P ← S;
while stop Criterion not reached do
    V = w * V + c1r1 × (P − X) + c2r2 × (g − X);
    S = S + V;
    fitness ← Eval(S);
    if best(fitness) is best than g then
        g ← best(fitness);
    end if
    if fitness(s) is best than p then
        p ← fitness(s);
    end if
end while
```

2.3 Differential Evolution

Differential Evolution (DE) is a metaheuristic developed by Storn e Price [13] in 1995. It works similarly to a Genetic Algorithm; however, using different operators. The Algorithm 3 presents its pseudo code. The DE algorithm starts initializing a random population along with its evaluation. Then, the mutation process selects three random individuals creating the vector v, which is also called vector of differences, where F is a constant chosen by the programmer. Afterward, a new individual is created using a gene from v if a random number is less than CR (*Crossover Rate*); otherwise, the gene comes from pop_{ij}. Finally, if the new individual is better than that one in the current population, the new one replaces it.

Algorithm 3 - DE Pseudo Code

```
pop ← InitPopulation();
fitness ← Eval(pop);
while stop Criterion not reached do
    Select 3 individuals randomly: indiv_1, indiv_2, indiv_3;
    v_j ← indiv_3 + F × (indiv_1 − indiv_2);
    if (rand() ¡ CR) then
        new_indiv_j ← v_j
    else
        new_indiv_j ← pop_ij
    end if
    if fitness(new_indiv) best than fitness(pop_i) then
        pop_i ← new_indiv;
    end if
end while
```

3 The Adaptive Metaheuristic

The adaptive metaheuristic was inspired in Carvalho's work [1], in which the authors use a similar process for choosing the proper genetic operators. The Algorithm 4 presents the pseudo code of our approach. Basically, the adaptive metaheuristic selects which one to use at execution time. All three algorithms start with a uniform distribution, *i.e.*, all of them have the same probability of being selected. Then, if the chosen algorithm improves the current solution then its probability of being selected increases by 1% while the other probabilities decrease by 0.5%; otherwise, the probability decreases, while the other ones increase with the same rate.

Also, the adaptation process can be done as many time as the programmer wants. In this work, we tested the adaptation done on each, 25, 50, 100 and 125 iterations. On each iteration means that the adaptation, *i.e.*, the algorithm is

Algorithm 4 - Adaptive Metaheuristic

```
Population ← Init_Population();
fitness ← eval(Population)
while Number of Generations not reached do
    Select Metaheuristic()
    Use Metaheuristic(Population, fitness)
    if (Metaheuristic improves solution) then
        Prob_Metaheuristic++;
    else
        Prob_Metaheuristic- -;
    end if
    Population ← NewPopulation;
end while
```

chosen on each iteration; on each 25, the adaptation is done in all iterations multiple of 25, and so on. For example, if the algorithm runs using 1000 iterations, will be performed 40 adaptation on each 25, 20 adaptation on each 50, and so on.

4 Computational Experiments

4.1 Experiment Setup

All experiments were conducted on an Intel Xeon X5650 2.67GHz, 24GB RAM, 500 GB Hard Disk on Ubuntu 16.04.2 LTS. In terms of parameters, all algorithms used a population of 50 individuals, 30 genes, and 50 trials. The number of trials were chosen based on the central limit theorem that allow us to use parametric tests. The algorithms were implemented in Java 7 using Eclipse Oxygen. Also, the following configurations were used on each kind of algorithms:

- GA: Probability of Crossover = 0.7; Probability of Mutation = 0.02; Selection method = Tournament; Tournament Size = 7; Crossover = Simple; Elitism = TRUE.
- PSO: $c_1 = 2.33$; $c_2 = 2.47$; linear inertia weight ($W_{max} = 0.9$, $W_{min} = 0.4$); Topology = Star (Fully Connected).
- DE: Crossover Rate = 0.6; $F = 0.815$; DE/Rand/1.
- Adaptive: Probability increasing = 1%; Probability decreasing = 0.5%; Iterations for adapting = 1, 25, 50, 100, and 125.

4.2 Benchmark Functions

In this work, we used four multimodal (several local optima) benchmark functions as presented in Table 1, which are minimization functions very common for testing metaheuristic. In this context, minimize a function $f(x), x \in \mathbf{R}^n$ is to discover a vector x dimension n in which the value of $f(x)$ is minimum. In the

Table 1: Benchmark functions

Code	Name	Function	Domain	Min.
ROS	Rosenbrock	$f_1(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_{i-1})^2]$	[-5,10]	0
RAS	Rastringin	$f_2(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)]$	[-5.12,5.12]	0
SCW	Schwefel	$f_3(x) = -\sum_{i=1}^{n} x_i * \sin\sqrt{\lvert x_i\rvert}$	[-500,500]	-12569.49
GRI	Griewank	$f_4(x) = \sum_{i=1}^{n}(\frac{x(i)^2}{4000}) - \prod_{i=1}^{n}(\frac{x(i)}{\sqrt{i}}) + 1;$	[-600,600]	0

referred table we can see what the domain of each gene and its optimum value are.

The Rosenbrock function is commonly considered as unimodal. Nonetheless, the Generalized Rosenbrock ($f_1(x)$) is multimodal in dimension higher than three [14].

4.3 Results for 1000, 2000 and 3000 iterations

Table 2 presents the result (best, mean, worst and standard deviation) of each canonical algorithm optimizing each benchmark function. Whereas Table 3 shows the results of the adaptive metaheuristic on each type of adaptation. Both tables are considering 1000 iterations. As we can see, the best results were presented by the adaptive algorithms performing the adaptation on every 100 and 125 iterations. On the other hand, the canonical PSO shows better outcomes in the mean and worst results for ROS function; and, GA reached best mean and worst for SCW. However, the Adaptive gives better outcomes for RAS, SCW and GRI functions. Regarding the worsts, the GA shows the best worst result for RAS, SCW, and GRI. Nevertheless, the important thing here is that the best solutions are reached by our adaptive metaheuristic.

Tables 4 and 5 show the results after 2000 iterations for the canonical algorithms and the adaptive metaheuristic, respectively. Again the best results were obtained by the adaptive metaheuristic. However, the canonical GA tended to reach the best results in terms of *mean* and *worst* in the Schwefel function, while PSO reached similar results in the Rosenbrock function.

Tables 6 and 7 present the outcomes after 3000 iterations for the canonical algorithms and the adaptive metaheuristic, respectively. In this experiment, we can notice that almost all results are better in the adaptive metaheuristic, excepting for Griewank function in which the DE presented the best result in *best* and *mean* columns.

4.4 Comparison Against an Adaptive GA

In this comparison, two characteristics have been changed. In the first one, the Schwefel function changed to $f_3(x) = 418d - \sum_{i=1}^{n} x_i * \sin\sqrt{|x_i|}$, in which d is the dimension of the function that in our case is 30. In the second one, the population size has been increased to 100. Table 8 shows the results for 1000 and

Table 2: Results for 1000 iterations - canonical

	GA			
	Best	Mean	Worst	Std. Dev.
ROS	29.732	105.830	175.230	45.658
RAS	0.256	0.843	**2.265**	0.421
SCW	-12568.247	**-12563.990**	**-12556.364**	3.308
GRI	0.612	0.984	**1.042**	0.077
	DE			
ROS	358.122	706.918	1505.896	234.446
RAS	137.588	160.048	179.952	10.394
SCW	-9330.961	-8612.050	-8091.159	288.092
GRI	1.029	1.061	1.095	0.018
	PSO			
ROS	28.551	**32.881**	**50.051**	4.924
RAS	0.088	19.328	61.822	16.761
SCW	-11306.211	-9155.361	-6479.682	1010.047
GRI	0.200	0.886	1.241	0.240

2000 iteration using the Adaptive GA [1], which stochastically chooses between four crossover and three mutation operators in execution time.

Table 9 presents the results of our approach considering the adaptation on "each 100" iterations. As we can observe, the adaptive metaheuristic presents better results compared to the adaptive GA.

A bicaudal-based t-test considering $\alpha = 0.05$ and a hypothesis test (H_0) that there are no differences between means, is presented in Table 10. Thus, if t is within [-2.009, 2.009], we accept H_0, otherwise we reject it. As we can see, we rejected H_0 in almost all cases. Therefore, the differences are meaningful in the majority of the benchmark functions. In other words, the Adaptive metaheuristic presents the best results compared to the adaptive GA. Even though the difference between the adaptive GA and the adaptive metaheuristic is not meaningful in Rosenbrock function after 2000 iterations, the mean of the metaheuristic algorithm is smaller as well as the standard deviation, demonstrating that the adaptive metaheuristic is much more stable than the adaptive GA

5 Conclusions

In this paper, we presented a stochastic adaptive metaheuristic based on GA, DE, and PSO. Experiments using 1000, 2000, and 3000 iterations have shown that our approach tends to present the best results with some variations in the means and in the worsts; however, those differences tends to disappear in favor of our approach as we increase the number of iterations. A comparison against an adaptive GA showed that the adaptive metaheuristic reached much better outcomes.

Table 3: Results for 1000 iterations - adaptive

	1 Iteration			
	Best	Media	Worst	Std. Dev.
ROS	32.547	133.657	204.346	42.589
RAS	0.188	**1.799**	3.948	1.067
SCW	-12568.710	-12560.413	-12534.404	7.355
GRI	0.267	0.893	1.049	0.194
	25 Iteration			
ROS	27.989	64.011	155.244	42.440
RAS	0.073	2.465	9.119	2.042
SCW	-12568.104	-12552.624	-12481.255	16.161
GRI	0.022	0.800	1.101	0.338
	50 Iteration			
ROS	27.534	49.323	157.457	38.451
RAS	0.258	3.951	15.306	3.430
SCW	-12569.344	-12560.870	-12517.038	11.205
GRI	0.071	0.764	1.192	0.318
	100 Iteration			
ROS	**26.362**	60.773	626.600	101.926
RAS	0.028	6.585	119.175	19.864
SCW	-12569.381	-12318.744	-8529.903	884.435
GRI	**0.006**	0.758	1.100	0.337
	125 Iteration			
ROS	26.945	55.341	218.926	45.152
RAS	**0.018**	8.177	176.832	29.462
SCW	**-12569.472**	-12294.613	-8428.805	881.318
GRI	0.041	**0.701**	1.120	0.403

Future work includes: use a self-adaptive approach on all parameters of our method; parallelization of the adaptive metaheuristic using a General Purpose Graphical Unit Processing (GP-GPU); and to use fuzzy logic to select which algorithm to execute in a particular iteration.

References

1. Carvalho, E., Cortes, O.A.C., Costa, J.P., Rau-Chaplin, A.: A stochastic adaptive genetic algorithm for solving unconstrained multimodal numerical problems. In: IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS). (May 2016) 130–137
2. Carvalho, E., Cortes, O.A.C., Costa, J.P., Vieira, D.: A parallel adaptive genetic algorithm for unconstrained multimodal numerical optimization. In: Simpósio Brasileiro de Automação Inteligente (SBAI). (Oct 2017)
3. Qin, A.K., Tang, K., Pan, H., Xia, S.: Self-adaptive differential evolution with local search chains for real-parameter single-objective optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC). (July 2014) 467–474

Table 4: Results for 2000 iterations - canonical

	GA			
	Best	Mean	Worst	Std. Dev.
ROS	6.470	79.330	151.804	45.048
RAS	0.030	**0.181**	**0.659**	0.125
SCW	-12569.150	-12568.279	**-12567.027**	0.548
GRI	0.276	0.690	1.009	0.181
	DE			
ROS	31.279	89.137	207.356	48.936
RAS	95.899	129.277	154.292	11.721
SCW	-11743.363	-10256.110	-9176.490	746.605
GRI	0.002	**0.083**	0.200	0.062
	PSO			
ROS	28.077	28.972	**30.422**	0.462
RAS	0.262	9.119	37.369	9.076
SCW	-11251.846	-9286.336	-7917.191	890.262
GRI	0.008	0.527	0.991	0.302

4. Agrawal, S., Silakari, S., Agrawal, J.: Adaptive particle swarm optimizer with varying acceleration coefficients for finding the most stable conformer of small molecules. **34**(11-12) (2015) 725–735
5. Fan, Q., Yan, X.: Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. IEEE Transactions on Cybernetics **46**(1) (Jan 2016) 219–232
6. Tambouratzis, G.: Modifying the velocity in adaptive pso to improve optimisation performance. In: 2017 Ninth International Conference on Advanced Computational Intelligence (ICACI). (Feb 2017) 149–156
7. Toriyama, N., Ono, K., Orito, Y.: Adaptive ga-based ar-hidden markov model for time series forecasting. In: 2017 IEEE Congress on Evolutionary Computation (CEC). (June 2017) 665–672
8. Zhang, X., Zhang, X., Wang, L.: Antenna design by an adaptive variable differential artificial bee colony algorithm. IEEE Transactions on Magnetics **PP**(99) (2017) 1–4
9. Kusetogullari, H., Yavariabdi, A.: Self-adaptive hybrid pso-ga method for change detection under varying contrast conditions in satellite images. In: 2016 SAI Computing Conference (SAI). (July 2016) 361–368
10. Costa, J.P.A., Cortes, O.A.C., Jnior, E.C.: An adaptive algorithm for updating populations on (spea2). In: Simpósio Brasileiro de Automação Inteligente (SBAI). (July 2017)
11. Holland, J.H.: Outline for a logical theory of adaptive systems. Journal of ACM **9**(3) (July 1962) 297–314
12. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks. Volume 4., Perth, Australia, IEEE Service Center, Piscataway, NJ (1995) 1942–1948
13. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces (1995)
14. al Rifaie, M.M., Aber, A. In: Dispersive Flies Optimisation and Medical Imaging. Springer International Publishing, Cham (2016) 183–203

Table 5: Results for 2000 iterations - adaptive

	1 Iteration			
	Best	Mean	Worst	Std. Dev.
ROS	10.991	80.763	149.699	51.184
RAS	0.040	0.282	0.938	0.212
SCW	-12569.233	-12568.156	-12565.010	0.909
GRI	0.054	0.451	1.000	0.260
	25 Iteration			
ROS	**0.378**	39.603	134.218	34.498
RAS	0.004	0.261	2.103	0.425
SCW	-12569.213	-12567.382	-12560.399	1.812
GRI	0.013	0.137	**0.485**	0.131
	50 Iteration			
ROS	5.221	28.030	78.754	9.638
RAS	**0.001**	0.225	2.049	0.391
SCW	-12569.476	**-12568.625**	-12560.359	1.684
GRI	0.001	0.085	0.532	0.125
	100 Iteration			
ROS	1.636	27.517	74.411	9.349
RAS	0.002	0.190	1.277	0.285
SCW	-12569.485	-12567.591	-12560.978	2.381
GRI	**0.000**	0.115	0.966	0.207
	125 Iteration			
ROS	1.962	**26.640**	33.528	4.699
RAS	**0.001**	0.504	7.167	1.240
SCW	**-12569.486**	-12568.512	-12560.069	1.826
GRI	**0.000**	0.141	0.934	0.262

Table 6: Results for 3000 iterations - canonical

	GA			
	Best	Mean	Worst	Std. Dev.
ROS	3.723	69.786	137.587	47.010
RAS	0.018	0.073	0.166	0.038
SCW	-12569.361	-12568.958	-12568.017	0.304
GRI	0.114	0.488	0.942	0.227
	DE			
ROS	25.537	27.366	51.774	4.368
RAS	88.416	116.684	134.385	9.051
SCW	-12444.607	-11851.003	-9850.102	511.812
GRI	**0.000**	**0.000**	0.002	0.001
	PSO			
ROS	26.566	28.543	29.442	0.462
RAS	0.004	5.257	28.130	7.292
SCW	-11138.709	-8869.721	-6479.683	1049.671
GRI	0.002	0.217	0.630	0.169

Table 7: Results for 3000 iterations - adaptive

	1 Iteration			
	Best	Mean	Worst	Std. Dev.
ROS	10.149	76.380	150.567	51.695
RAS	0.011	0.112	0.387	0.098
SCW	-12569.457	-12568.978	-12567.635	0.430
GRI	0.014	0.193	0.776	0.151
	25 Iteration			
ROS	0.413	35.954	134.695	33.232
RAS	0.001	**0.035**	**0.136**	0.032
SCW	-12569.388	-12568.692	-12566.106	0.743
GRI	**0.000**	0.033	0.159	0.039
	50 Iteration			
ROS	**0.027**	25.111	27.082	4.409
RAS	**0.000**	0.073	1.993	0.335
SCW	-12569.486	-12569.228	-12566.700	0.538
GRI	**0.000**	0.020	0.132	0.033
	100 Iteration			
ROS	0.082	**23.085**	**26.617**	7.149
RAS	**0.000**	0.059	0.949	0.181
SCW	**-12569.487**	**-12569.296**	-12567.423	0.460
GRI	**0.000**	0.023	0.253	0.053
	125 Iteration			
ROS	0.378	23.154	28.790	8.095
RAS	**0.000**	0.061	0.894	0.168
SCW	**-12569.487**	-12569.289	-12567.281	0.441
GRI	**0.000**	0.015	0.199	0.037

Table 8: Results for the Adaptive GA

Adaptive GA - 1000 Iterations				
	Best	Mean	Worst	Std. Dev.
ROS	41.837	248.705	1072.296	169.331
RAS	12.757	43.533	2176.657	19.680
SCW	1.807	3999.956	1371.780	1046.521
GRI	0.586	0.956	47.799	0.136
Adaptive GA - 2000 Iterations				
	Best	Mean	Worst	Std. Dev.
ROS	18.396	130.863	6543.140	902.405
RAS	4.7851	31.1575	73.7288	18.4707
SCW	0.329	1349.220	4118.256	1381.812
GRI	0.155	0.573	1.389	0.241

Table 9: Results for the Adaptive Metaheuristic Using 1000 and 2000 iterations

1000 Iteration				
Adaptation 100 Iterations				
	Best	Mean	Worst	Std. Dev.
ROS	18.506	33.126	134.860	18.076
RAS	0.006	9.561	159.507	26.413
SCW	0.054	16.316	444.853	63.163
GRI	0.017	0.621	1.081	0.385
2000 Iteration				
Adaptation 100 Iteration				
	Best	Mean	Worst	Std. Dev.
ROS	0.525	25.401	29.152	5.080
RAS	0.000	0.063	0.690	0.113
SCW	0.001	5.915	280.429	39.618
GRI	0.000	0.072	0.500	0.124

Table 10: T-test: Adpative Metaheuristic vs Adaptive GA

1000 Iteration					
	Adaptive GA		Adaptive MH		
	Mean	Std. Dev	Mean	Std. Dev	t
ROS	248.705	169.331	33.126	18.076	**8.951**
RAS	43.533	19.68	9.561	26.413	**7.293**
SCW	3999.956	1046.521	16.316	63.163	**26.867**
GRI	0.956	0.136	0.621	0.385	**5.801**
2000 Iteration					
	Adaptive GA		Adaptive MH		
	Mean	Std. Dev	Mean	Std. Dev	t
ROS	130.863	902.405	25.401	5.08	0.826
RAS	31.1575	18.4707	0.063	0.113	**11.904**
SCW	1349.22	1381.812	5.915	39.618	**6.871**
GRI	0.573	0.241	0.072	0.124	**13.071**

Constructive Metaheuristics for the Set Covering Problem

Broderick Crawford[1], Ricardo Soto[1], Gino Astorga[1,2], and José García[1,3]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile,
{broderick.crawford,ricardo.soto}@pucv.cl
[2] Universidad de Valparaíso, Valparaíso, Chile,
gino.astorga@uv.cl
[3] Centro de Investigación y Desarrollo Telefónica, Santiago, Chile,
joseantonio.garcia@telefonica.com

Abstract. Different criteria exist for the classification of the metaheuristics. One important classification is: improvement metaheuristics and constructive. On the one hand improvement metaheuristics, begins with an initial solution and iteratively improves the quality of the solution using neighborhood search. On the other hand, constructive metaheuristics, are those in which a solution is built from the beginning, finding in each iteration a local optimum. In this article, we to compare two constructive metaheuristics, Ant Colony Optimization and Intelligent Water Drops, by solving a classical NP-hard problem, such like the Set Covering Problem, which has many practical applications, including line balancing production, service installation and crew scheduling in railway, among others. The results reveal that Ant Colony Optimization has a better behavior than Intelligent Water Drops in relation to the problem considered.

Keywords: Intelligent Water Drops, Set Covering Problem, Constructive Metaheuristic

1 Introduction

The Set Covering Problem (SCP) is a NP-hard problem [10], which consists into find a subset of columns in a zero-one matrix such that they cover all the rows of the matrix at a minimum cost. Some of its applications includes line balancing production, emergency services location, crew scheduling in mass-transit companies, logical analysis of numerical data, metallurgical production, vehicle routing and treatment of boolean expressions.

Given the complex nature of the SCP and the huge size of real datasets, the problem has been studied and solved by several metaheuristics, such like genetic algorithms [3], simulated annealing [4], indirect genetic algorithms [1], ant colony optimization [11], cat swarm optimization [5], cuckoo search [14] and a meta optimization [7],among others.

Constructive metaheuristics for the SCP includes Ant Colony Optimization and Meta-RaPS. The aim of this article is to study the performance of the

Ant Colony Optimization (ACO) and Intelligent Water Drops (IWD) algorithms applied to the SCP. These constructive metaheuristics were introduced by: [8] to solve the Traveling Salesman Problem (TSP), based on the behavior of ant colonies and [13] to solve the Multiobjective Knapsack Problem (MKP) and it is based on the behavior of natural water drops flowing in the beds of rivers, carrying soil and moving between different branches to reach their destination; the interesting on this is that the path constructed to the destination tends to be optimal, despite the obstacles in the environment.

This article is organized as follows: In Section 2 we describe the SCP, in Section 3 we present the ACO algorithm, , in Section 4 we present the IWD algorithm, in Section 5 we describe the results obtained for several SCP instances and finally in section 6, we present the conclusions and future work.

In the construction phase of the solution a degree of randomness must be incorporated, with the aim of avoiding that the same solution is built in each iteration. Each iteration ends when the solution is found, therefore in this type of metaheuristics is avoided the problem of the infeasibility.

2 Set Covering Problem

The SCP consists into find a subset of columns in a zero-one matrix such that they can cover all the rows of that matrix at a minimum cost.

The SCP can be defined as:

$$Minimize \quad Z = \sum_{j=1}^{n} c_j x_j \tag{1}$$

Subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \in I \tag{2}$$

$$x_j \in \{0, 1\} \quad \forall j \in J \tag{3}$$

Let $A = (a_{ij})$ be a $m \times n$ binary matrix with $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ being the row and column sets respectively. We say that a column j can cover a row i if $a_{ij} = 1$. The cost of selecting the column j is represented by c_j, a non-negative value, and x_j is a decision variable to indicate if the column j is selected ($x_j = 1$) or not ($x_j = 0$).

One of the many practical applications of this problem is the location of fire stations. Lets consider a city divided in a finite number of areas which need to locate and build fire stations. Each one of this areas need to be covered by at least one station, but a single fire station can only bring coverage to its own area and the adjacent ones; also, the problem requires that the number of stations to build needs to be the minimum.

Being the SCP a NP-hard class problem, one of the many difficulties that benchmarks arise is regarding their size and the computational cost associated. To solve this, many authors have proposed to simplify (or "pre-process") the problem before apply any exact method or metaheuristic. By doing this, we are dealing with problems that are equivalent to original but smaller in terms of rows and columns. We introduce a preprocessing phase before run the metaheuristic; the goal of this phase is to reduce the size of instances and improve the performance of the algorithm. In this article, we use two methods that have proven to be more effective: Column Domination and Column Inclusion, presented in [2] and [9] respectively.

3 Ant Colony Optimization

Ant Colony Optimization Algorithm (ACO) it was inspired by the behavior of ant colonies in the search for their food, was proposed by [8], is a probabilistic technique that allows to find the shortest path in a graph. In the nature, the ants leave a chemical signal called pheromone, in the path through which pass. The pheromone has an important role in the survival of the ants allowing to find the shortest way to its power supply.

An ant exploratory moves in random searching for food, depositing pheromone in its path which is followed by more ants which reach the source of food found. When transiting more ants by this path, the amount of pheromone will be increased reinforcing the path. If there are two paths to the same food source, the shortest path will be the busiest because of its amount of pheromone, considering that in the longest path the pheromone will disappear because it is volatile. The behavior of the ants is shown in the Figure 1.

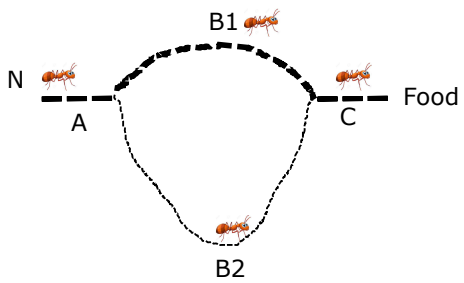


Fig. 1. Path of the ants to its source of food.

The main idea, is to model an optimization problem as the search for the lowest cost route in a graph by a colony of artificial ants. The ACO algorithms are essentially constructive, that is to say, for each iteration all the ants build a solution depositing pheromone, according to the quality of the solution, allowing to guide to the rest of the ants of the colony.

The ACO Algorithm can be applied directly to the SCP. The columns are chosen as the solution components and have associated a cost and a pheromone trail. Constraints say that each column can be visited by an ant once and only once and that a final solution has to cover all rows.

A walk of an ant over the graph representation corresponds to the iterative addition of columns to the partial solution obtained so far. Each ant starts with an empty solution and adds columns until a cover is completed. A pheromone trail τ_j and a heuristic information η_j are associated to each eligible column j. A column to be added is chosen with a probability that depends of pheromone trail and the heuristic information. The probability function is given by the equation Equation 5:

$$p_j^k(t) = \frac{\tau_j \eta_j^\beta}{\sum_{l \in N^k} \tau_l [\eta_l]^\beta} \quad \text{if } j \in N^k \tag{4}$$

where N^k is the feasible neighborhood of the ant k. The β parameter controls how important is η in the probabilistic decision. τ_j is the pheromone and η_j is the heuristic information.

In this work we use a dynamic heuristic information η_j that depends on the partial solution of an ant. We defined as $\eta_j = \frac{e_j}{c_j}$, where e_j is the so called cover value, that is, the number of additional rows covered when adding column j to the current partial solution, and c_j is the cost of column j. An ant ends the solution construction when all rows are covered.

An important step in ACO Algorithm is the pheromone update on the path. The pheromone trails are updated as given by the following equation:

$$\tau_j = p\ \tau_j + \omega_i, \quad \forall j \in J \tag{5}$$

where $p \in J\ [0, 1)$ is the pheromone persistence and $\omega_i \geq 0$ is the amount of pheromone put on column j.

The general pseudocode for the ACO is presented in Algorithm 1

Algorithm 1 Ant Colony System

```
{Step 1: initialize parameters}
while not stop condition do
    {Step 2: All the ants in the colony generate a solution}
    for each ant in the nest do
        generate a new solution
    end for
    {Step 3: Update}
    update local optimal
    update pheromone trails
end while
{Step 4: Best solution}
return the best solution found
```

Step 1. Algorithm parameters are initialized.

Step 2. Once each ant generated a solution, the local optimal solution is updated if needed. Due to evaporation, all pheromone trails are decreased uniformly. The ants deposit some amount of pheromone on good solution to mark promising areas for next iterations.

Step 3. The algorithm ends when a certain stop condition is reached

Step 4. The best solution found is returned

The Set Covering Problem has been solved by the following variants of the algorithm: Ant Colony Optimization (ACO), Ant Colony System (ACS), Max-Min Ant System (MMA'S) and Hyper-Cube Framework for ACO [12].

4 Intelligent Water Drops Algorithm

The IWD algorithm is a population-based constructive metaheuristics proposed in [13] and designed to imitate the flow properties of natural water drops, which tend to describe an optimal path to their destination considering the distance and despite the constraints in the environment.

In the algorithm, the population is composed by N water drops (denoted by IWD^k, $k \in [1..N]$) moving in a discrete environment, represented by a graph with N_c nodes in which the drops will move from one node to another.

Each drop has two main properties: the amount of soil it carries (denoted by $soil^k$) and the velocity (denoted by vel^k). Both properties can change how the drop flows in the environment.

In case of soil, it is expected that as iterations passes, the amount of soil carried by each drop will increase, making the drop bigger (Figure 2A). Also the velocity in a water drop determines the amount of soil removed; the faster the water drop is, the bigger the amount of soil removed (Figure 2B). However, the velocity of a water drop can increase or decrease according to the branch chosen in each iteration.

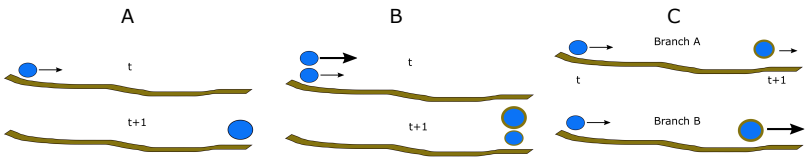


Fig. 2. Behavior of water drops in a river.

Choosing a branch (of the path) or another depends basically on how "desirable" -in terms of the amount of soil- the branch is; so, if the branch has a

high amount of soil then that branch is more difficult to flow (more soil, less velocity) than another branch with a less amount (Figure 2C). In the algorithm, this behavior is implemented by a probabilistic function of inverse of soil.

The IWD algorithm has been applied to several problems like: air robot path planning, smooth trajectory planning, vehicle routing problem, economic load dispatch problem, image processing, rough set feature selection, reservoir operation, code coverage, data aggregation in wireless sensor networks, multi-objective job shop scheduling, among others.

The general pseudocode for the IWD is presented in Algorithm 2

Algorithm 2 Intelligent Water Drops

```
{Step 1: Static Parameters Initialization}
Initialize parameters N, MAX_ITERATION, N_c.
Initialize soil parameters: a_s, b_s, c_s
Initialize velocity parameters: a_v, b_v, c_v
for i = 1 to MAX_ITERATION do
    {Step 2: Dynamic Parameter Initialization}
    for k = 1 to N do
        Initialize VC^k list as empty.
        Initialize soil^k value as zero.
        Initialize vel^k value as InitVel.
    end for
    {Step 3: Create and Distribute Water Drops}
    for k = 1 to N do
        Create the k^th water drop (IWD^k).
        Select randomly a node for IWD^k.
    end for
    {Step 4: Update Visited List of Water Drops}
    for k = 1 to N do
        Update VC^k.
    end for
    {Step 5: Complete Each Water Drop Solution}
    for k = 1 to N do
        Choose a path for IWD^k.
        Update velocity (vel^k).
        Compute the amount of soil (Δsoil) to be carried by IWD^k
        Remove Δsoil from the path and add it to IWD^k
    end for
    {Step 6: Find the Best Solution from the Iteration}
    for k = 1 to N do
        Calculate fitness^k
    end for
    {Step 7: Update Paths of the Best Solution from Iteration}
    {Step 8: Update the Total Best Solution}
end for
```

Step 1. In this step we set the static parameters to run the IWD algorithm; all of these values will remain constant during execution and can only change between experiments.

The two properties of intelligent water drops -soil and velocity- also have parameters to set in this step; these parameters are constants required to update the soil and velocity values in each iteration. For soil, parameters are: a_s, b_s and c_s; for velocity, parameters are: a_v, b_v, c_v.

Step 2. Each water drop (denoted by IWD^k, $k \in [1...N]$) need to update certain values in each iteration: a list of nodes visited, the value for soil property and the value for velocity property.

In case of nodes, a list (denoted by VC^k) will be updated by adding the last node visited; in case of the first iteration, this list will be set to empty ($VC^k = \{\}$).

For soil and velocity, the first iteration will set arbitrarily these values to the static parameters of *InitSoil* and *InitVel* respectively.

Step 3. In this step the water drops are created and then distributed randomly along the nodes. At this point, we do not use any probability function yet.

Step 4. With all water drops distributed, we can update their list of visited nodes by adding the nodes from Step 4.

Step 5. This step will build a valid path to solution by moving the recently created water drops across nodes and updating their soil and velocity properties each time. All steps described here will be executed in a loop until reach a solution.

Step 5.1. Select the next node (called j) to be visited by water drop IWD^k. Considering a water drop at node i and with a visited list of VC^k, this step will look at all nodes that have not been visited yet and will select one of them according to a probability function based on the amount of soil present in the path to that next node (Equation 6).

$$p_i^k(j) = \frac{f(soil(i,j))}{\sum_{\forall l \notin VC^k} f(soil(i,l))} \tag{6}$$

The function $f(soil(i,j))$ represents the inverse amount of soil in the path between nodes i and j respectively and uses a constant ε with the solely purpose to avoid zero division (Equation 7).

$$f(soil(i,j)) = \frac{1}{\varepsilon + g(soil(i,j))} \tag{7}$$

The function $g(soil(i,j))$ is introduced to always get a positive value when calculating the amount of soil between nodes i and j respect to the amount of soil in the visited path (Equation 8).

$$g(soil(i,j)) \begin{cases} soil(i,j) \;\; if \min\limits_{l \notin vc(IWD)} (soil(i,l)) \geq 0 \\ soil(i,j) - \min\limits_{l \notin vc(IWD))} (soil(i,l)) \; else \end{cases} \tag{8}$$

Step 5.2. Update velocity of the water drop. As long as the water drop moves between nodes i and j certain amount of soil will be carried by the drop, turning the drop bigger. To calculate this change, the algorithm will use a function based on the soil in the path between i and j (Equation 9).

$$vel^k(t+1) = vel^k(t) + \frac{a_v}{b_v + c_v \cdot soil(i,j)} \tag{9}$$

Step 5.3. Update soil amount of the water drop. Depending on the velocity value of the water drop when moving between i and j, the amount of soil removed from environment ($soil(i,j)$) and carried by the drop for the next iteration ($soil^k(t+1)$) will be different (Equations 10, 11 and 12). The more time it takes, the more soil the water drop will carry (Equation 13).

$$soil(i,j) = (1-\rho) \cdot soil(i,j) - \rho \cdot \Delta soil(i,j) \tag{10}$$

$$soil^k(t+1) = soil^k(t) + \Delta soil(i,j) \tag{11}$$

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s \cdot time(i,j : vel^k(t+1))} \tag{12}$$

$$time(i,j : vel^k(t+1)) = \frac{HUD(i,j)}{vel^k(t+1)} \tag{13}$$

The $HUD(i,j)$ is a local heuristic function proposed in [13] to measure the undesirability of the water drop to move from one node to another. In this article, we applied the same idea for SCP; the heuristic function applied considers the cost of moving to node j ($cost_j$) and the number of constraints covered by the node (R_j); the function is presented in Equation 14.

$$HUD(i,j) = \frac{cost_j}{R_j \notin VC^R} \tag{14}$$

Step 5.4. Remove soil from the path and add it to the water drop. In previous steps, we calculated the amount of soil removed from the path ($\Delta soil$); now, the value of soil property for each water drop ($soil^k$) needs to be updated.

Step 6. Once all water drops have completed their solutions, the algorithm requires to evaluate which one was the best in the current iteration (T^{IB}); to do this, we have to consider specifically the problem we are solving -SCP in this case- where the best solution is given by the one with the minimum cost associated. The equation that will compute this step is presented in Equation 15.

$$T^{IB} = \arg\min q(IWD^k) \tag{15}$$

Step 7. The path traveled by the best water drop in the iteration (T_{IB}) will modify the environment for the future drops. In this step, the algorithm will update the amount of soil in the arcs of the graph that were traveled by T^{IB} in order to reflect the impact of this drop on them.

The soil update then, will be done based on Equation 10 but considering the quality of the best solution found during the current iteration (T^{IB}). As better the solution is, then more soil will be removed. In the algorithm, this is calculated in terms of the soil present during the current iteration and the quality of the iteration-best solution (Equation 16). The parameter ρ_{IWD} is a negative constant.

$$soil(i,j) = (1 - \rho_{IWD}) \cdot soil(i,j) + \rho_{IWD} \cdot soil(i,j)^{IB} \cdot \frac{1}{q(T^{IB})} \tag{16}$$

Step 8. Finally, if the iteration-best solution is better than the global-best, then the global-best solution needs to be replaced with the new one (Equation 17).

$$T^{TB} = \begin{cases} T^{IB} & \text{if } q(T^{TB}) > q(T^{IB}), \\ T^{TB} & \text{else} \end{cases} \tag{17}$$

5 Computational Results

The proposed IWD algorithm has been implemented in Java language in an Intel CORE i7 CPU, 8GB of RAM, running Windows 7 Ultimate 64 bit.

5.1 Parameters

With the objective of finding a better behavior of the algorithm we consider different configurations for the static parameters based on the size of the families of instances. The values of each family is obtained when running the algorithm 10 times by varying the parameters: Number of intelligent water drops, Maximum number of iterations and initial soil value. The selected configuration is the one which corresponds to the solution with best RPD.

The parameters tuning for the IWD algorithm is detailed in Table 1 considering the different instances families.

Table 1. Tuning for Static Parameters in IWD.

Dataset	m	n	N	$MAX\ ITERATION$	$Initial\ Soil$	a_s	b_s	c_s	$Initial\ Vel$	a_v	b_v	c_v	ε	ρ
4	200	1000	250	300	1600	1000	0.01	1	3	10	0.01	1	0.01	0.95
5	200	2000	300	600	1600	1000	0.01	1	3	10	0.01	1	0.01	0.95
6	200	1000	250	600	1600	1000	0.01	1	3	10	0.01	1	0.01	0.95

The instances tested are from Beasley's OR Library. Details on instances are presented in Table 2.

Table 2. Set Covering Instances.

Instance set	No. of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1000	[1, 100]	2	Known
5	10	200	2000	[1, 100]	2	Known
6	5	200	1000	[1, 100]	5	Known

After performed experiments for the 3 families presented before, results can be seen at Table 3. This tables, presents the instance number, the optimum result known (Z_{opt}), the minimum result obtained by our experiments (Z_{min}), the average result obtained (Z_{avg}) and the relative percentage deviation (RPD) for IWD with Pre-Processing and IWD [6] techniques.
The RPD value quantifies the deviation of the objective value Z_{min} from the optimal known Z_{opt}. To calculate it, we use Equation 18

$$RPD = \left(\frac{Z_{min} - Z_{opt}}{Z_{opt}} \right) \times 100 \tag{18}$$

Table 3. Computational Results

Instance	Z_{opt}	Z_{min}	Z_{avg}	RPD_{ACO}	Z_{min}	Z_{avg}	$RPD_{IWD_{preprocess}}$	$Diff_{RPD}$
4,1	429	443	449	0,23	430	434	3,26	92,94
4,2	512	560	579	0,00	512	518	9,38	100,00
4,3	516	546	561	0,00	516	518	5,81	100,00
4,4	494	536	554	0,20	495	501	7,84	97,45
4,5	512	552	592	0,39	514	519	7,81	95,01
4,6	560	614	263	2,68	575	575	9,64	72,20
4,7	430	456	478	0,70	433	437	6,05	88,43
4,8	492	536	568	0,41	494	497	8,94	95,41
4,9	641	706	721	0,78	646	654	10,14	92,31
4,10	514	586	596	0,78	518	522	14,01	94,43
5,1	253	277	301	0,79	255	259	9,49	91,68
5,2	302	334	357	1,32	306	309	10,60	87,55
5,3	226	245	264	2,65	232	234	8,41	68,49
5,4	242	263	291	0,41	243	243	8,68	95,28
5,5	211	232	254	0,47	212	212	9,95	95,28
5,6	213	234	250	0,00	213	216	9,86	100,00
5,7	293	325	351	1,71	298	298	10,92	84,34
5,8	288	316	344	0,35	289	289	9,72	96,40
5,9	279	325	355	0,72	281	282	16,49	95,63
5,10	265	291	302	1,13	268	269	9,81	88,48
6,1	138	156	181	2,90	142	145	11,54	74,87
6,2	146	184	194	4,79	153	156	20,65	76,80
6,3	145	172	189	0,00	145	149	15,70	100,00
6,4	131	152	181	2,29	134	136	13,82	83,43
6,5	161	188	198	0,62	162	169	14,36	95,68

In accordance with the results showed in the table 3 it is visualized that ACO was mejor in all the instances achieving 4 optimals, however also it is appreciated that the RPD of IWD is similar for all instances.

6 Conclusions and Future Work

This article presents the comparison of two constructive metaheuristics (IWD and ACO), solving a classic combinatorial problem as SCP which has been used for modeling problems of the industry. For this problem ACO had a better behavior than IWD, reaching 4 optimum for the instances used in contrast to IWD that did not reach any. We used the instances of the groups 4,5,6. For group 4, ACO obtained 2 optimum, 1 in group 5 and 1 in group 6. In order to improve the behavior of IWD, it incorporated a stage of preprocessing which helped to improve the response but without reaching a reach some optimal. Although the results of IWD have not been better than ACO, these are encouraging. In the same line, the future work proposed it is related to improve the tuning of parameters to improve the results. Also, another very interesting line is related to test the algorithm for the remaining instances from OR Library and other SCP libraries, such like the Unicost (available at OR-Library website), Italian railways, American airlines and the Euclidean benchmarks.

Acknowledgements

Broderick Crawford is supported by grant CONICYT / FONDECYT / REGULAR 1171243 and Ricardo Soto is supported by Grant CONICYT / FONDECYT / REGULAR / 1160455, Gino Astorga is supported by Postgraduate Grant, Pontificia Universidad Catolica de Valparaíso, 2015 and José García is supported by INF-PUCV 2016. The authors are grateful for the support of the Project CORFO 14ENI2-26905 "Nueva Ingeniería para el 2030" PUCV.

References

1. Uwe Aickelin. An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society*, 53:1118–1126, 2002.
2. John Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31:85–93, 1987.
3. John Beasley and P. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94:392–404, 1996.
4. Michael Brusco, Larry Jacobs, and Garry Thompson. A morphing procedure to supplement a simulated annealing heuristic for cost and coverage correlated set covering problems. *Annals of Operations Research*, 86:611–627, 1999.
5. Broderick Crawford, Ricardo Soto, Natalia Berríos, Franklin Johnson, Fernando Paredes, Carlos Castro, and Enrique Norero. A binary cat swarm optimization algorithm for the non-unicost set covering problem. *Mathematical Problems in Engineering*, 2015, 2015.
6. Broderick Crawford, Ricardo Soto, Jorge Córdova, and Eduardo Olguín. A nature inspired intelligent water drop algorithm and its application for solving the set covering problem. In *Artificial Intelligence Perspectives in Intelligent Systems*, pages 437–447. Springer, 2016.
7. Broderick Crawford, Ricardo Soto, Eric Monfroy, Gino Astorga, José García, and Enrique Cortes. A meta-optimization approach for covering problems in facility location. In *Workshop on Engineering Applications*, pages 565–578. Springer, 2017.

8. Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.
9. Marshall Fisher and Pradeep Kedia. Optimal solution of set covering/partitioning problems using dual heuristics. *Management science*, 36(6):674–688, 1990.
10. Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.
11. Lucas Lessing, Irina Dumitrescu, and Thomas Stützle. A comparison between aco algorithms for the set covering problem. In *Proceedings of ANTS 2004, Lecture Notes in Computer Science*, volume 3172, pages 1–12, 2004.
12. Lucas Lessing, Irina Dumitrescu, and Thomas Stützle. A comparison between aco algorithms for the set covering problem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 1–12. Springer, 2004.
13. Hamed Shah-Hosseini. Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem. *International Journal of Intelligent Computing and Cybernetics*, 1(2):193–212, 2008.
14. Ricardo Soto, Broderick Crawford, Cristian Galleguillos, Jorge Barraza, Sebastián Lizama, Alexis Muñoz, José Vilches, Sanjay Misra, and Fernando Paredes. Comparing cuckoo search, bee colony, firefly optimization, and electromagnetism-like algorithms for solving the set covering problem. In *Computational Science and Its Applications–ICCSA 2015*, pages 187–202. Springer, 2015.

An Approach for Recovering Distributed Systems from Disasters

Ichiro Satoh

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
ichiro@nii.ac.jp

Abstract. This paper presents an approach to recovering distributed applications, which consist of software agents running on different computers from drastic damages by disasters. The approach is inspired from regeneration mechanisms in living things, e.g., tails of lizards. When an agent delegates a function to another agent coordinating with it, if the former has the function, this function becomes less-developed and the latter's function becomes well-developed like differentiation processes in cells. It can also initialize and restart differentiated software agents, when some agents cannot be delegated like regeneration processes. It is constructed as a general-purpose and practical middleware system for software agents on real distributed systems consisting of embedded computers or sensor nodes.

1 Introduction

Hundreds of natural disasters occur in many parts of the world every year, causing billions of dollars in damages. This fact may contrast with the availability of distributed systems. Distributed systems are often treated to be dependable against damages, because in distributed systems data can be stored and executed at multiple locations and processing must not be performed by only one computer. However, all existing distributed systems are not resilient to damages in the sense that if only one of the many computers fails, or if a single network link is down, the system as a whole may become unavailable. Furthermore, in distributed systems partially damaged by disasters surviving computers and networks have no ability to fill functions lost with damaged computers or networks.

On the other hand, several living things, including vertebrates, can *regenerate* their lost parts, where *regeneration* is one of developmental mechanisms observed in a number of animal species, e.g., lizard, earthworm, and hydra, because regeneration enables biological systems to recover themselves against their grave damages. For example, reptiles and amphibians can partially regenerate their tails, typically over a period of weeks after cutting the tails. *Regeneration* processes are provided by (de)differentiation mechanism by which cells in a multicellular organism become specialized to perform specific functions in a variety of tissues and organs. The key idea behind the approach proposed in this paper was inspired from *(de)differentiation* as a basic mechanism for

regeneration like living things. The approach introduces a (de)differentiation mechanism into middleware systems for distributed systems, instead of any simulation-based approaches.[1]

Our middleware system aims at building and operating distributed applications consisting of self-adapting / tuning software components, called agents, to regenerate / differentiate their functions according to their roles in whole applications and resource availability, as just like cells. It involves treating the undertaking/delegation of functions in agents from/to other agents as their differentiation factors. When an agent delegates a function to another agent, if the former has the function, its function becomes less-developed in the sense that it has less computational resources, e.g., active threads, and the latter's function becomes well-developed in the sense that it has more computational resources.

2 Example Scenario

Let us suppose a sensor network to observe a volcano. Its sensor nodes are located around the volcano. Each of the nodes have sensors to measure accelerations result from volcano tectonic earthquakes around it in addition to processors and wired or wireless network interfaces. The locations of sensor nodes tend to be irregular around the volcano,

A disaster may result in drastic damages in sensor networks. For example, there are several active or dormant volcanos in Japan. Sensor networks to detect volcano ash and tremor are installed at several spots in volcanoes. Volcanic eruptions, including phreatic eruptions, seriously affect such sensor networks. More han half sensor nodes may be damaged by eruptions. Nevertheless, the sensor networks should continue to monitor volcano tectonic earthquakes with only their surviving nodes as much as possible.

Sensor nodes in a volcano are located irregularly, because it is difficult for people to place such nodes at certain positions in volcanos, because there are many no-go zones and topographical constraints. Instead, they are distributed from manned airplanes or unmanned ones. Therefore, they tend to be overpopulated in several areas in the sense that the coverage areas of their sensors are overlap or contained. To avoid congestion in networks as well as to save energy consumption, redundant nodes should be inactivated.

3 Requirements

To support example scenarios discussed in the previous section, our approach needs to satisfy the following requirements: *Self-adaptation* is needed when environments and users' requirements change. To save computational resources and energy, distributed systems should adapt their own functions to changes in their systems and environments. *Saving resources* is important in distributed systems used in field, e.g., sensor networks, rather than data centers, including cloud computing. Our approach should conserve limited computational resources, e.g., processing, storage resources, networks, and energy,

[1] There is often a gap between the real systems and simulations. We believe that adaptive distributed systems need more experiences in the real systems.

at nodes as much as possible. *Non-centralized management* can support reliability and availability. Centralized management may be simple but can become a single point of failures. Therefore, our adaptation should be managed in a peer-to-peer manner. Distributed systems essentially lack no global view due to communication latency between computers. Software components, which may be running on different computers, need to coordinate them to support their applications with partial knowledge about other computers. Our approach should be practical so that it is implemented as a general-purpose middleware system. This is because applications running on distributed systems are various. Each of software components should be defined independently of our adaptation mechanism as much as possible. As a result, developers should be able to concentrate their application-specific processing.

4 Approach: Regeneration and Differentiation

The goal of the proposed approach is to introduce a *regeneration* mechanism into distributed systems like living things. Regenerations in living things need redundant information in the sense that each of their cells have genes as plans for other cells. When living things lose some parts of their bodies, they can regenerate such lost parts by encoding genes for building the parts with differentiation mechanisms. Differentiation mechanisms can be treated as selections of parts of genes to be encoded. Since a distributed application consists of software components, which may be running on different computers like cells, we assume that software components have program codes for functions, which they do not initially provide and our differentiation mechanisms can select which functions should be (in)activated or well/less-developed.

Each software component, called agent, has one or more functions with weights, where each weight indicates the superiority and development of its function in the sense that the function is assigned with more computational resources. Each agent initially intends to progress all its functions and periodically multicasts messages about its differentiation to other agents of which its distributed application consist. Such messages lead other agents to degenerate their functions specified in the messages and to decrease the superiority of the functions. As a result, agents complement other agents in the sense that each agent can provide some functions to other agents and delegate other functions to other agents that can provide the functions.

5 Design

Our approach is maintained through two parts: runtime systems and agents. The former is a middleware system for running on computers and the latter is a self-contained and autonomous software entity. It has three protocols for regeneration/differentiation.

5.1 Agent

Each agent consists of one or more functions, called the *behavior* parts, and its state, called the *body* part, with information for (de)differentiation, called the *attribute* part.

The body part maintains program variables shared by its behaviors parts like instance variables in object orientation. When it receives a request message from an external system or other agents, it dispatches the message to the behavior part that can handle the message. The behavior part defines more than one application-specific behavior. It corresponds to a method in object orientation. As in behavior invocation, when a message is received from the body part, the behavior is executed and returns the result is returned via the body part. The attribute part maintains descriptive information with regard to the agent, including its own identifier. The attributes contains a database for maintaining the weights of its own behaviors and for recording information on the behaviors that other agents can provide.

5.2 Regeneration

We outline our differentiation processes for regeneration (Fig. 1) . The Appendix describes the processes in more detail.

- *Invocation of behaviors:* Each agent periodically multicasts messages about the weights of its behaviors to other agents. When an agent wants to execute a behavior, even if it has the behavior, it compares the weights of the same or compatible behaviors provided in others and it. It select one of the behaviors, whose weights are the most among the weights of these behaviors. That is, the approach selects more developed behaviors than less developed behaviors.
- *Well/Less developing behaviors:* When a behavior is executed by other agents, the weight of the behavior increase and the weights of the same or behaviors provided from others decrease. That is, behaviors in an agent, which are delegated from other agents more times, are well developed, whereas other behaviors, which are delegated from other agents fewer times, in a cell are less developed.
- *Removing redundant behaviors:* The agent only provides the former behaviors and delegates the latter behaviors to other agents. Finally, when the weights of behaviors are zero, the behaviors become dormant to save computational resources.
- *Increasing resources for busy behaviors:* Each agent can create a copy of itself when the total weights of functions provided in itself is the same or more than a specified value. The sum of the total weights of the mother agent and those of the daughter agent is equal to the total weights of the mother agent before the agent is duplicated.
- *Reactivating dormant behaviors:* When an agent does not receive messages about the weights of behaviors provided in agents, treats such behaviors to be lost. When it has the same or compatible behaviors, which are dormant, it resets the wights of the behaviors, to their initial values. Therefore, they are regenerated and differentiated according to the above process again.

6 Implementation

To evaluate our proposed approach, we constructed it as a middleware system with Java (Figure 2), which can directly runs on Java VM running on VMs in IaaS, e.g., Amazon

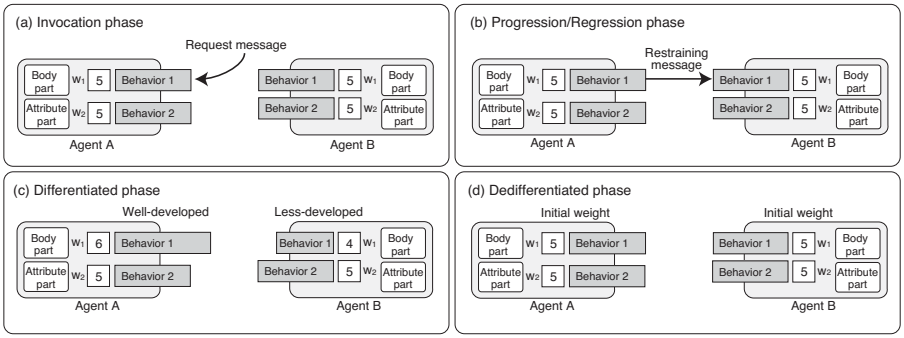


Fig. 1. Regeneration in Agents

EC2. It is responsible for executing duplicating, and deploying agents based on several technologies for mobile agent platforms. It is also responsible for executing agents and for exchanging messages in runtime systems on other IaaS VMs or PaaS runtime systems through TCP and UDP protocols. Messages for exchanging information about the weights of differentiation are transmitted as multicast UDP packets. Application-specific messages for invoking methods corresponding to behaviors in agents are implemented through TCP sessions.

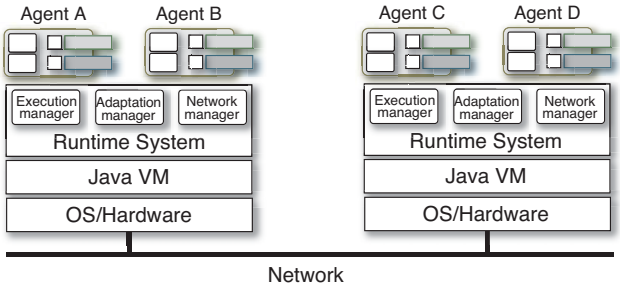


Fig. 2. Runtime system

Each agent is an autonomous programmable entity. The body part maintains a key-value store database, which is implemented as a hashtable, shared by its behaviors. We can define each agent as a single JavaBean, where each method in JavaBean needs to access the database maintained in the body parts. Each method in such a JavaBean-based agent is transformed into a Java class, which is called by another method via the body part, by using a bytecode-level modification technique before the agent is executed. Each body part is invoked from agents running on different computers via our original remote method invocation (RMI) mechanism, which can be automatically handled in network disconnection unlike Java's RMI library. The mechanism is managed by run-

time systems and provided to agents to support additional interactions, e.g., one-way message transmission, publish-subscription events, and stream communications. Since each agent records the time the behaviors are invoked and the results are received, it selects behaviors provided in other agents according to the average or worst response time in the previous processing. When a result is received from another agent, the approach permits the former to modify the value of the behavior of the latter under its own control. For example, agents that want to execute a behavior quickly may increase the weight of the behavior by an extra amount, when the behavior returns the result too soon.

7 Evaluation

This section describes the performance evaluation of our implementation.

7.1 Basic performance

Although the current implementation was not constructed for performance, we evaluated several basic operations in distributed systems consisting of eights embedded computers, where each computer is a Raspberry Pi computer, which has been one of the most popular embedded computers (its processor was Broadloom BCM2835 (ARM v6-architecture core with floating point) running at 700 Mhz and it has 1 GB memory and SD card storage (16 GB SDHC), with a Linux operating system optimized to Raspberry Pi, and OpenJDK. The cost of transmitting a message through UDP multicasting was 17 ms. The cost of transmitting a request message between two computers was 28 ms through TCP. These costs were estimated from the measurements of round-trip times between computers. We assumed in the following experiments that each agent issued messages to other agents every 110 ms through UDP multicasting.

We evaluated the speed of convergence in our differentiation. Each computer had one agent having three functions, called behavior A, B and C, where behavior A invoked B and C behaviors every 200 ms and the B and C behaviors were null behaviors. We assigned at most one agent to each of the computers. B or C, selected a behavior whose weight had the highest value if its database recognized one or more agents that provided the same or compatible behavior, including itself. When it invokes behavior B or C and the weights of its and others behaviors were the same, it randomly selected one of the behaviors. We assumed in this experiment that the weights of the B and C behaviors of each agent would initially be five and the maximum of the weight of each behavior and the total maximum of weights would be ten.

Differentiation started after 200 ms, because each agent knows the presence of other agents by receiving heartbeat messages from them. The right of Fig. 3 details the results obtained from our differentiation between four agents on four computers and The left of Fig. 3 between eight agents on eight computers. Finally, two agents provide behavior B and C respectively and the others delegate the two behaviors to the two agents in both the cases. Although the time of differentiation depended on the period of invoking behaviors, it was independent of the number of agents. This is important to prove that this approach is scalable.

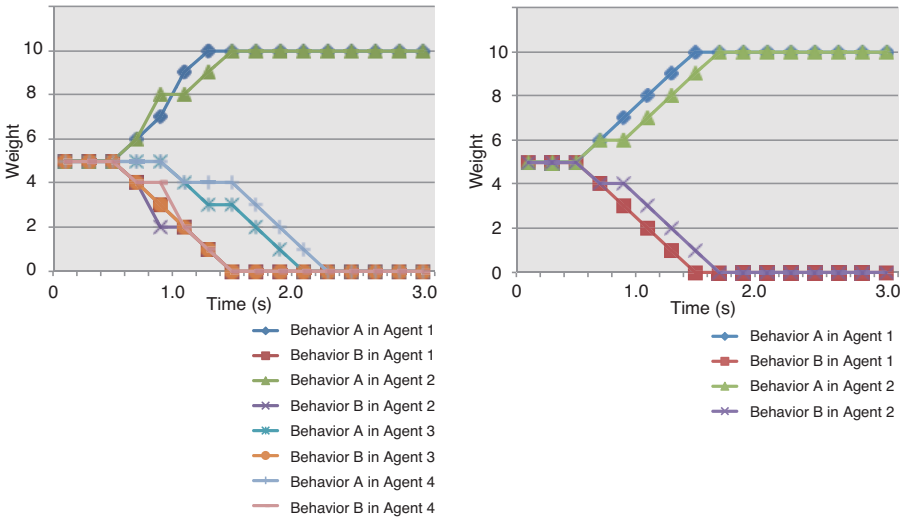


Fig. 3. Convergence in four agents with two behaviors (Left) and Convergence in eight agents with two behaviors (Right)

7.2 Sensor networks recovering from damaged by disasters

Let us suppose a sensor-network system consisting of 15 × 15 nodes connected through a grid network, as shown in Figure 4. The system was constructed on a commercial IaaS cloud infrastructure (225 instances of Amazon EC2 with Linux and JDK 1.7). This experiment permitted each node to communicate with its eights neighboring nodes and the diameter of a circle in each node represents the weight of a behavior. Nodes were connected according to the topology of the target grid network and could multicast to four neighboring runtime systems through the grid network. We assume that each agent monitors sensors in its current node and every node has one agent.

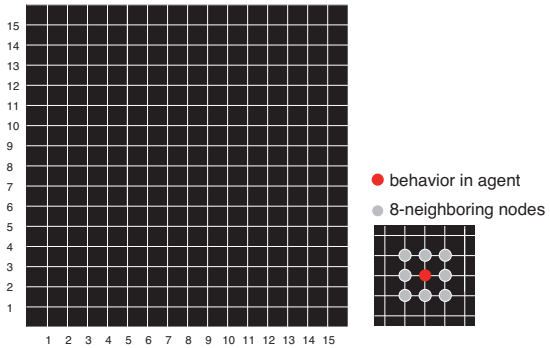


Fig. 4. 15×15-Grid network on cloud computing

We put agents at all nodes and evaluated removing of redundant agents. Each agent has conflict with agents at its eights neighboring nodes, because it can delegate its function to them, vice versa. Figure 5 (i) shows the initial weights of agents. (ii) and (iii) show the weights of behaviors in agents eight and sixteen seconds later. Even though differentiated behaviors were uneven, they could be placed within certain intervals, i.s., two edges on the grid network. This proved that our approach was useful in developing particular functions of software components at nodes.

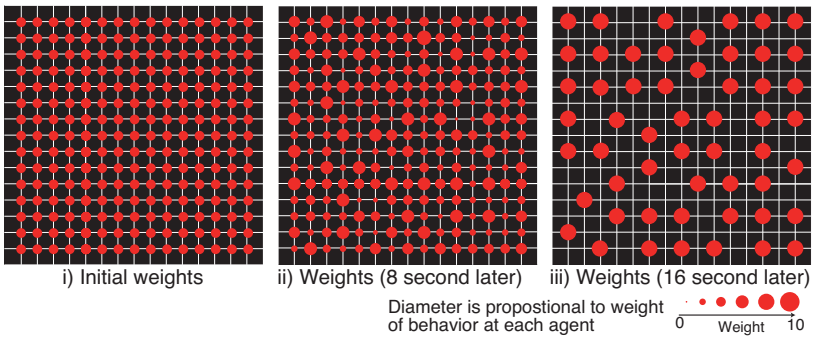


Fig. 5. Removing redundant agents

Figure 6 (i) was the initial weights of agents on the network. We explicitly made a flawed part in the network (Figure 6 (ii)). Some agents dedifferentiate themselves in nodes when a flawed part made in the network. In the experiment agents around the hole started to activate themselves through dedifferentiation. The weights of their behaviors converged according to the weights of their behaviors to the behaviors of other newly activated agents in addition to existing agents. Finally, some agents around the hole could support the behaviors on behalf of the dismissed agents with the flawed part. This result prove that our approach could remedy such a damage appropriatively in a self-organized manner. This is useful for sensing catastrophes, e.g., earthquakes and deluges.

Next, we assume each node could multicast to all agents through the grid network. Figure 7 shows only one agent is activated and the others are inactivated after their differentiations, because the latter can delegate the function to the former. We partitioned the grid network as shown Figure 8 (ii). The above half has a well-developed behavior and the below half lacks such behavior. Therefore, all agents in the below half reset their weights as shown Figure 8 (iii) and they are differentiated. Finally, only one agent is activated on the below half part (Figure 8 (iv)).

8 Related Work

We compare between our approach and other existing bio-inspired approaches for distributed systems. The Anthill project [1] by the University of Bologna developed a

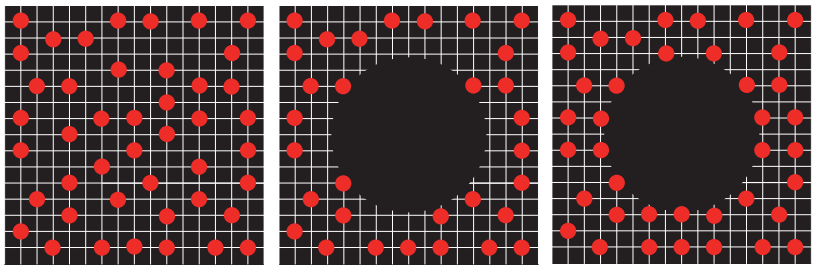


Fig. 6. Regeneration to recover damage

bio-inspired middleware for peer-to-peer systems, which is composed of a collection of interconnected nests. Autonomous agents, called ants can travel across the network trying to satisfy user requests. The project provided bio-inspired frameworks, called Messor [5] and Bison [6]. Messor is a load-balancing application of Anthill and Bison is a conceptual bio-inspired framework based on Anthill. One of the most typical self-organization approaches to distributed systems is swarm intelligence [2,3]. Although there is no centralized control structure dictating how individual agents should behave, interactions between simple agents with static rules often lead to the emergence of intelligent global behavior. Suda et al. proposed bio-inspired middleware, called Bio-Networking, for disseminating network services in dynamic and large-scale networks where there were a large number of decentralized data and services [7,10]. Although they introduced the notion of energy into distributed systems and enabled agents to be replicated, moved, and deleted according to the number of service requests, they had no mechanism to adapt agents' behavior unlike ours. As most of their parameters, e.g., energy, tended to depend on a particular distributed system. so that they may not have been available in other systems. Our approach should be independent of the capabilities of distributed systems as much as possible.

Finally, we compare between our approach and our previous ones, because we constructed several frameworks for adaptive distributed systems. One of them enabled distributed components to be dynamically federated [8]. We also presented an early version of the proposed approach [9], but the version was designed for adaptive services over enrich distributed systems, e.g., cloud computing. They did not support any disaster management.

9 Conclusion

This paper proposed an approach to recovering distributed applications from violent damages, which might result from disasters. The approach is unique to other existing approaches for disaster-tolerant approaches for distributed systems. It was inspired from a bio-inspired mechanism, regeneration in living things. It was also available at the edge of networks, e.g., sensor networks and Internet-of-Thing (IoT). It enabled agents, which were implemented as software components, to be differentiated. When a com-

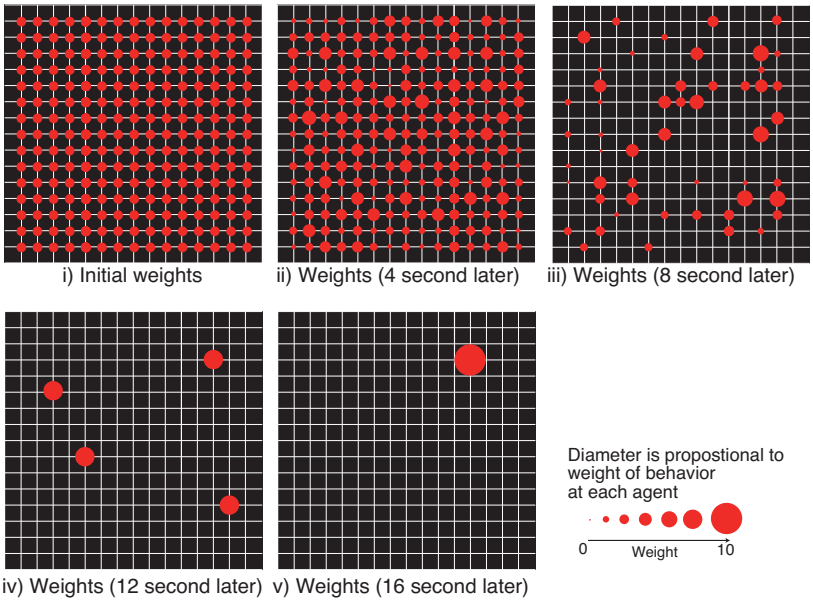


Fig. 7. Agents are differentiated in broadcasting to all agents

ponent delegated a function to another component coordinating with it, if the former had the function, this function became less-developed and the latter's function became well-developed like differentiation processes in cells. It could also initialize and restart differentiated software components, when some components could not be delegated like regeneration processes in lizards. It was constructed as a general-purpose and practical middleware system for software components on real distributed systems consisting of embedded computers or sensor nodes.

References

1. O. Babaoglu and H. Meling and A. Montresor, Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems, Proceeding of 22th IEEE International Conference on Distributed Computing Systems, July 2002.
2. E. Bonabeau, M. Dorigo, and G. Theraulaz: Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, 1999.
3. M. Dorigo and T. Stutzle: Ant Colony Optimization, MIT Press, 2004.
4. W. Lin, G. Lin, and H. Wei: Dynamic Auction Mechanism for Cloud Resource Allocation In Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'2010), pp.591–592, 2010.
5. A. Montresor, H. Meling, and O. Babaoglu, Messor: Load-Balancing through a Swarm of Autonomous Agents, Proceedings of International Workshop on Agents and Peer-to-Peer Computing, July 2002.

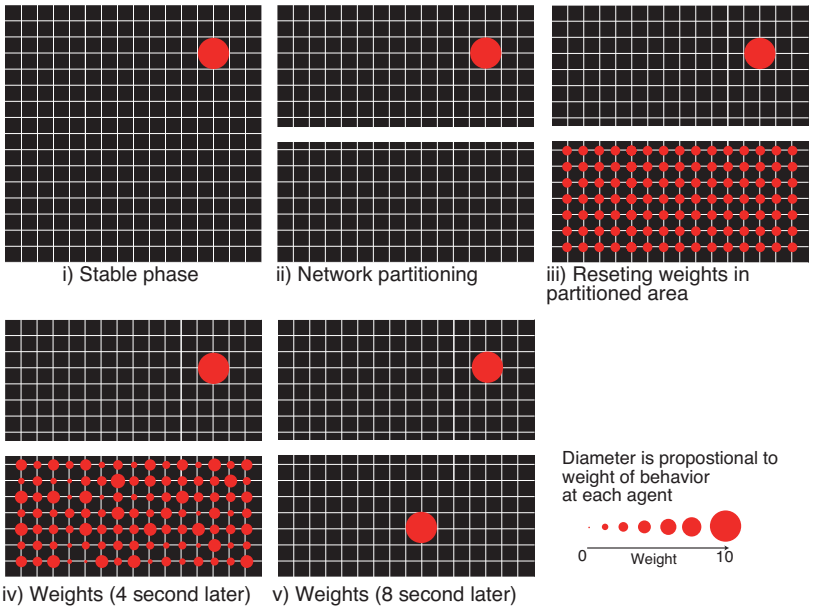


Fig. 8. Agents are differentiated in network partitioning

6. A. Montresor and O. Babaoglu, Biology-Inspired Approaches to Peer-to-Peer Computing in BISON Proceedings of International Conference on Intelligent System Design and Applications, Oklahoma, August 2003.
7. T. Nakano and T. Suda: Self-Organizing Network Services With Evolutionary Adaptation, IEEE Transactions on Neural Networks, vol.16, no.5, pp.1269-1278, 2005.
8. I. Satoh, Self-organizing Software Components in Distributed Systems, to appear in 20th International Conference on Architecture of Computing Systems (ARCS'2007), Lecture Notes in Computer Science, Vol.4415, pp.185-198, Springer, March 2007.
9. I. Satoh: Resilient Architecture for Complex Computing Systems. In Proceedings of 18th International Conference on Engineering of Complex Computer Systems, pp.256-259, IEEE Computer Society, 2013.
10. T. Suda and J. Suzuki: A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications. IEEE Journal on Selected Areas in Communications, vol.23, no.2, pp.249-260, 2005.

Appendix

This appendix we describe our model for regenerating software components, called agents, by using a differentiation mechanism in detail. We specify from 1-th to n-th behaviors of k-th agent, as $b_1^k, \ldots, b_n^k$ and the weight of behavior b_i^k as w_i^k. Each agent (k-th) assigns its own maximum to the total of the weights of all its behaviors. The W_i^k is the maximum of the weight of behavior b_i^k. The maximum total of the weights of its behaviors in the k-th agent must be less than W^k. ($W^k \geq \sum_{i=1}^{n} w_i^k$), where $w_j^k - 1$

is 0 if w_j^k is 0. The W^k may depend on agents. In fact, W^k corresponds to the upper limit of the ability of each agent and may depend on the performance of the underlying system, including the processor.

Invocation of behviors

1: When an agent (k-th agent) receives a request message from another agent, it selects the behavior (b_i^k) that can handle the message from its behavior part and dispatches the message to the selected behavior (Figure 1 (a)).
2: It executes the behavior (b_i^k) and returns the result.
3: It increases the weight of the behavior, w_i^k.
4: It multicasts a restraining message with the signature of the behavior, its identifier (k), and the behavior's weight (w_i^k) to other agents (Figure 1 (b)). [2]

The key idea behind this approach is to distinguish between internal and external requests. When behaviors are invoked by their agents, their weights are not increased. If the total weights of the agent's behaviors, $\sum w_i^k$, is equal to their maximal total weight W^k, it decreases one of the minimal (and positive) weights (w_j^k is replaced by $w_j^k - 1$ where $w_j^k = \mathbf{min}(w_1^k, \ldots, w_n^k)$ and $w_j^k \geq 0$). The above phase corresponds to the degeneration of agents.

Well/Less developing bahviors

1: When an agent (k-th agent) wants to execute a behavior, b_i , it looks up the weight (w_i^k) of the same or compatible behavior and the weights ($w_i^j, \ldots, w_i^m$) of such behaviors ($b_i^j, \ldots, b_i^m$).
2: If multiple agents, including itself, can provide the wanted behavior, it selects one of the agents according to selection function ϕ^k, which maps from w_i^k and $w_i^j, \ldots, w_i^m$ to b_i^l, where l is k or $j, \ldots, m$.
3: It delegates the selected agent to execute the behavior and waits for the result from the agent.

The approach permits agents to use their own evaluation functions, ϕ, because the selection of behaviors often depends on their applications. Although there is no universal selection function for mapping from behaviors' weights to at most one appropriate behavior like a variety of creatures, we can provide several functions.

Removing redundant behaviors

1: When an agent (k-th agent) receives a restraining message with regard to b_i^j from another agent (j-th) , it looks for the behaviors ($b_m^k, \ldots b_l^k$) that can satisfy the signature specified in the receiving message.
2: If it has such behaviors, it decreases their weights ($w_m^k, \ldots w_l^k$) and updates the weight (w_i^j) (Figure 1 (c)).
3: If the weights ($w_m^k, \ldots, w_l^k$) are under a specified value, e.g., 0, the behaviors ($b_m^k, \ldots b_l^k$) are inactivated.

[2] Restraining messages correspond to cAMP in differentiation.

Computational Intelligence for Demand Side Management and Demand Response Programs in Smart Grids

Taha Abdelhalim Nakabi, Keijo Haataja, and Pekka Toivanen

University of Eastern Finland, School of Computing, Kuopio Campus, P.O. Box 1627, 70211 Kuopio, Finland
tahanak@uef.fi, Keijo.Haataja@uef.fi, Pekka.Toivanen@uef.fi

***Abstract*—The reliability of a power system relies more on its flexibility, which is no longer easy to achieve on the production side because of the large integration of renewable energy sources. With the advances in smart grids, demand side management can achieve this flexibility by making customers participate in demand-response programs. In this paper, we mainly discuss a variety of techniques mostly relying on computational methods to implement efficient demand response programs. The discussed works used several techniques from the fields of artificial intelligence and optimization algorithms to handle problems like optimal power flow, optimal pricing, optimal device scheduling, economic dispatch, and customer behavior learning. In this review, we show that demand response programs that are based on computational methods can solve a variety of problems and can open many opportunities for electricity companies to make profit of the energy market. As an example of a potential opportunity to implement a demand response program, we propose a customer behavior learning model based on deep neural networks to learn the consumption patterns of a customer in response to a set of electricity prices. We also propose a profit maximization approach as an application of the learning model. The optimization is based on Genetic algorithms. The models can used in a variety of cases including reducing energy cost for customers as well as maximizing the retailers profit.**

Keywords— artificial neural networks; computational intelligence; machine learning; optimization algorithms; smart grid.

I. INTRODUCTION

The concept of Smart grid have been first mentioned in [1] as a response to the challenges that encountered the North American power grid. Since then, several papers have been published in the field, including solutions to integrate distributed renewable energy generation, energy storage and demand response optimization. There are many challenges to implement a smart grid that can make extensive use of renewable energy and maintain its reliability. Demand-side management is a key solution to tackle these challenges by shifting the flexibility of the power system to the customer side. In this paper, we will discuss the features related to demand-side management and more specifically demand response (DR) programs.

DR programs can be divided into two categories: price-based DR and incentive-based DR [2]. In price-based DR the customer responds to the varying prices during the day or price signals in peak hours to reduce his consumption or shift his loads from high price periods to lower price periods. In incentive-based DR the customer is given incentives to reduce his consumption in response to a signal about energy shortage in a certain period. In this paper, we focus more on price-based DR as it enables a good energy management throughout the day and not just in peak hours. A price-based DR can rely on various pricing schemes, such as time-of-used (TOU), day- ahead pricing (DAP), or real-time pricing (RTP). Each of these schemes have its own properties, advantages, and challenges.

In time-of-use pricing the utility company provides the customer with prices depending on the period of consumption throughout the day. The price during the peak hour is high, and it is low during the valley period. DAP is almost the same as TOU, but the prices are announced for one day 24 hours before. The DAP scheme relies on short-term prediction of energy consumption in the next 24 hours to offer the optimal prices. The RTP pricing scheme is different as it relies on the real time (or up to 1 hour ahead) energy consumption. The prices depend on supply and demand laws and therefore will be dynamic and difficult to predict.

In this paper, we review the major works in demand side management and DR programs; we focus on the computational aspect of the solutions, and discuss several methods for optimization and learning systems. Moreover, we will propose a novel research idea to solve the problem of optimal pricing and profit maximization. We combine two computational methods; the first is a double learning model that will be trained to make predictions of the hourly energy consumption of each customer given the day ahead hourly prices, assuming that the customers are price sensitive. Then a genetic algorithm will be used to find the best prices to propose to customers in order to maximize the retailer profit.

The rest of this paper is organized as follows: Section II reviews and classifies several works dealing with DSM and DR programs. Section III proposes a research idea for optimal pricing and customer behavior learning. Finally, Section IV concludes the paper and sketches some new research work ideas.

II. A LITERATURE REVIEW

A. Optimal power flow and DSM

The optimal power flow (OPF) problem consists of finding an optimal operational point of a power system that minimizes an appropriate cost function, such as generation cost or transmission loss subject to certain constraints on power and voltage variables [3]. OPF started receiving big interest since 1962 by Carpentier in [4]. Since then several methods and algorithms in the literature have approached this highly non- convex problem of OPF [5–7], including nonlinear programming [8], Newton-Raphson [9], quadratic programming [10], Lagrange relaxation [11], and interior point methods [12]. Recently, better results have been achieved by approaching the problem from the perspective of global optimization methods, such as genetic algorithm [13– 15], evolutionary programming [16–17], particle swarm optimization [18], simulated annealing [19], and tabu search [20]. In [21], an EMS was implemented together with a DC-OPF. Differential evolution was used to solve the minimization problem of the total generation cost. However, these methods present a heavy computation load and require high computational capabilities, which make their integration in EMS a complex task [22]. To overcome this limitation, the authors in [22] propose a DSM based on the combination of a fully connected neural network

(FCN) and OPF. They proposed a learning architecture to learn the input/outputs and reproduce the behavior of an OPF that generates numerical data based on genetic algorithm and fuzzy logic. The benefit was the reduction of computational time necessary to find the optimal schedule of generation and price-responsive loads.

B. DR in Day-Ahead Market

The main objective of a demand response program is to enable the users to participate in the electric system stability by reducing their loads in response to power grid needs and economic signals. As mentioned in Section I, this can be achieved by incentive based or price based DR. The day-ahead pricing scheme was addressed in the literature using various methods. In [23], authors presented a multi-objective optimization for air-conditioning day-ahead control to minimize the electricity expenses and expected error for the desired indoor temperature while retaining the degree of comfort in a room. The optimization problem was solved by an Immune Clonal Selection algorithm. In [24], authors presented an algorithm for the utility company to determine the optimal day ahead price values for customers who participate in a DR program. The demand function is considered as an unknown parameter. The problem was approached by a learning algorithm based on stochastic approximation. The behavior of a day-ahead electricity market was studied in [25] where a hierarchical multi-agent framework was presented for Day-Ahead planning and decision-making in retail electrical energy markets. On the retailer's side, agents are using machine learning techniques to model their environment and optimize their behavior to deal with interoperability and decision-making under incomplete information in a system that maintains the data privacy of the customers. On the customer's side air conditioning devices are controlled by agents that employ Q-learning algorithms [26] to optimize their consumption pattern. It is shown that this approach will reduce overall power consumption cost, maximize retailer's profit, and reduce peak load.

C. DR and Real-Time Pricing

In a demand response program, users are required to schedule their energy consumption according to meet the energy availability and reduce consumption peaks. The energy retailer provides the customers with a pricing scheme that reflects the energy cost and availability during the day. A comparison between real-time pricing and time-of-use pricing schemes in [27] showed that real-time pricing is more beneficial for both customers and retailers, because it enables the power system to flatten the load profile by providing enough information on the true time-variant electricity supply costs and enough financial incentives to end customers to adjust their energy consumptions. However, implementing a real-time price is a complex task, because the user may not know his energy prices and demand ahead of time. It is even more challenging because the users' scheduling decisions are coupled, since the demand of a user affects the price that is charged to all users. To tackle these challenges, a DR management for real-time price should be implemented to assist customers automatically by determining the optimal operations of appliances. In [28], an automatic residential energy consumption scheduling framework was designed. It attempts to achieve a desired trade-off between minimizing the electricity payment and minimizing the waiting time for the operation of each appliance in household in presence of a real-time pricing tariff combined with inclining block rates. The model was based on simple linear programming and it requires a minimum user effort. In [29] an optimization model for individual customers was developed to adjust the customer's decisions in response to time varying electricity prices. Robust optimization approaches were employed to model the price uncertainty in order to avoid considerable distortion to the optimal solution that can occur due to data uncertainty. In [30], authors proposed a real-time DR management model using scenario-based stochastic optimization and robust optimization approaches. The model can be embedded into smart meters to automatically determine the optimal operation in the next 5- minute time interval while considering future electricity price uncertainties.

D. DR for User's Cost Minimization

The previous works focus only on users trying to minimize their cost in short period of time and did not mention how the proposed scheduling algorithms can be used to minimize the long-term cost. Alternatively, [31–34] tackle this issue by introducing techniques for long-term cost minimization. In [31] a reinforcement learning approach was proposed to minimize a bill payment and a discomfort cost function for a household user. The model deals with stochastic demand arrivals and make a demand scheduling based on the devices allowable delays. A decentralized based heuristic method was implemented as well as an approximation approach based on Q-learning. A comparison of the results shows that each approach has advantages over the other under different scenarios. Q-learning approach works under more general settings, while the heuristic approach is better at delivering solutions in a much faster manner for regular sized problems. In [32] a reinforcement learning was proposed to automatically schedule the appliances in a household. The system schedules the time of operation by calculating the customer's trade-offs between delay and energy prices. It learns these trade-offs by observing energy customers' behavior and observing the patterns of energy pricing. Over time, the EMS learns how to make the best decisions for energy customers in the sense that it balances energy cost and the delay in energy usage in the same way that the customer would do. With the same technique [33] introduced a dynamic pricing algorithm based on reinforcement learning for the service provider to minimize the system cost. The model was then extended by adopting multi-agent learning structure where each customer can decide its energy consumption scheduling based on the observed retail price aiming at minimizing its expected cost. The problem was formulated as a dynamic pricing problem in the framework of a Markov-Decision Process (MDP). This problem was solved using Q-learning algorithm with two improvements: alternative state definition and virtual experience [34]. Moreover, the algorithm is fast and dynamic and does not require information about the system uncertainties. In [35], authors proposed a batch reinforcement-learning algorithm for residential demand response to schedule controllable loads, such as water heater and heat- pump thermostat. The authors considered the situation when a forecast of the exogenous data is provided and proposed a policy adjustment method that exploits general expert knowledge. A model-free Monte-Carlo method was introduced to find a day-

ahead consumption plan.

E. DR for Multiple Users

However, these works only deal with one user without considering the decision of multiple users trying to optimize their long-term costs. These interactions between users was captured by the authors in [36] where they studied the problem of electricity sharing among multiple foresighted users who participate in a demand response program with real-time pricing scheme. They consider an energy storage system shared across a group of homes, to store energy at lower price and sell it back to the grid when the price is high. They focus on the centralized control of such a shared battery. The scheduling problem of each individual user was formulated as a Markov decision process to study the optimal cost savings region for a finite capacity battery assuming a zero tolerance for activity delay. The price dependence on real-time demand and renewable generation was modeled in [37] by a non-cooperative game of incomplete information among the users with heterogeneous, but correlated consumption preferences. The Bayesian Nash Equilibria (BNE) in these games was considered as the optimal user behavior. Given this anticipating behavior two pricing policies were proposed: the first aims to achieve a target return rate and the second aims to minimize consumption peak-to-average ratio (PAR). In [38] a load scheduling learning algorithm was designed for users who schedule their appliances in response to RTP information. The study of long-term interactions among foresighted users enables the authors to model the users' decision making with uncertainty about the price and load demand as a Markov decision process with different states for different possible scenarios. The distributed load scheduling learning algorithm was based on actor-critic method [39–40] and converges to the Markov Perfect Equilibrium policy. The algorithm is online and model free, which enables the system to learn from the consequences of past decisions.

F. Economic Dispatch and DR

While DR focus on the customer side and his reaction to prices and incentives to reduce load, economic dispatch (ED) focus on the supply to minimize the power generation cost subject to some constraints. Combination of demand response and dynamic economic dispatch is of a big interest, because it does not just schedule the generation units optimally, but also the prices will be optimized at the same time, allowing a win-win strategy and increasing the reliability of the power system [27]. This concept was presented in [2, 41–42]. In [41] the combined model of ED and DR has been studied through a model based on price signals sent to the customers at the peak hour to reduce their loads and shift their consumption to the off-peak hours, in exchange of incentives paid to them for their participation in the EMS. The method used to determine the optimal price signal at peak hour is based on genetic optimization algorithm with a stochastic variable that estimates the probability of the customer's participation. However, this model is limited because it only considers the peak hours and not the whole day. A new solution method for solving the combined problem was presented in [42]. The authors integrated DEED (Dynamic Economic Emission Dispatch) and game theory based DR under a deregulated environment. This method is able to give feedback and update inaccurate solutions. The results show that this model is superior to independent optimization of DR or DEED. In [43] the integrated model of DED and EDRP (Emergency Demand Response Program) was presented with non-linear responsive load models. The optimization problem of fuel cost and collaboration incentives was solved using RDPSO (Random Drift Particle Swarm Optimization). This model was lately rebuilt in [2] to make a common price based DRP, namely TOU, instead of incentives. The objective is to minimize generation cost and determine the optimal prices during different periods simultaneously. The total cost was obtained by 4 meta-heuristic algorithms namely PSO, GA, ABC, and BCO, but ICA (Imperialist Colony Algorithm) was eventually adopted, because it gave better results.

G. Customer Behavior Learning and DR

In demand response, the information about customer energy consumption is extremely valuable for optimization and pricing. The reaction of customers to the electricity price can enable the retailer to extract valuable insights about customer behavior by recognizing different consumption patterns. In [44], authors proposed algorithms for learning the future price elasticity of customers based on their responses to previous pricing updates. The task was formulated as a linear regression problem and consider the aggregated changes in consumption over the distribution network as a weighted sum of all individual changes in consumption. In [45], authors studied the customer price elasticity of demand using an agent-based model. The model was used to demonstrate and quantify the economic impact of price elasticity of demand in electricity markets. In [46] the problem of learning customers' behavior was approached by short term load forecasting in a pricesensitive environment with real-time pricing scheme, where customers are reacting to price signals. The model used a neuro-fuzzy approach by building a two-stage forecaster consisting of an artificial neural network load forecaster followed by a fuzzy logic system. All these models are built for aggregated customers whereas [47] presented a model for individual customers. This is important, because it can identify valuable information about different behaviors and usage patterns between different customers in response to the price and temperature signals. The proposed model is based on probabilistic Bayesian behavior model to learn the energy usage patterns of shiftable appliances and a price-demand model to predict the hourly energy consumption of curtailable appliances. The authors also proposed a distributed pricing optimization based on genetic algorithm for the utility company to maximize its profit based on the learning results. The reviewed works tackle different types of problems in demand side management using various computation methods and approaches. Table 1 classifies these works in addition to other works according to the issue and computation method used.

III. OUR NOVEL RESEARCH PROPOSAL

The previously discussed works have approached the problem of DSM and specifically the demand response problem by valuable approaches. Most of works that tackle the issue of device scheduling assume that the household is equipped with a Home Energy Management System (HEMS) that automate the task of turning on the appliances at lower price periods and turning them off at high price periods. Despite these works have achieved valuable results, the requirements to implement it in a large scale is too expensive for most practical situations.

Table 1: CLASSIFICATION OF DIFFERENT WORKS AND TECHNIQUES

	OPF	Optimal device scheduling	Optimal pricing	DR and ED combination	Customer behavior learning
Linear programming		[28]		[41]	
evolutionary optimization	[13], [14], [15], [16],	[53], [2]	[47]	[2]	
Particle swarm optimization	[18]	[2], [54]		[43], [2]	
Neural networks	[22]				[46]
Fuzzy logic	[22]				[46]
Immune Clonal selection		[23]			
Stochastic programming		[30]	[24]		
Multi-agent system		[33]	[25],[34]		[45]
Machine learning			[25]		[55], [56]
Reinforcement learning		[25], [31], [32], [33],			
Robust optimization		[29, [30]			
Game theory		[36], [37]		[42]	
Bayesian model					[47], [56], [52]

Alternatively, smart meters infrastructure is already deployed in a large scale around the world and still expanding. In 2015, U.S. electric utilities had about 64.7 million advanced (smart) metering infrastructure (AMI) installations [48]. By 2020, it is expected that almost 72% of European customers will have a smart meter for electricity [49]. It is expected that in the U.K., all households will have smart meters installed by 2020 according to [50]. Therefore, it is important to develop methods and techniques for demand side management to implement demand response programs that take advantage of this source of information by using the real-time data provided by the two-way communication channels in smart grid. The approach proposed in [47], presented a valuable method in this matter, it tackles the problem of learning the customer behavior and maximizing the retailer's profit and the total energy cost according to the learning model, without increasing the electricity bill and without using any type of HEMS. The whole approach is based on the smart meters infrastructure and day ahead pricing.

Our idea is to tackle almost the same problem with different methods and with different assumptions. In [47], the pattern of energy usage learning approach was based on device by device consumption data extracted using a nonintrusive appliance load monitoring (NILM) to extract hourly consumption and on/off time information of each appliance from the entire electrical power consumption of each house by using signal analysis algorithms [51]. The appliances were then classified into shiftable and curtailable devices. In each shiftable device, the learning is based on Bayesian probabilistic model, whereas curtailable devices was modeled by linear regression including historical data of hourly prices and temperatures. However, the learning approach of shiftable devices considers just the ranking of the cheapest periods without considering the period itself. This information is extremely valuable because without it we cannot know if the user is price sensitive or is just using a device in an exact time every day and it coincides with the cheapest period. In our novel proposal, we will handle the aggregate shiftable devices instead of learning the consumption of each device. This will reduce the computational requirement without affecting the accuracy of the model. We will implement and test 2 learning models; the first one will learn the consumption patterns from shiftable appliances during 24 hours. The second model will learn the response of curtailable devices, namely air conditioning systems, to an electricity price and a temperature corresponding to a time slot of one hour. The model used for learning shiftable appliances consumption patterns takes 24 inputs representing the day ahead hourly prices, and 24 outputs representing the hourly loads from shiftable appliances. This architecture is justified by the fact that the customer is shifting his appliances according to the whole set of prices over one day either using a scheduling system or manually. In the second learning model, the inputs are the electricity price and temperature. This is justified by the assumption that air-conditioning systems' loads depend only on the electricity price and temperature at a certain time slot. The learning approach in the first model will use deep NNs. This choice is justified by the ability of these models to learn the mapping between an input and output vectors with high dimensions and complexity. In our case, we have an input and an output with 24 entries, which is a complex and high dimensional learning problem. Fig.1 illustrate this architecture. In the second learning model, we will implement different regression algorithms, such as linear regression used in [47], support vector machines regression, along with deep NNs, then evaluate and compare the results in order to choose the best model for this problem. The two models will be trained on historical data, and updated every week with the new consumption data to track consumption changes. After implementing the learning model, we will handle the

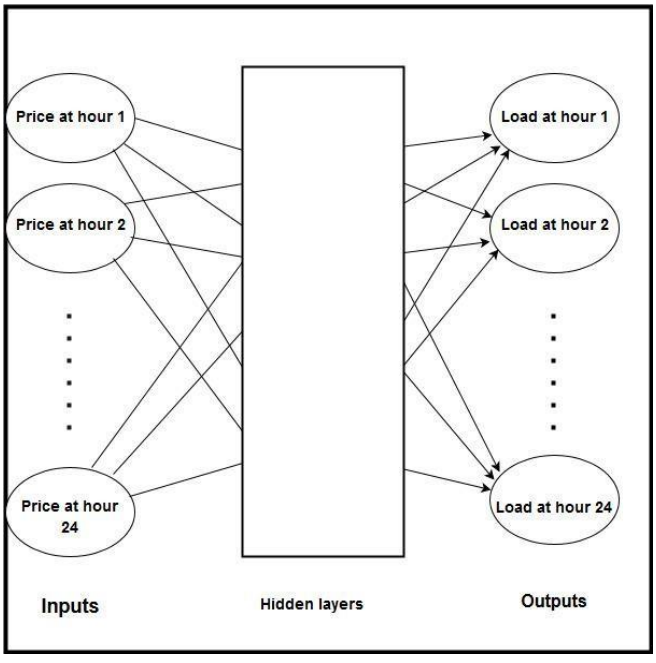


Fig.1: Neural network architecture for shiftable devices

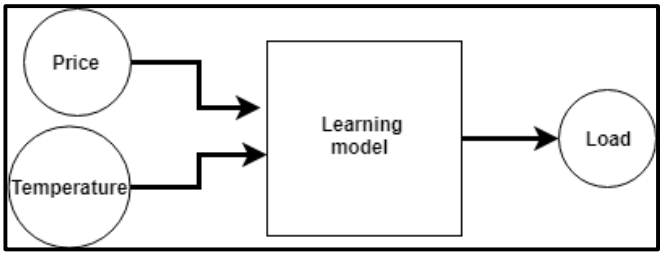


Fig.2: Learning model for curtailable devices

optimization problem to maximize the retailer's profit without increasing the bill payment. We will evaluate 5 different meta-heuristics for combinatorial optimization namely: PSO, GA, ACO, ABC, and ICA. Then we make a comparison of results using real data from a local electricity retail company.

Using the NILM we separate the data of shiftable and curtailable devices. We train our models using these sets of data and then we use the trained models to predict the day ahead hourly energy consumption given the day ahead prices and temperatures. This process will be executed each week using the most recent consumption data.

Based on these models, an optimization algorithm will be implemented to maximize the retailer's profit. The aim of this algorithm is to find the prices that will maximize the difference between the energy cost and the benefit from all customers considering economic and political constraints about the amount of energy provided and the bill payment. We show here a GA based optimization algorithm using results from the learning models to find the optimal pricing strategy. Figure 3 shows the process of finding the optimal pricing strategy. First, a population of PN strategies is initialized randomly; each strategy is represented by a 24-dimensional vector of hourly prices. Each prices vector is used as an input in the neural networks for shiftable and curtailable appliances. 24h temperature forecast for the day ahead is also used for curtailable appliances model. The energy consumption for each customer will be predicted then the bill payment will be calculated. The benefit will be calculated by summing the predicted bills of all customers in response to the strategy. The cost of the total energy consumption will be calculated. We calculate the predicted profit, which is the difference between benefit and cost. We check the constraints satisfaction and obtain the fitness function. We iterate this process for all the strategies. Then generate a new population of strategies by using the selection, crossover, and mutation. We repeat this process until a stop criterion, i.e., the situation when the solution is not improving anymore. At the end, we obtain an optimal strategy that the retailer will announce via smart meters

IV. CONCLUSION AND FUTURE WORK

Demand response programs have a big potential in increasing the power system's flexibility and reliability by involving the customer in the decision process. Several methods from the literature for optimization and automation were discussed in this paper. Computational methods from the field of artificial intelligence and optimization, were used to solve the problems of OPF, optimal pricing, optimal device scheduling, DR with economic dispatch, and customer behavior learning. Our novel proposal solves the combined problem of customer behavior learning and optimal pricing. Two learning model architectures were proposed for customer behavior learning as well as a genetic algorithm for price optimization. The future work will bring more details and numerical results about the proposed idea.

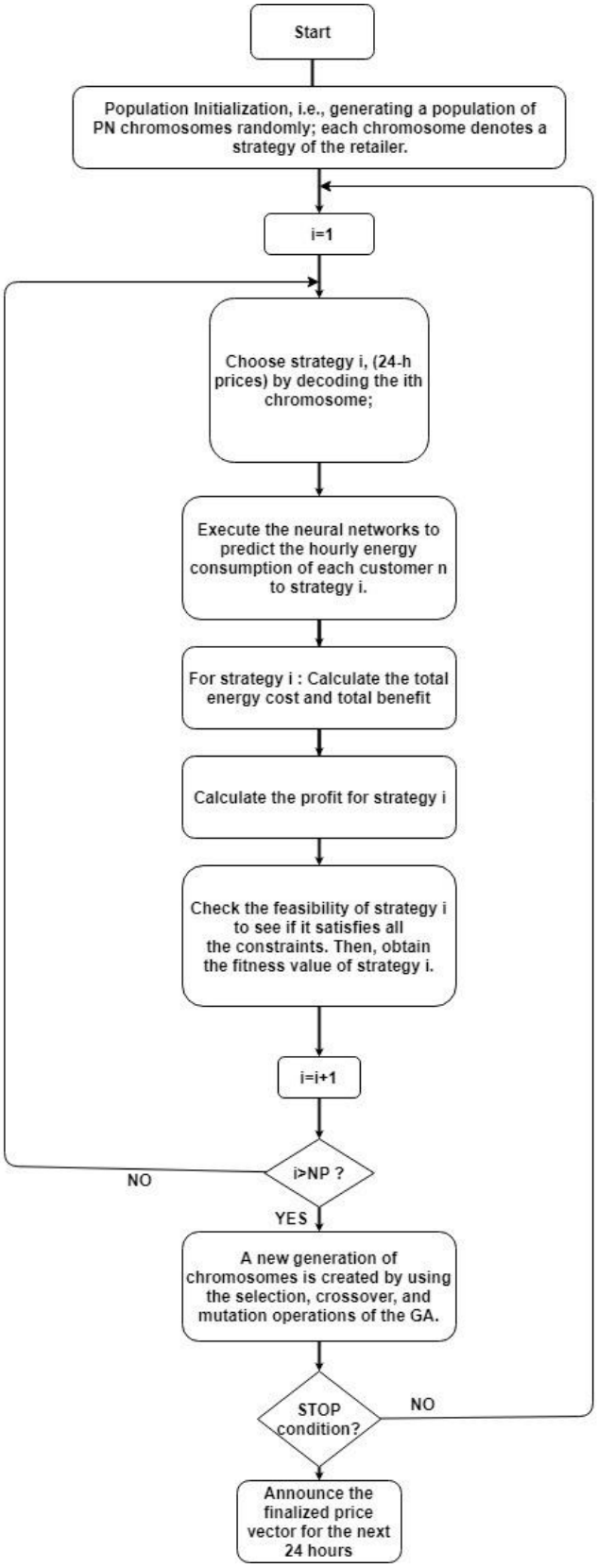


Fig.3: GA-based Optimization algorithm for retailer's profit maximization

Advances in artificial intelligence and machine learning are deeply influencing our life in many aspects. AI takes advantage

of the large availability of data nowadays and the big computational power. Therefore, this huge power of AI and ML are likely to be the next revolution in the most vital aspect of the modern life: *energy*. Smart grids are well appropriate and much convenient to implement this idea of integrating AI and ML in the energy management process. Thanks to the two-ways communication infrastructure and the large data collected via smart meters. AI and ML techniques have a large potential in taking advantage of this data and infrastructure to tackle the challenges of the new smart grid allowing more integration of renewable energies, reducing carbon emission, saving generation cost, integrating electric vehicles, increasing network reliability, and offering better products to the final customers at lower prices. To achieve these expectations, several researches are focusing on different aspects of the smart grid. The challenge in the future will be to implement a scalable system that can englobe different computation techniques and general AI to handle different aspects like demand side management, Electric vehicles, virtual power plants, energy prosumers, and self- healing networks.

REFERENCES:

[1]. S. Massoud Amin, B.F. Wollenberg, "Toward a smart grid: power delivery for the 21st century", IEEE Power and Energy Magazine, vol. 3, pp. 34-41, Sept.-Oct. 2005

[2]. Ehsan Dehnavi, Hamdi Abdi, "Optimal pricing in time of use demand response by integrating with dynamic economic dispatch problem", Energy, 109 pp. 1086-1094, 2016.

[3]. J. A. Momoh, "Electric Power System Applications of Optimization", New York: Markel Dekker, 2001.

[4]. J. Carpentier, "Contribution to the economic dispatch problem," Bull. Soc. Francaise Elect., vol. 3, no. 8, pp. 431–447, 1962

[5]. M. Huneault and F. D. Galiana, "A survey of the optimal power flow literature,"IEEE Trans. Power Syst, vol. 6, no. 2, pp. 762–770, May 1991.

[6]. L. Torres and V. H. Quintana, "Optimal power flow by a nonlinear complementarity method," IEEE Trans. Power Syst, vol. 15, no. 3, pp. 1028–1033, Aug. 2000.

[7]. H. Wang, C. E. Murillo-Sanchez, R. D. Zimmerman, and R. J. Thomas, "On computational issues of market-based optimal power flow,"IEEE Trans. Power Syst, vol. 22, no. 3, pp. 1185–1193, Aug. 2007.

[8]. M.Santos-Neitoand V.H. Quintana. "Linear Reactive Power Studies for Longitudinal Power Systems," 9th PSCC Conference, pp 783-787, 1987.

[9]. Rong-Mow Jan ; Nanming Chen. "Application of the fast Newton-Raphson economic dispatch and reactive power/voltage dispatch by sensitivity factors to optimal power flow," IEEE Trans. on Energy Conversion, vol. 10, pp 293 - 301, 1995.

[10]. A.D. Papalexopoulos, C.F. Imparato, and F. F. Wu, "Large Scale Optimal Power Flow: Effects of Initialization Decoupling and Discretization,"IEEE Trans.on Power Systems, vol. PWRS-4, No.2, pp 748-739, May 1989

[11]. K.H. Abdul-Rahman, S.M. Shahidehpour, M. Aganagic S. Mokhtari, "A practical resource scheduling with OPF constraints", IEEE Trans.on Power Systems, Vol. 11, No.1, pp 254-259, February 1996.

[12]. L. Torres, V. H. Quintana, "An interior-point method for nonlinear optimal power flow using voltage rectangular coordinate" IEEE Trans.on Power Systems, Vol. 13, No. 4, pp. 1211-1218, November 1998,

[13]. L. L. Lai, J. T. Ma, R. Yokoyama, and M. Zhao, "Improved genetic algorithms for optimal power flow under both normal and contingent operation states," J. l. Pwr. En. Syst., vol. 19, no. 5, pp. 287–292, 1997.

[14]. A.G. Bakirtzis, P. N. Biskas, C. E. Zoumas, and V. Petridis, "Optimal power flow by enhanced genetic algorithm," IEEE Trans. Pwr. Syst., vol. 17, pp. 229–236, 2002.

[15]. M.S. Kumari, S. Maheswarapu, "Enhanced Genetic Algorithm based computation technique for multi-objective Optimal Power Flow solution", Electrical Power and Energy Systems, vol 32, No. 6, pp. 736–742, July 2010.

[16]. Yuryevich , Kit Po Wong, "Evolutionary programming based optimal power flow algorithm", IEEE Trans.on Power Systems , Vol: 14, Issue: 4 pp. 1245 – 1250 , Nov 1999.

[17]. W. Ongsakul, T. Tantimaporn, "Optimal Power Flow by Improved Evolutionary Programming", Electric Power Components and Systems, vol. 34, pp. 79–95, 2006.

[18]. M.A. Abido, "Optimal power flow using particle swarm optimization", International Journal of Electrical Power & Energy Systems, Vol 24, Issue 7, pp. 563–571, October 2002.

[19]. L. Chen, H. Suzuki, and K. Katou, "Mean-field theory for optimal power flow," IEEE Trans. Pwr. Syst., vol. 12, pp. 1481–1486, 1997.

[20]. T. Kulworawanichpong and S. Sujitjorn, "Optimal power flow using tabu search," IEEE Pwr. Engr. Rev., pp. 37– 40, 2002.

[21]. P. P. Verma, P. Soumya, K. S. Swarup, "Optimal Day- Ahead Scheduling in Smart Grid with Demand Side Management", 2016 IEEE 6th International Conference on Power Systems (ICPS), March 2016

[22]. P. Siano, C. Cecati, H. Yu, and J. Kolbusz, "Real time operation of smart grids via FCN networks and optimal power flow," IEEE Trans. Ind. Informat., vol. 8, no. 4, pp. 944 –952, Nov. 2012.

[23]. Y. Y. Hong, J. K. Lin, C. P. Wu, and C. C. Chuang, "Multi-objective air conditioning control considering fuzzy parameters using immune clonal selection programming," IEEE Trans. Smart Grid, vol. 3, no. 4, pp. 1603–1610, Dec. 2012.

[24]. L. Jia, Q. Zhao, and L. Tong, "Retail pricing for stochastic demand with unknown parameters: An online machine learning approach," in Proc. Of Allerton Conf.

on Communication, Control, and Computing, Monticello, IL, Oct. 2013.

[25]. K. Dehghanpour, M. Hashem Nehrir, John W. Sheppard, Nathan C. Kelly, "Agent-Based Modeling of Retail Electrical Energy Markets with Demand Response", Accepted for publication in IEEE Trans. on smart grid, NOV. 13TH, 2016, pp 1-1

[26]. Watkins, C.J.C.H. & Dayan, P. Mach Learn (1992) 8: 279. doi:10.1007/BF00992698

[27]. S. Borenstein, M. Jaske, and A. Rosenfeld, "Dynamic pricing, advanced metering and demand response in electricity markets," Center for the Study of Energy Markets, UC Berkeley, 2002.

[28]. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," IEEE Trans. Smart Grid, vol. 1, no. 2, pp. 120–133, Sep. 2010.

[29]. A.J.Conejo, J.M.Morales,.Baringo,"Real-time demand response model," IEEE Trans. Smart Grid, vol. 1, no. 3, pp. 236–242,Dec. 2010.

[30]. Z. Chen, L. Wu, and Y. Fu, "Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization," IEEE Trans. Smart Grid, vol. 3, no. 4, pp. 1822–1831, Dec. 2012

[31]. Y. Liang, L. He, X. Cao, and Z. J. Shen, "Stochastic control for smart grid users with flexible demand," IEEE Trans. on Smart Grid, vol. 4, no. 4, pp. 2296–2308, Dec. 2013

[32]. Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," IEEE Trans. on Smart Grid, vol. 6, no. 5, pp. 2312–2324, Sept. 2015.

[33]. B. Kim, Y. Zhang, M. van der Schaar, and J. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," IEEE Trans. on Smart Grid, vol. 7, no. 5, pp. 2187–2198, Sept. 2016

[34]. N. Mastronarde and M. van der Schaar, "Joint physical- layer and system level power management for delay- sensitive wireless communications," IEEE Trans. Mobile Comput., vol. 12, no. 4, pp. 694–709, Apr. 2013.

[35]. F. Ruelens, B. J. Claessens, S. Vandael, B. D. Schutter, R. Babuska, and R. Belmans, "Residential demand respo nse of thermostatically controlled loads using batch reinforcement learning," accepted for publication in IEEE Trans. on Smart Grid, 2016.

[36]. Yao, and P. Venkitasubramaniam, "Optimal end user energy storage sharing in demand response," in Proc. of IEEE Smart GridComm, Miami,FL, Nov. 2015.

[37]. C. Eksin, H. Delic, and A. Ribeiro, "Demand response management in smart grids with heterogeneous customer preferences," IEEE Trans. On Smart Grid, vol. 6, no. 6, pp. 3082–3094, Nov. 2015

[38]. S. Bahrami, V. W.S. Wong, and J. Huang, "An Online Learning Algorithm for Demand Response in Smart Grid," IEEE Trans. on Smart Grid (Volume: PP, Issue: 99), pp. 1-1, February 2017

[39]. Konda and J. Tsitsiklis, "On actor-critic algorithms," SIAM Journalon Control and Optimization, vol. 42, no. 4, pp. 1143–1166, Aug. 2003.

[40]. S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor-critic algorithms," Automatica, vol. 45, no. 11, pp. 2471–2482, Nov.2009

[41]. Ashfaq A, Yingyun S, Zia Khan A. Optimization of economic dispatch problem integrated with stochastic demand side response. In: IEEE International Conference on intelligent energy and power systems; 2014. pp. 116-121.

[42]. Nwulu NI, Xia X. Implementing a model predictive control strategy on the dynamic economic emission dispatch problem with game theory based demand response programs. Energy 2015; vol. 91: pp. 404-419.

[43]. Abdi H, Dehnavi E, Mohammadi F. "Dynamic economic dispatch problem integrated with demand response (DEDDR) considering non-linear responsive load models." IEEE Trans. Smart Grid 2015, issue 6, vol.7, pp. 2586 – 259.

[44]. Gómez, M. Chertkov, S. Backhaus, and H. J. Kappen, "Learning price-elasticity of smart customers in power distribution systems," in Proc. IEEE 3rd Int. Conf. Smart Grid Commun. (SmartGridComm), Tainan City, Taiwan, 2012, pp. 647–652.

[45]. P. R. Thimmapuram and J. Kim, "Customers' price elasticity of demand modeling with economic effects on electricity markets using an agent based model," IEEE Trans. Smart Grid, vol. 4, no. 1, pp. 390–397, Mar. 2013.

[46]. A. Khotanzad, E. Zhou, and H. Elragal, "A neuro-fuzzy approach to short-term load forecasting in a price-sensitive environment," IEEE Trans. Power Syst., vol. 17, no. 4, pp. 1273–1282, Nov. 2002.

[47]. Fan-Lin Meng, Xiao-Jun Zeng, "A Profit Maximization Approach to Demand Response Management with Customers Behavior Learning in Smart Grid," IEEE Trans. smart grid, VOL. 7, NO. 3, MAY 2016

[48]. The U.S. Energy Information Administration (EIA) https://www.eia.gov/tools/faqs/faq.php?id=108&t=3

[49]. European Commission Energy, https://ec.europa.eu/energy/en/topics/markets-and-customers/smart-grids-and-meters.

[50]. Department of Energy and Climate Change. (Mar. 2014). Helping Households to Cut Their Energy Bills. [Online]. Available: https://www.gov.uk/government/policies/helping-households-to-cut-theirenergy-bills/supporting-pages/smart-meters,

[51]. G. W. Hart, "Nonintrusive appliance load monitoring," Proc. IEEE, vol. 80, no. 12, pp. 1870–1891, Dec. 1992.

[52]. Mikhail V. Goubko, Sergey O. Kuznetsov. Alexey A. Neznanov. Dmitry I. Ignatov., "Bayesian Learning of Customer Preferences for Residential Demand

Response", Volume 49, Issue 32, 2016, Pages 24-29.

[53]. Khourya, R. Mbayedb, G. Salloumb, E. Monmassona, "Predictive demand side management of a residential house under intermittent primary energy source conditions", Volume 112, 15 January 2016, pp: 110–120

[54]. Sofana Reka S., Ramesh V. (2016) "Demand Side Response Modeling with Controller Design Using Aggregate Air Conditioning Loads and Particle Swarm Optimization". In: Choudhary R., Mandal J., Auluck N., Nagarajaram H. (eds) Advanced Computing and Communication Technologies. Advances in Intelligent Systems and Computing, vol 452. Springer, Singapore.

[55]. Dong Zhang, Shuhui Li, Min Sun, and Zheng O'Neill, "An Optimal and Learning-Based Demand Response and Home Energy Management System", IEEE Trans. on smart grid, VOL. 7, NO. 4, JULY 2016, pp: 1790-1801.

[56]. Heng Shi, Minghao Xu, Ran Li, "Deep Learning for Household Load Forecasting – A Novel Pooling Deep RNN," IEEE Trans. on Smart Grid, Volume: PP:1-1, Issue: 99, 2017.

Robust Multiobjective Portfolio Optimization Problem

Samira Bokhari[1][0000−0002−1710−0265] and Méziane Aïder[2][0000−0001−7195−810X]

LaROMaD, Fac. Maths, USTHB, PO 32, 16111 Bab Ezzouar, Algeria
sbokhari@usthb.dz
m-aider@usthb.dz

Abstract. Portfolio optimizations are widely studied because of their real-world applications in finance and banking. The Classical formulations of the portfolio optimization problem, such as mean-variance or Value-at-Risk approaches, can result in a portfolio extremely sensitive to errors in the data, such as mean and covariance matrix of the returns. In this paper we propose a new robust optimization model as a way to alleviate this problem in a tractable manner. We suppose that the distribution of returns and the covariance matrix are uncertain and bounded. We suggest robust formulations to take into account the worst case criterion to cope with the sensitivity of the optimal portfolio linked to uncertain market parameters. We treat the maximization of returns and the minimization of risks as two separate objectives. Multi-Objective Evolutionary Algorithms such as NSGA-II and SPEA2 are ideal for solving multi-objective portfolio problems. Finally we assess the performance of our robust formulations using the Hypervolume Performance Assessment Indicator (I_H).

Keywords: Multiobjective, Heuristics, Optimization, Robust Optimization, Multiobjective portfolio optimization, Worst-Case Criterion, NSGA-II, SPEA2, Hypervolume Performance Indicator

1 Literature overview and positioning

The first mathematical model for portfolio selection under uncertainty is attributed to Markowitz[18][19][20]. It looked at portfolio selection as an optimization problem in which an asset mix is chosen so that the portfolio variance is minimal for any given level of expected return, and simultaneously, the expected return is maximal for any given level of portfolio variance.A single linear return on a portfolio is measured by the expected value of the random portfolio return, and a single convex nonlinear risk is quantified by the variance of the portfolio return.

Despite the theoretical success of the mean-variance model [19], practitioners have shied away from this model. The following quote summarizes the problem: Although Markowitz efficiency is a convenient and useful theoretical framework for portfolio optimality, in practice it is an error-prone procedure that often

results in error-maximized and makes irrelevant investment portfolios [40]. This behavior is a reflection of the fact that solutions of optimization problems are often very sensitive to perturbations in the parameters of the problem; since the estimates of the market parameters are subject to statistical errors, the results of the subsequent optimization are not very reliable [46]. Various aspects of this phenomenon have been extensively studied in the literature about portfolio optimizations. Generally, the Markowitz model is criticized as less efficient with axiomatic models of preferences for choice under risk [7]. Levy affirmed that models with regard to the preferences are based on the relation of stochastic dominance or on the expected utility theory [16].For that reason, Ballestero and Romero, suggested maximizing the investor expected utility of returns over the efficient frontier [6].

Several techniques have been suggested to reduce the sensitivity of the Markowitz-optimal portfolios to input uncertainty: Chopra [46], Frost and Savarino [35] proposed constraining portfolio weights, Chopra and al[46] proposed using a James-Stein estimator for the means, while Klein and Bawa [41], Frost and Savarino [35], Black and Litterman [11] suggested Bayesian estimation of means and covariances. Although these techniques reduce the sensitivity of the portfolio composition to the parameter estimates, they are not able to provide any guarantees on the risk-return performance of the portfolio. Recently scenario-based stochastic programming models have also been proposed for handling the uncertainty in parameters. All approaches cited above do not provide any hard guarantees on the portfolio performance and become very inefficient as the number of assets grows. Although many computational techniques have been developed for this purpose, most of them are single objective approaches, even though this problem clearly consists of two conflicting objectives. However, this combinatorial optimization problem has a highly complex search space due to the abundant choices of available financial assets. Thus, portfolio optimization continues to pose a challenge for efficient optimization techniques. Although many computational techniques have been developed for this purpose, most of them are single objective approaches, even though this problem clearly consists of two conflicting objectives. However, there are an increasing number of multi-objective approaches being developed, particularly multi-objective evolutionary algorithms (MOEA)[33].

The main advantage of evolutionary multi-objective portfolio optimization [42], is that an estimation of the efficient risk-return frontier can be obtained in a single run as opposed to the multiple runs needed in the case of single objective approaches.

Robust optimization [15] means finding solutions to a given optimization problems with uncertain input parameters that will achieve good objective values for all, or most realizations of the uncertain input parameters [43]. The concept of robust systems was first introduced by Taguchi [15], who studied uncertainties which may occur during the design of a product, even if today we use methods that are different from the ones previously mentioned, the goal remains the same, it is to find a robust solution that is acceptable in many scenarios and that is never too bad [44]. Accordingly, it is argued that the solution is robust only if its value does not significantly change when the decision vector is slightly disturbed [7]. The least sensitive to disturbance parameters solution is the most robust [45]. Robust solutions are those that optimize the robustness measurement in every scenario. Eventually, define robustness as an ability to resist about or areas of ignorance in order to protect impacts judged regrettable. Several techniques have been suggested about the optimization under uncertainty. Uncertainty may appear either in the costs coefficients or in the constraints matrix or in the second member as well as it may appear in all of the model.To measure this uncertainty, many robustness criteria exist which among criteria found in the literature such as the worst-case and the maximum regret [34]. The maximum regret criterion was introduced by Savage [24] et al [37]. Several authors studied the application of this criterion. The first studies were those of Shimizu and Aiyoshi [29], the work of Mausser and Laguna [17] who used uncertain linear programs, and Averbakh [22]; we also cite the tree covering problem which was treated with Yaman et al [21][22], Montemanni et al [39] [38]. The complexity of this problem is given by Aron and Hentenryck [3], and an integer formulation is proposed by Yaman [21]. For the worst-case criterion, we give some references to some issues that caught our attention: we first mention the work of Yaman [21] who is interested in the tree covering problem. There is also the robust shortest path problem which is processed by Yu and Yang [14]. We finally mention the work of Yu et al [14][13] on the robust knapsack problem, and also the work of Gabrel and Murat [13] on the Robustness and duality in linear programming.

Structure and Contribution of this paper

In this paper we propose alternative deterministic models that are robust to parameters uncertainty and estimation errors [18] and how to formulate and solve the Multi-objective portfolio optimization problems under uncertainty [6] [10]. The perturbations in the market parameters are modeled as unknown, but bounded, and we take the pessimistic view of robustness and look for a solution that has the best performance under its worst case behavior of these perturbations.This robust optimization framework was introduced in Ben-Tal and Nemirovski [2] for linear programming and in Ben-Tal and Nemirovski (1998)

for general convex programming.To find the robust efficient frontier we are opting for the multi-objective genetic approach NSGA-II (Non-Dominated Sorting Genetic Algorithm) [27] [26] which has the advantage of simultaneously taking into account two crucial criteria for the optimization methods that are intensification and diversification, using as metric respectively elitism and the distance crowding. A comparative approach would be to choose a genetic evolutionary algorithm SPEA2 (Strength Pareto Evolutionary Algorithm) [9] which is based on the use of a record represented by an external force A_t of size $N_{archive}$. This fixed-size package is designed to contain a limited number of non-dominated solutions. The classification of individuals is based on the principle of dominance over the value of Strength $S(i) = |\{j : j \in P_t \cup A_t; i < j\}|$. To define which method NSGA-II or SPEA2 is more robust, we introduce the hypervolume performance indicator (I_H).

The structure of this paper is as follows: Section 2 recalls the background behind portfolio management and the robust optimization, section 3 looks at optimization methods such as multi-objective and genetic algorithms and multi-objective indicators, section 4 contains computational results on the performance of two of the suggested algorithms, in section 5 we conclude and give some future perspectives.

References

1. A. Ben-Tal and A. Nimerovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25 :1-13, 1999.
2. A. Ben-Tal and A. Nimerovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88 : 411-224, 2000.
3. D. Aron and P. Hentenryck. On the complexity of the robust spanning tree problem with interval data. *Operations Research Letters*, 40, 2004.
4. D. Bertsimas, B. Brown and C. Caramanis. Theory and Application of Robust Optimization. *SIAM Review*, 3: 464-501 ,2011.
5. D. E. Bell and H. Raiffa Risky choice revisited. Decision Making: Descriptive, Normative and Prescriptive Interactions. *Cambridge University Press, Cambridge*, 1988.
6. E. Ballestero and C. Romero. Portfolio selection: A compromise programming solution. *Journal of Operational Research Society*, 47, 1996.
7. E. Daniel and A. Salazar. Robustness analysis : An information-based perspective. *European Working Group : Multiple Criteria Decision Aiding*, 2006.
8. E-G. Talbi. Metaheuristics: from design to implementation. Wiley, 2009.
9. E. Zitzler, M. Laumanns and L.Thiele. SPEA2:Improving the Performance of the Strength Pareto evolutionary Algorithm, *Technical Report 103, Computer Engineering and Communication Networks lab(Tik), Swiss Federal Institute of Technology*, May 2001.
10. F. Ben Abdelaziz, P. Lang and R. Nadeau. Efficiency in multiple criteria under uncertainty. *Theory and Decision*, 1999.
11. F. Black and R. Litterman. Asset allocation: Combining investor views with market equilibrium *Technical report. Fixed Income Research, Goldman, New York*, 1990.
12. F. Oustry, H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM J. Optim*, 33-52 ,1998.

13. F. Taniguchi, T. Yamada and S. Kataoka. Heuristic and exact algorithms for the maxmin optimization of the multi-scenario knapsack problem. *Computers and Operations Research*, 2008.
14. G. Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 415, 1996.
15. H. G. Beyer and B. Sendhoff. Robust Optimization-A Comprehensive *survey.ELSEVIER*, 3190-3218, 2007.
16. H.Levy Stochastic dominance and expected utility: Survey and analysis. *Management Science*, 593, 1992.
17. H. Mausser and M. Laguna. Minimizing the maximum relative regret for linear programs with interval objective function coefficents. *European Journal of Operational Research*, 1070, Oct. 1999.
18. H. Markowitz, Portfolio Selection. *The Journal of Finance*, 7, March 1952.
19. H. Markowitz, Portfolio Selection: Efficient Diversification of Investments. *New York: John Wiley & Sons*, 1959.
20. H. Markowitz, Portfolio selection: efficient diversification of investments. *2nd ed, Cambridge, MA: Basil Blackwell*, 1991.
21. H. Yaman, O. Karasan and M. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 40, 2001.
22. I. Averbakh and V. Lebedev. On the complexity of minmax regret linear programming. *European Journal of Operational Research*
23. J. Dreo, A. Petrowski, P.Siarry and E.Taillard. Mtaheuristiques pour l'optimisation difficile. *Eyrolles*, February 10, 2005
24. J. Savage. The foundation of statistics. *New York : Wiley*, 1954.
25. J. D. Schaffer, Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms *Ph.D. dissertation, Vanderbilt University, Nashville, TN, USA*, 1984.
26. K. Deb. Multi-objective Optimization Using Evolutionary Algorithm. *KanGAL Report*, 1-24, February 10, 2012
27. K. Deb. A Fast and Elitiste Multiobjective Genetic Algorithm: NSGA-II. *Journal*, 182-197, April 2002.
28. K. Deb, S. Agrawal, A. Pratapand and T. Meyarivan A Fast and Elitist multiobjective genetic algorithm: NSGA-II. *Technical report, Indian Institute of Technology Kanpur*, 20001, 2000.
29. K. Shimizu and E. Aiyoshi. Necessary conditions for min-max problems and algorithm by relaxation procedure. *IEEE Transaction on Automatic control*, 66, 1980.
30. L. Bradstreet The Hypervolume Indicator for Multi-objective Optimization: Calculation and Use. *PhD Thesis, University of Western Australia*, April 2011.
31. L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 1035-1064, 1997.
32. L. Liu Approximate portfolio analysis. *European Journal of Operational Research*, 49, 1999.
33. M. G. Castillo Tapia and C. A. Coello Coello, Applications of multiobjective evolutionary algorithms in economics and finance *A survey in IEEE Congress on Evolutionary Computation*, 532-539, Sept 2007.
34. M. Oks and F. Oustry. Worst-case value-at-risk and robust portfolio optimization: A conic programming approach. *Oper. Res. Forthcoming*,2002.
35. P. A. Frost and J. E. Savarino An empirical Bayes approach to efficient portfolio selection. *J. Fin. Quan. Anal*, 293-305, 1986.

36. R. Armananzas and J. A. Lozano, A multiobjective approach to the portfolio optimization problem *Technical report, ISG - Department of Computer Science and Artifical Intelligence*, 2005.
37. R. D. Luce and H. Raiffa. Games and decisions : Intoduction and critical survey. *New York : Wiley*, 1957.
38. R. Montemanni. A benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 1490, 2006.
39. R. Montemanni and L. Gambardella. An exact algorithm for the robust shortest path problem with interval data. *Computers and Operations Research*, 1680, 2004.
40. R. O. Michaud Efficient Asset Management: A Practical Guide to Stock Portfolio Management and Asset Allocation. *Financial Management Association, Survey and Synthesis Series. HBS Press, Boston* ,1998.
41. R. W. Klein and V. S. Bawa The effect of estimation risk on optimal portfolio choice. *J. Financial. Econom*, 215-231, 1976.
42. S. C. Chiam, K. C. Tan and A. Al Mamum, Evolutionary multiobjective portfolio optimization in practical context *International Journal of Automation and Computing*, 67-80, January 2008.
43. V. Baudoui. *Optimisation robuste multiobjectif par modle de substitution.* Thse de Doctorat, Universit de Toulouse, Toulouse, 2012.
44. V. Gabrel and C.Murat. Robust shortest path problems. *LAMSAD*, 71-94, 2007.
45. V. Gabrel et C. Murat. Robustness and duality in linear programming. Journal of the *Operational Research Society*, 1296, 2010.
46. V. K. Chopra. Improving optimization. *J. Investing 8(Fall)*, 51-59, 1993.
47. C. A. Coello Coello and G. B. Lamont, Applications of Multi-Objective Evolutionary Algorithms *World Scientific, Singapore*, 2004.

How Distance based Parameter Adaptation Affects Population Diversity

Adam Viktorin[1][0000−0003−0861−0340], Roman Senkerik[1][0000−0002−5839−4263], Michal Pluhacek[1][0000−0002−3692−2838], and Tomas Kadavy[1][0000−0002−3341−4336]

Tomas Bata University in Zlin, Faculty of Applied Informatics, T. G. Masaryka 5555, 760 01, Zlin Czech Republic
aviktorin@utb.cz, senkerik@utb.cz, pluhacek@utb.cz, kadavy@utb.cz

Abstract. This paper discusses the effect of distance based parameter adaptation on the population diversity of the Success-History based Adaptive Differential Evolution (SHADE). The distance-based parameter adaptation was designed to promote exploration over exploitation and provide better search capabilities of the SHADE algorithm in higher dimensional objective spaces. The population diversity is recorded on the 15 test functions from the CEC 2015 benchmark set in two-dimensional settings, 10*D* and 30*D*, to provide the empiric evidence of a beneficial influence of the distance based parameter adaptation in comparison with the objective function value based approach. ...

Keywords: Distance-based parameter adaptation, SHADE, Population diversity

1 Introduction

The original Differential Evolution (DE) algorithm that was proposed for global optimization by Storn and Price in 1995 [1] has three main control parameters: population size *NP*, scaling factor *F* and crossover rate *CR*. As it was shown in [2] and [3], the setting of these control parameters is crucial for the performance of the algorithm, and there seems to be no universal setting, which is in accordance with the famous no free lunch theorem [4]. Due to this fact, researchers in the DE field are trying to overcome this problem with self-adaptive variants of DE, which do not require fine-tuning of the control parameters to the given optimization task. And since the DE research community is fairly active, there have been numerous updated and improved DE versions over the last few years. Various directions of the research were recently nicely surveyed in the Das, Mullick and Suganthan's paper [5].

One of the most successful novel variants of adaptive DE algorithm is Success-History based Adaptive Differential Evolution (SHADE) [6]. Its superiority was proved on the last five CEC competitions in continuous optimization, where SHADE or its updated variants placed on the top ranks (CEC2013 – SHADE placed 3^{rd}, CEC2014 – L-SHADE [7] placed 1^{st}, CEC2015 – SPS-L-SHADE-EIG [8] placed 1^{st}, CEC2016 – LSHADE_EpSin [9] placed on joint 1^{st} place, CEC2017 – jSO [10] placed 1st). Therefore, the SHADE algorithm was selected as a basis for this study.

The adaptive mechanism in Tanabe and Fukunaga's SHADE is based on the improvement in objective function value from the original individual to the trial individual.

Scaling factor and crossover rate values that were used for the generation of successful trial individuals are then subject to the comparison based on the objective function value improvement and the ones with the highest improvement have the highest weights in the forthcoming calculation of the values that will be stored in the algorithm's memory of successful control parameter settings. Thus, this approach benefits exploitation of the objective space rather than the exploration. Due to this fact, the algorithm is subject to premature convergence when solving optimization problems of higher dimensionalities. In this paper, a novel approach, which considers the distance between the original and trial individuals rather than the objective function improvement is analyzed from the perspective of performance and its effect on population diversity. Maintaining of the population diversity is an interesting task which was lately studied in numerous papers. In [11], auto-enhanced population diversity is proposed, which regenerates individuals components based on the detection of stagnation in respective dimension, in [12], a diversity-based population strategy serves for population size management, in [13], population diversity is maintained by scattering individuals from the centre of the population whenever the variance in objective function values of the population drops below certain level, and finally in [14], population diversity is maintained at a predefined value by increasing or decreasing the population size after each generation. The aforementioned approaches to population diversity maintaining are based on artificial changes to the population, whereas approach proposed in this paper is based on a different view at the information exchange between individuals, where the position change is more valuable for the optimization than the objective function improvement. Therefore, such approach does not lose any of the population shared knowledge, which might be lost in artificial changes of the population. Proposed distance based adaptation is also applicable to any SHADE-based algorithm.

The rest of the paper is structured as follows: The next Section describes original DE algorithm, the Section that follows provides the description of SHADE and Section 4 is devoted to the distance based parameter adaptation mechanism. Sections 5, 6 and 7 deal with experimental setting, results, their discussion and conclusion correspondingly.

2 Differential Evolution

The DE algorithm is initialized with a random population of individuals $\boldsymbol{P}$, that represent solutions of the optimization problem. The population size NP is set by the user along with other control parameters – scaling factor F and crossover rate CR.

In continuous optimization, each individual is composed of a vector $\boldsymbol{x}$ of length D, which is a dimensionality (number of optimized attributes) of the problem, and each vector component represents a value of the corresponding attribute, and of objective function value $f(\boldsymbol{x})$.

For each individual in a population, three mutually different individuals are selected for mutation of vectors and the resulting mutated vector $\boldsymbol{v}$ is combined with the original vector $\boldsymbol{x}$ in the crossover step. The objective function value $f(\boldsymbol{u})$ of the resulting trial vector $\boldsymbol{u}$ is evaluated and compared to that of the original individual. When the quality (objective function value) of the trial individual is better, it is placed into the next generation, otherwise, the original individual is placed there. This step is called selection.

The process is repeated until the stopping criterion is met (e.g., the maximum number of objective function evaluations, the maximum number of generations, the low bound for diversity between objective function values in population).

The following sections describe four steps of DE: Initialization, mutation, crossover, and selection.

2.1 Initialization

As aforementioned, the initial population $\boldsymbol{P}$ with NP individuals is randomly generated. For this purpose, the individual vector xi components are generated by Random Number Generator (RNG) with uniform distribution from the range which is specified for the problem by lower and upper bound (1).

$$\boldsymbol{x}_{j,i} = U\left[lower_j,\ upper_j\right] \text{ for } j = 1,\ \ldots,\ D \tag{1}$$

Where i is the index of a current individual, j is the index of current attribute and D is the dimensionality of the problem.

In the initialization phase, a scaling factor value F and crossover value CR has to be assigned as well. The typical range for F value is [0, 2] and for CR, it is [0, 1].

2.2 Mutation

In the mutation step, three mutually different individuals $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, $\boldsymbol{x}_{r3}$ from a population are randomly selected and combined by the mutation strategy. The original mutation strategy of canonical DE is "rand/1" and is depicted in (2).

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F\left(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}\right) \tag{2}$$

Where $r1 \neq r2 \neq r3 \neq i$, F is the scaling factor, and $\boldsymbol{v}_i$ is the resulting mutated vector.

2.3 Crossover

In the crossover step, the mutated vector $\boldsymbol{v}_i$ is combined with the original vector $\boldsymbol{x}_i$ to produce the trial vector $\boldsymbol{u}_i$. The binomial crossover (3) is used in canonical DE.

$$u_{j,i} = \begin{cases} v_{j,i} & \text{if } U\,[0,1] \leq CR \text{ or } j = j_{rand} \\ x_{j,i} & \text{otherwise} \end{cases} \tag{3}$$

Where CR is the used crossover rate value, and j_{rand} is an index of an attribute that has to be from the mutated vector $\boldsymbol{v}_i$ (this ensures generation of a vector with at least one new component).

2.4 Selection

The selection step ensures that the optimization will progress towards better solutions because it allows only individuals of better or at least equal objective function value to proceed into the next generation *G+1* (4).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} \text{ if } f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} \qquad \text{otherwise} \end{cases} \tag{4}$$

Where G is the index of the current generation. The basic concept of the DE algorithm is depicted in pseudo-code below.
Algorithm pseudo-code 1: DE

Algorithm 1 DE

Set *NP, CR, F* and stopping criterion;
$G = 0$, $\boldsymbol{x}_{best} = \{\}$;
Randomly initialize (1) population $\boldsymbol{P} = (\boldsymbol{x}_{1,G}, \ldots, \boldsymbol{x}_{NP,G})$;
$\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
while stopping criterion not met **do**
    for $i = 1$ to NP **do**
        $\boldsymbol{x}_{i,G} = \boldsymbol{P}[i]$;
        $\boldsymbol{v}_{i,G}$ by mutation (2);
        $\boldsymbol{u}_{i,G}$ by crossover (3);
        if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ **then**
            $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$;
        else
            $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$;
        end if
        $\boldsymbol{x}_{i,G+1} \rightarrow \boldsymbol{P}_{new}$;
    end for
    $\boldsymbol{P} = \boldsymbol{P}_{new}$, $\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
end while
return $\boldsymbol{x}_{best}$ as the best found solution

3 SHADE

In SHADE, the only control parameter that can be set by the user is population size *NP*, the other two (*F, CR*) are adapted to the given optimization task, a new parameter *H* is introduced, which determines the size of *F* and *CR* value memories. The initialization step of the SHADE is, therefore, similar to DE. Mutation, however, is completely different because of the used strategy "current-to-*p*best/1" and the fact that it uses different scaling factor value F_i for each individual. Crossover is still binary, but similarly to the mutation and scaling factor values, crossover rate value CR_i is also different for each individual. The selection step is the same, and therefore following sections describe only the different aspects of initialization, mutation and crossover.

3.1 Initialization

As aforementioned, the initial population $\boldsymbol{P}$ is randomly generated as in DE, but additional memories for F and CR values are initialized as well. Both memories have the same size H and are equally initialized, the memory for CR values is titled $\boldsymbol{M}_{CR,}$ and the memory for F is titled $\boldsymbol{M}_F$. Their initialization is depicted in (5).

$$M_{CR,i} = \ M_{F,i} = 0.5 \text{ for } i = 1, \ldots, H \tag{5}$$

Also, the external archive of inferior solutions $\boldsymbol{A}$ is initialized. Since there are no solutions so far, it is initialized empty $\boldsymbol{A} = \emptyset$, and its maximum size is set to NP.

3.2 Mutation

Mutation strategy "current-to-$p$best/1" was introduced in [15] and unlike "rand/1", it combines four mutually different vectors $pbest \neq r1 \neq r2 \neq i$ (6).

$$\boldsymbol{v}_i = \ \boldsymbol{x}_i + F_i \left(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i \right) + F_i \left(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2} \right) \tag{6}$$

Where $\boldsymbol{x}_{pbest}$ is randomly selected from the best $NP \times p$ individuals in the current population. The p value is randomly generated for each mutation by RNG with uniform distribution from the range $[p_{min}, 0.2]$. Where $p_{min} = 2/NP$. Vector $\boldsymbol{x}_{r1}$ is randomly selected from the current population, and vector $\boldsymbol{x}_{r2}$ is randomly selected from the union of current population $\boldsymbol{P}$ and archive $\boldsymbol{A}$. The scaling factor value F_i is given by (7).

$$F_i = C \left[M_{F,r}, 0.1 \right] \tag{7}$$

Where $M_{F,r}$ is a randomly selected value (by index r) from $\boldsymbol{M}_F$ memory and C stands for Cauchy distribution, therefore the F_i value is generated from the Cauchy distribution with location parameter value $M_{F,r}$ and scale parameter value 0.1. If the generated value $F_i > 1$, it is truncated to 1, and if it is $F_i \leq 0$, it is generated again by (7).

3.3 Crossover

Crossover is the same as in (3), but the CR value is changed to CR_i, which is generated separately for each individual (8). The value is generated from the Gaussian distribution with a mean parameter value of $M_{CR,r}$, which is randomly selected (by the same index r as in mutation) from $\boldsymbol{M}_{CR}$ memory and standard deviation value of 0.1.

$$CR_i = N \left[M_{CR,r}, 0.1 \right] \tag{8}$$

3.4 Historical Memory Updates

Historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ are initialized according to (5), but its components change during the evolution. These memories serve to hold successful values of F and CR used in mutation and crossover steps (successful regarding producing trial individual better than the original individual). During one generation, these successful values

are stored in corresponding arrays $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$. After each generation, one cell of $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ memories is updated. This cell is given by the index k, which starts at 1 and increases by 1 after each generation. When it overflows the memory size H, it is reset to 1. The new value of k-th cell for $\boldsymbol{M}_F$ is calculated by (9) and for $\boldsymbol{M}_{CR}$ by (10).

$$M_{F,k} = \begin{cases} \text{mean}_{WL}\left(\boldsymbol{S}_F\right) \text{ if } \boldsymbol{S}_F \neq \emptyset \\ M_{F,k} \qquad \text{otherwise} \end{cases} \tag{9}$$

$$M_{CR,k} = \begin{cases} \text{mean}_{WL}\left(\boldsymbol{S}_{CR}\right) \text{ if } \boldsymbol{S}_{CR} \neq \emptyset \\ M_{CR,k} \qquad \text{otherwise} \end{cases} \tag{10}$$

Where $\text{mean}_{WL}()$ stands for weighted Lehmer (11) mean.

$$\text{mean}_{WL}\left(\boldsymbol{S}\right) = \frac{\sum_{k=1}^{|\boldsymbol{S}|} w_k \bullet S_k^2}{\sum_{k=1}^{|\boldsymbol{S}|} w_k \bullet S_k} \tag{11}$$

Where the weight vector $\boldsymbol{w}$ is given by (12) and is based on the improvement in objective function value between trial and original individuals.

$$w_k = \frac{\text{abs}\left(f\left(\boldsymbol{u}_{k,G}\right) - f\left(\boldsymbol{x}_{k,G}\right)\right)}{\sum_{m=1}^{|S_{CR}|} \text{abs}\left(f\left(\boldsymbol{u}_{m,G}\right) - f\left(\boldsymbol{x}_{m,G}\right)\right)} \tag{12}$$

Moreover, since both arrays $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$ have the same size, it is arbitrary which size will be used for the upper boundary for m in (12).

The pseudo-code of the SHADE algorithm is depicted below.

4 Distance based Parameter Adaptation

The original adaptation mechanism for scaling factor and crossover rate values uses weighted forms of means (11), where weights are based on the improvement in objective function value (12). This approach promotes exploitation over exploration, and therefore might lead to premature convergence, which could be a problem especially in higher dimensions.

The distance approach is based on the Euclidean distance between the trial and the original individual, which slightly increases the complexity of the algorithm by replacing simple difference by Euclidean distance computation for the price of stronger exploration. In this case, scaling factor and crossover rate values connected with the individual that moved the furthest will have the highest weight (13).

$$w_k = \frac{\sqrt{\sum_{j=1}^{D}\left(u_{k,j,G} - x_{k,j,G}\right)^2}}{\sum_{m=1}^{|S_{CR}|}\sqrt{\sum_{j=1}^{D}\left(u_{m,j,G} - x_{m,j,G}\right)^2}} \tag{13}$$

Therefore, the exploration ability is rewarded, and this should lead to avoidance of the premature convergence in higher dimensional objective spaces. Such approach might also be useful for constrained problems, where constrained areas could be overcame by increased changes of individual's components.

Algorithm 2 SHADE

Set NP, H and stopping criterion;
$G = 0$, $\boldsymbol{x}_{best} = \{\}$, $k = 1$, $p_{min} = 2/NP$, $\boldsymbol{A} = \emptyset$;
Randomly initialize (1) population $\boldsymbol{P} = (\boldsymbol{x}_{1,G}, \ldots, \boldsymbol{x}_{NP,G})$;
Set $\boldsymbol{M}_F$ *and* $\boldsymbol{M}_{CR}$ according to (5);
$\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
while stopping criterion not met **do**
    $\boldsymbol{S}_F = \emptyset$, $\boldsymbol{S}_{CR} = \emptyset$;
    for $i = 1$ to NP **do**
        $\boldsymbol{x}_{i,G} = \boldsymbol{P}[i]$;
        $r = U[1, H]$, $p_i = U[p_{min}, 0.2]$;
        Set F_i by (7) and CR_i by (8);
        $\boldsymbol{v}_{i,G}$ by mutation (6);
        $\boldsymbol{u}_{i,G}$ by crossover (3);
        if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ **then**
            $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$;
            $\boldsymbol{x}_{i,G} \rightarrow \boldsymbol{A}$;
            $F_i \rightarrow \boldsymbol{S}_F$, $CR_i \rightarrow \boldsymbol{S}_{CR}$;
        else
            $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$;
        end if
        if $|\boldsymbol{A}| > NP$ **then**
            Randomly delete an ind. from $\boldsymbol{A}$;
        end if
        $\boldsymbol{x}_{i,G+1} \rightarrow \boldsymbol{P}_{new}$;
    end for
    if $\boldsymbol{S}_F \neq \emptyset$ **and** $\boldsymbol{S}_{CR} \neq \emptyset$ **then**
        Update $\boldsymbol{M}_{F,k}$ (9) and $\boldsymbol{M}_{CR,k}$ (10), k++;
        if $k > H$ **then**
            $k = 1$;
        end if
    end if
    $\boldsymbol{P} = \boldsymbol{P}_{new}$, $\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
end while
return $\boldsymbol{x}_{best}$ as the best found solution

5 Experimental Setting

The CEC 2015 benchmark set states that each function should be run 51 times and the stopping criterion should be set to 10,000 × D objective function evaluations. These requirements were adhered to, and two-dimensional settings were selected 10D and 30D to provide a robust comparison. The convergence and population diversity were recorded for both versions of the tested algorithm – original SHADE and SHADE with distance based parameter adaptation abbreviated to Db_SHADE. The used population diversity measure is described in the next section.

Both algorithm variants had the same set of variable parameters – population size NP was set to 100, the maximum size of the optional archive $|A|$ was set to NP, and the historical memory size H was set to 10.

5.1 Population Diversity Measure

The Population Diversity (PD) measure used in this paper was described in [14] and is based on the sum of deviations (15) of individual's components from their corresponding means (14).

$$\overline{x_j} = \frac{1}{NP} \sum_{i=1}^{NP} x_{ij} \tag{14}$$

$$PD = \sqrt{\frac{1}{NP} \sum_{i=1}^{NP} \sum_{j=1}^{D} \left(x_{ij} - \overline{x_j}\right)^2} \tag{15}$$

Where i is the population member iterator and j is the vector component iterator.

6 Results and Discussion

The comparative results for both dimensional settings are in Tables 1 and 2, where the last column depicts the result of the Wilcoxon rank-sum test with significance level of 0.05. No significant difference in performance between SHADE and Db_SHADE algorithm is represented by "=" sign when the SHADE algorithm performs significantly better; there would be "-" sign and when the distance based version performs significantly better, the "+" sign is used. As it can be seen from results in Table 1, the performance of both versions is comparable with only one win for the Db_SHADE algorithm. This was suspected as the dimensionality of the problem is quite low, and the SHADE algorithm does not tend to converge prematurely. The situation is more interesting in the second table, where there are 6 significantly different results in performance, and all of them are in favor of the Db_SHADE algorithm.

The convergence comparison plots and population diversity plots with confidence intervals are provided for the six functions in 30D, where the performance is significantly different in Figures 1 to 3. In these figures, it can be seen that the population diversity is maintained longer, which leads to a more explorative manner during the exploration phase of the algorithm, while the exploitation phase is still present in the later generations. As for the other functions from the benchmark with no significant

difference in results, the population diversity is also higher in the case of Db_SHADE algorithm, but it does not help to improve the optimization result significantly. Therefore, the further study of the on-line effects of the distance based adaptation is needed.

Table 1. SHADE vs. Db_SHADE on CEC2015 in 10*D*.

	SHADE		**Db_SHADE**		
f	Median	Mean	Median	Mean	Result
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
3	2.00E+01	1.89E+01	2.00E+01	1.92E+01	=
4	3.07E+00	2.97E+00	3.06E+00	2.98E+00	=
5	2.21E+01	3.42E+01	2.98E+01	4.52E+01	=
6	2.20E-01	2.97E+00	4.16E-01	8.08E-01	=
7	1.67E-01	1.88E-01	1.73E-01	1.91E-01	=
8	8.15E-02	2.69E-01	4.28E-02	2.06E-01	=
9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=
10	2.17E+02	2.17E+02	2.17E+02	2.17E+02	=
11	3.00E+02	1.66E+02	3.00E+02	2.01E+02	=
12	1.01E+02	1.01E+02	1.01E+02	1.01E+02	+
13	2.78E+01	2.78E+01	2.79E+01	2.76E+01	=
14	2.94E+03	4.28E+03	2.98E+03	4.66E+03	=
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=

7 Conclusion

This paper provided an analysis of the effect of distance based parameter adaptation in SHADE algorithm to population diversity. The analysis was done on two test cases – 15 test functions from the CEC2015 benchmark set in 10*D* and 30*D*. The presumption that the effect will be more visible in higher dimensional setting was confirmed. It can be seen, that the diversity of the population is maintained for a longer period, therefore prolonging the exploration phase and avoiding premature convergence of the algorithm. This is in turn beneficial for the result of the optimization.

However, there is still too much of unused computational time and the tendency for premature convergence is still strong. Therefore, the future research direction for the authors is to address these issues.

Acknowledgements. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation

Table 2. SHADE vs. Db_SHADE on CEC2015 in 30*D*.

	SHADE		**Db_SHADE**		
f	Median	Mean	Median	Mean	Result
1	3.73E+01	2.62E+02	2.12E+01	2.42E+02	=
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
3	2.01E+01	2.01E+01	2.01E+01	2.01E+01	=
4	1.41E+01	1.41E+01	1.32E+01	1.31E+01	=
5	1.55E+03	1.50E+03	1.54E+03	1.52E+03	=
6	5.36E+02	5.73E+02	3.37E+02	3.48E+02	+
7	7.17E+00	7.26E+00	6.81E+00	6.74E+00	+
8	1.26E+02	1.21E+02	5.27E+01	7.38E+01	+
9	1.03E+02	1.03E+02	1.03E+02	1.03E+02	+
10	6.27E+02	6.22E+02	5.29E+02	5.32E+02	+
11	4.53E+02	4.50E+02	4.10E+02	4.16E+02	+
12	1.05E+02	1.05E+02	1.05E+02	1.05E+02	=
13	9.52E+01	9.50E+01	9.47E+01	9.50E+01	=
14	3.21E+04	3.24E+04	3.22E+04	3.24E+04	=
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=

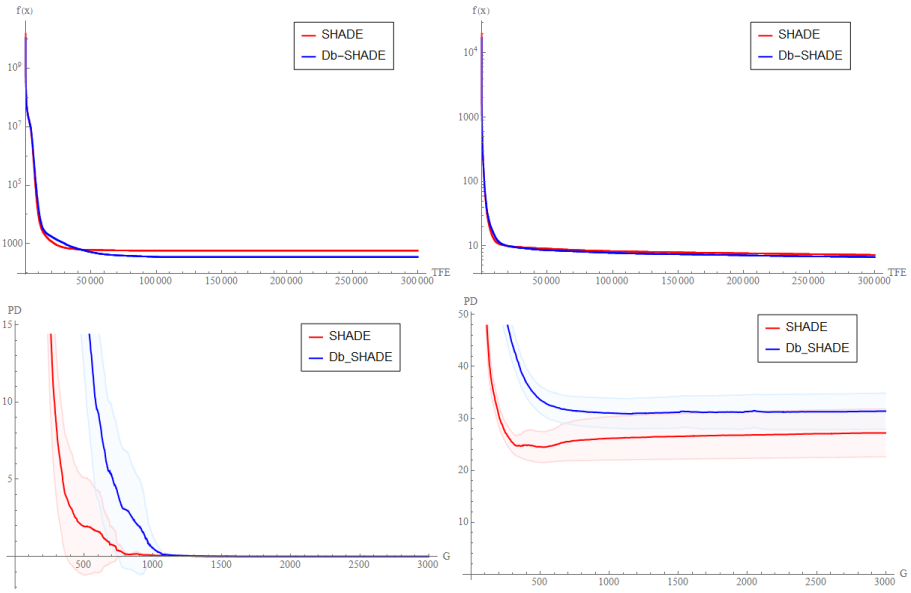


Fig. 1. Convergence plots (top) and population diversity plots (bottom) of CEC2015 test functions *f6* (left) and *f7* (right) in 30*D*.

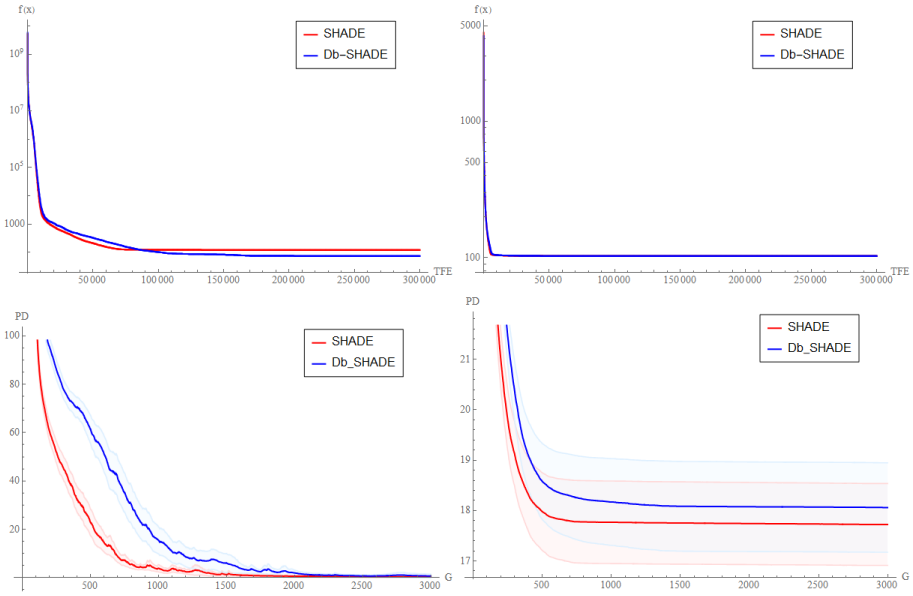


Fig. 2. Convergence plots (top) and population diversity plots (bottom) of CEC2015 test functions *f8* (left) and *f9* (right) in 30*D*.

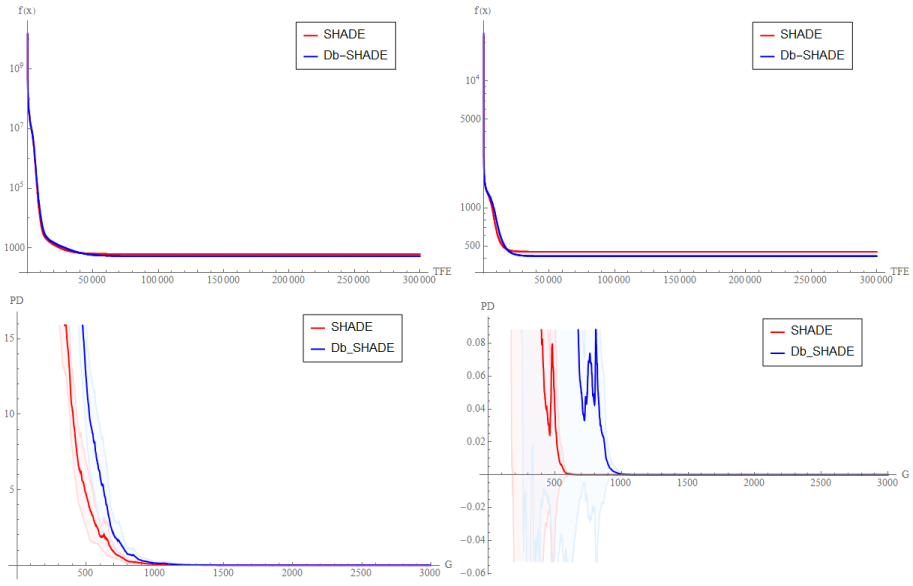


Fig. 3. Convergence plots (top) and population diversity plots (bottom) of CEC2015 test functions *f10* (left) and *f11* (right) in 30*D*.

by Joining Theory and Practice (ImAppNIO), and Action IC406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet).

References

1. Storn, R., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces (Vol. 3). Berkeley: ICSI.
2. Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. Advances in intelligent systems, fuzzy systems, evolutionary computation, 10, 293-298.
3. Liu, J., & Lampinen, J. (2002). On setting the control parameter of the differential evolution method. In Proceedings of the 8th international conference on soft computing (MENDEL 2002) (pp. 11-18).
4. Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), 67-82.
5. Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution–An updated survey. Swarm and Evolutionary Computation, 27, 1-30.
6. Tanabe, R., & Fukunaga, A. (2013, June). Success-history based parameter adaptation for differential evolution. In Evolutionary Computation (CEC), 2013 IEEE Congress on (pp. 71-78). IEEE.
7. Tanabe, R., & Fukunaga, A. S. (2014, July). Improving the search performance of SHADE using linear population size reduction. In Evolutionary Computation (CEC), 2014 IEEE Congress on (pp. 1658-1665). IEEE.
8. Guo, S. M., Tsai, J. S. H., Yang, C. C., & Hsu, P. H. (2015, May). A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 1003-1010). IEEE.
9. Awad, N. H., Ali, M. Z., Suganthan, P. N., & Reynolds, R. G. (2016, July). An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (pp. 2958-2965). IEEE.
10. Brest, J., Maučec, M. S., & Bošković, B. (2017, June). Single objective real-parameter optimization: Algorithm jSO. In Evolutionary Computation (CEC), 2017 IEEE Congress on (pp. 1311-1318). IEEE.
11. Yang, M., Li, C., Cai, Z., & Guan, J. (2015). Differential evolution with auto-enhanced population diversity. IEEE transactions on cybernetics, 45(2), 302-315.
12. Zhang, C., Zhao, Z., Yang, T., & Fan, B. (2016, June). Adaptive differential evolution with coordinated crossover and diversity-based population. In Control and Automation (ICCA), 2016 12th IEEE International Conference on (pp. 947-950). IEEE.
13. Zhao, L., Sun, C. J., Huang, X. C., & Zhou, B. X. (2016, July). Differential Evolution with strategy of improved population diversity. In Control Conference (CCC), 2016 35th Chinese (pp. 2784-2787). IEEE.
14. Poláková, R., Tvrdík, J., Bujok, P., & Matoušek, R. (2016). Population-size adaptation through diversity-control mechanism for differential evolution. In MENDEL, 22th International Conference on Soft Computing (pp. 49-56).
15. Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, 13(5), 945-958.

Comparing Boundary Control Methods for Firefly Algorithm

Tomas Kadavy[0000−0002−3341−4336], Michal Pluhacek[0000−0002−3692−2838], Adam Viktorin[0000−0003−0861−0340], and Roman Senkerik[0000−0002−5839−4263]

Tomas Bata University in Zlin, T.G.Masaryka 5555, Zlin, Czech Republic,
kadavy@utb.cz, pluhacek@utb.cz, aviktorin@utb.cz, senkerik@utb.cz

Abstract. This paper compares four different methods for handling the roaming behavior of fireflies in the firefly algorithm. The problems of boundary constrained optimiza-tion forces the algorithm to actively keep the fireflies inside the feasible area of possible solutions. The recent CEC17 benchmark suite is used for the perfor-mance comparison of the methods and the results are compared and tested for sta-tistical significance.

Keywords: Firefly Algorithm, Boundary, Levy flight

1 Introduction

Firefly Algorithm (FA) [1], [2] is one of the modern and versatile optimization algorithms developed by Yang in 2008. Since then, the FA has proven its robust perfor-mance either on single objective [3] or many/multi-objective optimization problems [4]. Recently, many new modifications have been introduced to improve the results and overall quality of FA. Modifications like Levy flights [5], or chaos-driven FA [6] show the large potential of this modern algorithm.

One of the tasks left to discuss lies in the question what to do if fireflies (or particles in general) try to violate defined boundaries by particular optimization problem. When optimizing the real problem, very often are optimized parameters limited. This is caused in many cases due to simple physical nature of the problem (for example length cannot be in negative numbers). The violation of particle could happen whenever a new position is evaluated thanks to the nature of the metaheuristic optimization algorithm. Select the most suitable border strategy is a difficult task as the numerous similar studies for PSO show [7], [8].

Since there are no such studies for FA available in the literature, we have decided to perform and present this original experimental research. In this paper, three relatively common borders strategies (or rather methods) are implemented and compared on CEC17 benchmark set [9]. Also, a new hypersphere border strategy [10], initially developed for PSO, is also tested and compared given its promising results on some test functions.

The paper is structured as follows. The FA and its Levy flight modification are described in Section 2. The border strategies are in detail described in

Section 3. The experiment setup is detailed in Section 4. Section 5 contains statistical overviews of the results and performance comparisons obtained during the evaluation on benchmark set. Finally, the paper is concluded in Section 6.

2 Firefly Algorithm

This optimization nature-based algorithm was developed and introduced by Yang in 2008 [1]. The fundamental principle of this algorithm lies in simulating the mating behavior of fireflies at night when fireflies emit light to attract a suitable partner. The main idea of Firefly Algorithm (FA) is that the objective function value that is optimized is associated with the flashing light of these fireflies. The author for simplicity set a couple of rules to describe the algorithm itself:

- The brightness of each firefly is based on the objective function value.
- The attractiveness of a firefly is proportional to its brightness. This means that the less bright firefly is lured towards, the brighter firefly. The brightness depends on the environment or the medium in which fireflies are moving and decreases with the distance between each of them.
- All fireflies are sexless, and it means that each firefly can attract or be lured by any of the remaining ones.

The movement of one firefly towards another one is then defined by equation (1). Where $x_i^{'}$ is a new position of a firefly i, x_i is the current position of firefly i and x_j is a selected brighter firefly (with better objective function value). The is a randomization parameter and *sign* simply provides random direction -1 or 1.

$$x_i^{'} = x_i + \beta \cdot (x_j - x_i) + \alpha \cdot sign \tag{1}$$

The brightness I of a firefly is computed by the equation (2). This equation of brightness consists of three factors mentioned in the rules above. On the objective function value, the distance between two fireflies and the last factor is the absorption factor of a media in which fireflies are.

$$I = \frac{I_0}{1 + \gamma r^m} \tag{2}$$

WhereI_o is the objective function value, the γ stands for the light absorption parameter of a media in which fireflies are and the m is another user-defined coefficient and it should be set $m \geq 1$. The variable r is the Euclidian distance(3) between the two compared fireflies.

$$r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{3}$$

Where r_{ij} is the Euclidian distance between fireflies x_i and x_j. The d is current dimension size of the optimized problem.

The attractiveness β (4) is proportional to brightness I as mentioned in rules above and so these equations are quite similar to each other. The β_0 is the initial attractiveness defined by the user, the γ is again the light absorption parameter and the r is once more the Euclidian distance. The m is also the same as in equation (2).

$$\beta = \frac{\beta_0}{1 + \gamma r^m} \tag{4}$$

One of the recent and quite commonly used modifications of FA lies in the introduction of Lévy flights [5], [11]. With this Lévy flights characteristic the new modification of FA is called Lévy-flight Firefly Algorithm (LFA). This modification only customizes the computation of the position of fireflies described originally in equation (1). The new modified version with Lévy flight is defined in (5).

$$x_i^{'} = x_i + \beta \cdot (x_j - x_i) + \alpha \cdot sign \bigoplus Lévy \tag{5}$$

The Lévy stands for Lévy flight randomization together with λ being the randomization parameter as in the original equation. The product $\bigoplus$ means entrywise multiplications. The Lévy distribution is drawn as (6).

$$Lévy \sim u = t^{-\lambda} \tag{6}$$

Where parameter λ is another user-defined variable that controls the Lévy distribution described in [5] and it should be set in range $1 < \lambda \leq 3$. The pseudocode below shows the fundamentals of FA operations.

```
1.  FA initialization
2.  while(terminal condition not met)
3.    for i = 1 to all fireflies
4.      for j = 1 to all fireflies
5.        if(Ij < Ii) then
6.          move xi to xj
7.          evaluate xi
8.        end if
9.      end for j
10.   end for i
11.   record the best firefly
12. end while
```

Fig. 1. FA pseudocode.

3 Border Strategies

Every time, when a single objective function optimization problem has defined a range where the best value is being found by the metaheuristic algorithm, one of the many difficult tasks could arise to an operator or a user. After each step

of an algorithm, in this case after position update of a firefly, the new position should be checked if it lies in the appropriate range or boundaries (inside space of feasible solution). In case that the new position of the particle is outside this allowed region a certain correction has to be made. Several possible correction methods or strategies could do the trick. However, select the most appropriate is not an easy task since each of them could have a very different effect on the algorithm ability to achieve a good solution. For this paper, the few most common ones were selected and compared together to show how they could affect the FA on different benchmark functions.

3.1 Hard borders

The particle (or in this case firefly) cannot cross the given boundaries in each dimension. This strategy is very simple to implement and is described as (7).

$$x_i^{'} = \begin{cases} x_i = b^u, \ if \ x_i > b^u \\ x_i = b^l, \ if \ x_i < b^l \\ x_i, \ otherwise \end{cases} \tag{7}$$

Where x_i is the position of i firefly before boundary check, the $x_i^{'}$ is a newly updated position after the boundary check and the b^u and b^l are the upper and lower boundary given to each dimension.

3.2 Random position

If a firefly violates the boundary in any dimension, the new position for this firefly for a particular dimension is created between the lower and upper boundary (with a pseudo-random uniform distribution). Again this strategy is rather simple and very easy to implement.

3.3 Hypersphere

This strategy tries to simulate an endless hypersphere. To simplify this statement, an example is given. If a firefly violates upper boundary limit, it appears then in the search space but from the lower boundary. In other words, the upper boundary is neighboring the lower one in corresponding dimension and vice versa.

This strategy also brings one interesting and also an important feature. The firefly has now two options how to achieve a new position when flying towards another firefly. The new possible way is throughout the boundaries which are now passable. The modification of previous equation (5) is given in (8).

$$x_i^{'} = x_i + \beta \cdot v_{ij} + \alpha \cdot sign \bigoplus Lévy \tag{8}$$

Where the v_{ij} is the vector of difference between particles i and j defined as (9).

$$v_{ij} = \begin{cases} \hat{v}_{ij} \ , \ if \ \ |\hat{v}_{ij}| \leq d \\ \hat{v}_{ij} \ mod \ (-d) \ , \ if \ \ (|\hat{v}_{ij}| > d \wedge |\hat{v}_{ij}| > 0) \\ \hat{v}_{ij} \ mod \ (+d) \ , \ if \ \ (|\hat{v}_{ij}| > d \wedge |\hat{v}_{ij}| \leq 0) \end{cases} \tag{9}$$

The $\hat{v}_{ij}$ and range d are computed by formulas (10) and (11), where b^u and b^l are again the upper boundary and lower boundary limits and x_i and x_j are the fireflies i and j.

$$\hat{v}_{ij} = x_j - x_i \tag{10}$$

$$d = \frac{\left|b^u - b^l\right|}{2} \tag{11}$$

3.4 Reflection

The reflection strategy [7] reflects the particle back to feasible space of solution if it tries to violate the defined borders. This strategy tries to emulate the reflection characteristic of for example a mirror. For violated dimension, the correction of a position of a particle is computed as (12). Where again the b^u and b^l are the upper boundary limit and lower boundary limit.

$$x_i^{'} = \begin{cases} x_i^{'} = b^u - (x_i - b^u) \, , \ if \ x_i > b^u \\ x_i^{'} = b^l + \left(b^l - x_i\right) \, , \ if \ x_i < b^l \\ x_i, \ otherwise \end{cases} \tag{12}$$

4 Experimental Setup

The experiments were performed on a set of well-known benchmark functions CEC'17 which are detailly described in [9]. The tested dimensions were 10 and 30. The maximal number of function evaluation was set as 10 000 $\cdot$ *dim* (dimension size). The lower and upper boundary was as $b^l = -100$ and $b^u = 100$ according to CEC'17 definition. The number of fireflies was set to 40 for both dimension sizes. Every test function was repeated for 30 independent runs and the results were statistically evaluated. The benchmark itself includes 30 test functions in four categories: unimodal, multimodal, hybrid and composite types. The global minimum of each function is easy to determine as it is $100 \cdot f_i$ where i is an order of the particular test function.

The parameters of LFA were experimentally set as $\alpha = 0.2$, $\lambda = 1.5$, γ =0.01, $\beta_0 = 0.5$, and $m = 1$.

5 Results

The results of performed experiments are given in this section. Firstly, the results overviews and comparions are presented in Table 1 and Table 2, which contain

the simple statistic like mean, std. dev., min. and max. values. Further, examples of convergence behavior of the compared methods are given in Figs. 2 – 5.

Furthermore, we present the Friedman ranks with critical distance evaluated according to the Bonferroni Dunn post-hoc test for multiple comparisons. The visual outputs of multiple comparisons with rankings are given in Figure 6. The dashed line represents the critical distance from the best boundary method (the lowest mean rank). The critical distance (CD) value for this experiment has been calculated as 0.8586; according to the definition given in (13) and value $q_a = 2.5758$; using $k = 4$ boundary methods and a number of data sets $N = 30$ (30 repeated runs). From the results, it is noticeable, that the hard border and the reflective boundaries are the most favorable methods among the four compared.

$$CD = q_a \sqrt{k\,(k+1)\,/\,(6N)} \tag{13}$$

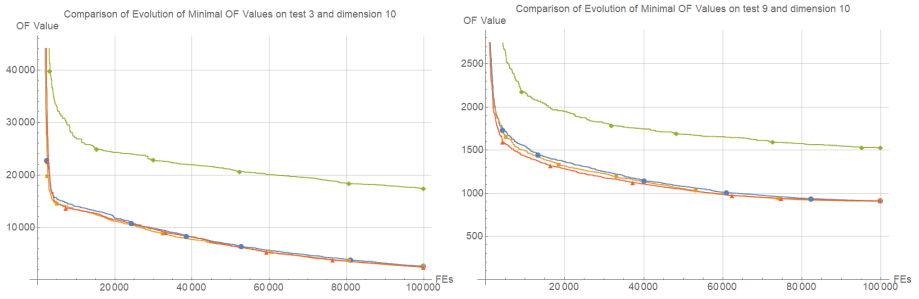


Fig. 2. Convergence plots of CEC2017 test functions *f3* (left) and *f9* (right) in 10D. The blue line stands for Hard borders strategy, the orange line is for Random position, The Hypersphere is in green color and the Reflection is in red.

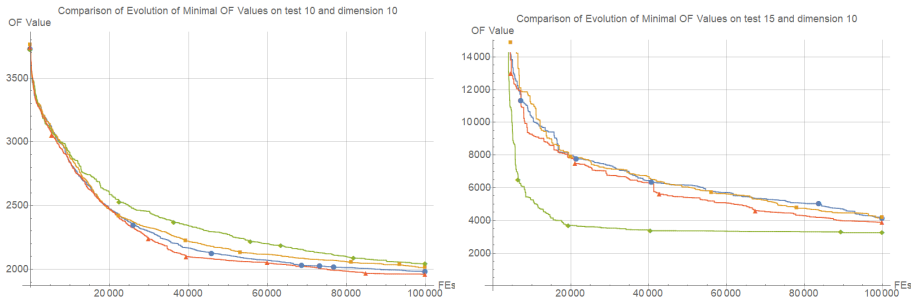


Fig. 3. Convergence plots of CEC2017 test functions *f10* (left) and *f15* (right) in 10D. The blue line stands for Hard borders strategy, the orange line is for Random position, The Hypersphere is in green color and the Reflection is in red.

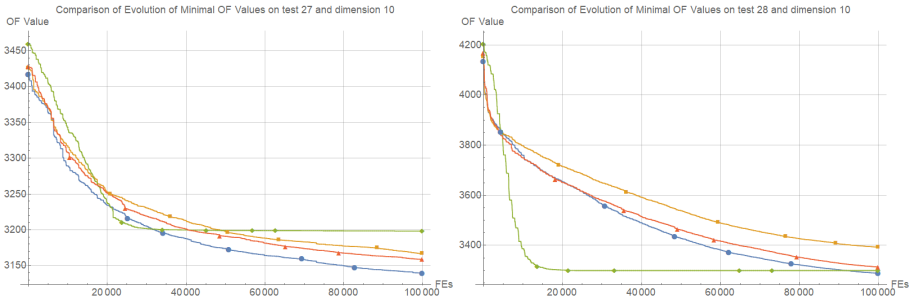


Fig. 4. Convergence plots of CEC2017 test functions *f27* (left) and *f28* (right) in 10*D*. The blue line stands for Hard borders strategy, the orange line is for Random position, The Hypersphere is in green color and the Reflection is in red.

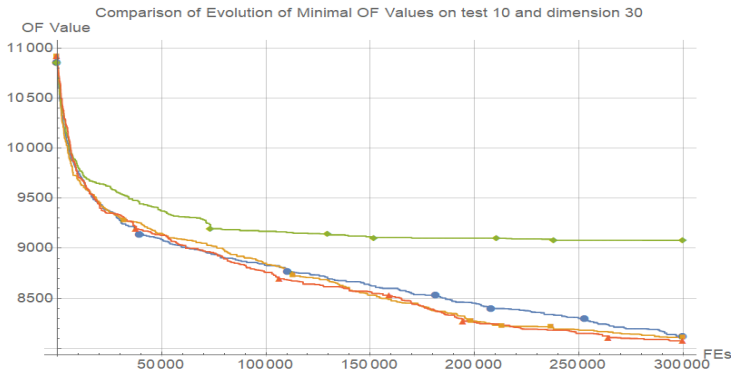


Fig. 5. Convergence graph for f_{10} dimension 30. The blue line stands for Hard borders strategy, the orange line is for Random position, The Hypersphere is in green color and the Reflection is in red.

The illustrative comparisons depicted in Figures 2 – 5 supported by the results of Friedmann rank tests lend weight to the argument that the hypersphere strategy often gives the much slower convergence speed. Although, there are two exceptions to this statement (Fig. 4 and Fig. 5). The other strategies frequently reach almost the same results. The differences between these strategies become noticeable with the increase of the dimension size.

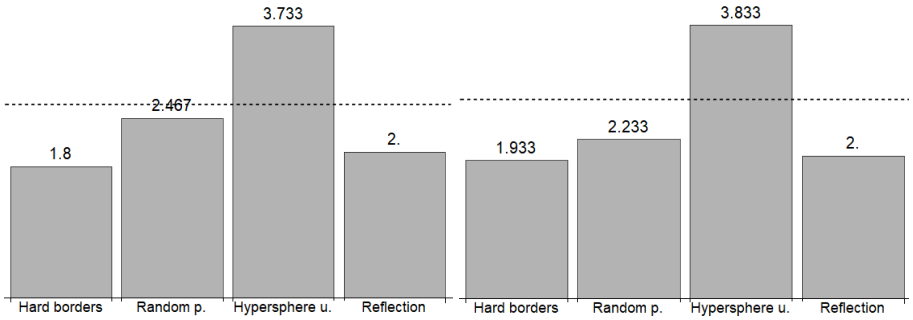


Fig. 6. Friedmann rank test comparison on CEC2017 test functions in $10D$ (left) and $30D$ (right) in $10D$.

Table 1. Statistical results for dimension 10 (mean, std. dev., min., max.)

Function	*Hard*				*Random*				*Hypersphere*				*Reflection*			
f_1	1.38E06	7.85E05	4.81E05	4.39E06	1.44E06	6.64E05	3.84E05	3.10E06	6.42E09	6.12E09	3.10E08	1.97E10	1.55E06	1.10E06	5.17E05	5.28E06
f_2	4.36E05	1.27E06	7.62E05	5.32E06	1.62E05	3.65E05	4.47E03	1.88E06	2.18E12	1.10E13	4.33E05	6.14E13	3.26E05	1.37E06	2.79E03	7.65E06
f_3	2.59E03	1.97E03	5.67E02	9.83E03	2.51E03	1.71E03	5.54E02	8.09E03	1.74E04	9.02E03	6.72E03	4.20E04	2.44E03	1.59E03	6.99E02	7.70E03
f_4	4.09E02	1.21E01	4.00E02	4.69E02	4.13E02	1.49E01	4.01E02	4.57E02	7.52E02	4.60E02	4.35E02	2.31E03	4.11E02	1.37E01	4.00E02	4.67E02
f_5	5.37E02	1.11E01	5.11E02	5.63E02	5.43E02	1.12E01	5.18E02	5.70E02	5.67E02	2.04E01	5.20E02	6.11E02	5.38E02	9.92E00	5.14E02	5.65E02
f_6	6.13E02	7.79E00	6.00E02	6.29E02	6.17E02	8.29E00	6.04E02	6.33E02	6.38E02	1.05E01	6.14E02	6.61E02	6.15E02	8.07E00	6.03E02	6.36E02
f_7	7.37E02	5.74E00	7.23E02	7.49E02	7.36E02	5.64E00	7.26E02	7.51E02	8.61E02	9.53E01	7.54E02	1.08E03	7.36E02	5.78E00	7.22E02	7.48E02
f_8	8.21E02	5.68E00	8.13E02	8.34E02	8.21E02	5.47E00	8.11E02	8.33E02	8.39E02	1.37E01	8.17E02	8.80E02	8.20E02	5.18E00	8.10E02	8.29E02
f_9	9.12E02	2.84E02	9.00E02	1.03E03	9.06E02	1.13E01	9.00E02	9.40E02	1.52E03	3.48E02	9.00E02	2.44E03	9.10E02	2.60E01	9.00E02	1.04E03
f_{10}	1.98E03	1.91E02	1.48E03	2.28E03	2.01E03	2.47E02	1.43E03	2.55E03	2.04E03	3.82E02	1.50E03	2.95E03	1.96E03	2.93E02	1.32E03	2.50E03
f_{11}	1.15E03	3.77E01	1.11E03	1.25E03	1.14E03	2.99E01	1.11E03	1.28E03	1.64E03	6.23E02	1.12E03	4.12E03	1.13E03	1.73E01	1.11E03	1.17E03
f_{12}	8.09E05	7.21E05	4.67E04	2.75E06	1.22E06	1.26E06	1.30E05	6.23E06	6.03E06	1.70E07	9.29E04	9.70E07	5.09E05	6.25E05	3.55E04	3.07E06
f_{13}	8.00E03	2.79E03	2.35E03	1.54E04	8.71E03	3.82E03	3.03E03	2.09E04	1.29E04	5.62E03	3.70E03	2.90E04	7.50E03	2.92E03	2.71E03	1.44E04
f_{14}	1.96E03	7.55E02	1.44E03	4.23E03	2.34E03	1.32E03	1.49E03	7.25E03	2.54E03	1.13E03	1.47E03	5.73E03	1.99E03	8.66E02	1.44E03	6.13E03
f_{15}	4.12E03	2.32E03	1.89E03	1.16E04	4.15E03	1.64E03	1.85E03	8.45E03	3.25E03	9.22E02	1.82E03	5.45E03	3.88E03	1.83E03	1.67E03	8.49E03
f_{16}	1.90E03	9.21E01	1.69E03	2.05E03	1.89E03	8.27E01	1.74E03	2.02E03	1.91E03	9.81E01	1.73E03	2.11E03	1.89E03	1.02E02	1.61E03	2.01E03
f_{17}	1.75E03	1.42E01	1.72E03	1.79E03	1.75E03	1.25E01	1.72E03	1.78E03	1.78E03	3.24E01	1.73E03	1.86E03	1.76E03	1.19E02	1.73E03	1.79E03
f_{18}	6.42E03	5.73E03	2.31E03	2.74E03	5.81E03	4.31E03	2.15E03	2.19E03	1.27E04	7.65E03	3.31E03	4.18E04	6.23E03	3.73E03	1.96E03	1.69E04
f_{19}	3.09E03	1.40E03	1.94E03	7.63E03	3.14E03	1.33E03	1.93E03	8.12E03	4.09E03	2.51E03	1.94E03	1.29E04	3.42E03	1.41E03	1.95E03	8.04E03
f_{20}	2.12E03	4.97E01	2.04E03	2.19E03	2.12E03	4.79E01	2.05E03	2.20E03	2.14E03	6.69E03	2.02E03	2.28E03	2.12E03	5.49E03	2.03E03	2.20E03
f_{21}	2.28E03	5.75E03	2.20E03	2.35E03	2.30E03	5.58E03	2.20E03	2.36E03	2.33E03	4.46E01	2.21E03	2.40E03	2.30E03	4.49E01	2.20E03	2.35E03
f_{22}	2.30E03	1.22E01	2.24E03	2.31E03	2.30E03	2.15E00	2.30E03	2.31E03	2.54E03	3.57E02	2.28E03	3.65E03	2.30E03	3.21E00	2.30E03	2.32E03
f_{23}	2.65E03	4.08E01	2.46E03	2.69E03	2.66E03	2.01E01	2.63E03	2.72E03	2.77E03	5.28E01	2.65E03	2.87E03	2.65E03	6.45E01	2.32E03	2.71E03
f_{24}	2.64E03	1.35E02	2.50E03	2.81E03	2.65E03	1.26E02	2.50E03	2.82E03	2.78E03	1.42E02	2.50E03	2.93E03	2.64E03	1.26E02	2.50E03	2.81E03
f_{25}	2.92E03	2.21E01	2.89E03	2.94E03	2.91E03	5.82E01	2.61E03	2.94E03	3.18E03	3.00E02	2.92E03	3.99E03	2.92E03	2.16E01	2.89E03	2.94E03
f_{26}	3.02E03	3.59E02	2.61E03	3.91E03	3.02E03	3.51E02	2.62E03	4.24E03	3.89E03	5.16E02	3.00E03	4.87E03	3.11E03	3.64E02	2.60E03	4.28E03
f_{27}	3.13E03	2.39E01	3.10E03	3.19E03	3.16E03	2.88E01	3.11E03	3.22E03	3.19E03	5.82E00	3.16E03	3.20E03	3.15E03	3.22E01	3.10E03	3.24E03
f_{28}	3.28E03	1.42E02	2.93E03	3.48E03	3.39E03	1.18E02	3.10E03	3.49E03	3.29E03	0.82E00	3.29E03	3.30E03	3.31E03	1.35E02	3.10E03	3.48E03
f_{29}	3.24E03	3.31E01	3.18E03	3.31E03	3.24E03	3.75E01	3.18E03	3.34E03	3.33E03	8.19E01	3.19E03	3.49E03	3.26E03	4.82E01	3.19E03	3.38E03
f_{30}	4.20E05	4.19E05	1.38E04	1.94E06	6.45E05	5.52E05	3.24E04	2.00E06	3.18E05	5.64E05	3.94E03	2.74E06	5.28E05	4.94E05	5.30E04	2.16E06

Table 2. Statistical results for dimension 30 (mean, std. dev., min., max.)

Function	*Hard*				*Random*				*Hypersphere*				*Reflection*			
f_1	1.02E10	1.90E09	7.20E09	1.42E10	1.07E10	2.39E09	7.00E09	1.64E10	1.21E11	1.71E10	9.19E10	1.65E11	1.03E10	2.85E09	5.89E09	1.82E10
f_2	1.95E34	5.28E34	5.88E28	2.24E35	8.16E33	2.54E34	8.37E27	1.34E35	1.58E53	8.77E53	7.08E43	4.88E54	5.98E34	2.20E35	5.15E28	1.16E36
f_3	6.46E04	1.39E04	3.81E04	9.88E04	5.59E04	1.06E04	3.84E04	7.90E04	1.84E05	3.36E04	1.32E05	2.64E05	6.05E04	9.94E03	4.49E02	8.10E04
f_4	2.23E03	4.67E02	1.36E03	3.49E03	2.46E03	4.65E02	1.69E03	3.80E03	4.25E04	1.04E04	2.37E04	6.63E04	2.30E03	4.98E02	1.44E03	3.90E03
f_5	7.89E02	2.33E01	7.50E02	8.50E02	7.90E02	1.51E01	7.49E02	8.11E02	1.12E03	5.41E01	1.02E03	1.20E03	7.92E02	2.19E01	7.21E02	8.29E02
f_6	6.60E02	6.41E03	6.49E02	6.73E02	6.61E02	4.39E00	6.50E02	6.69E02	7.14E02	8.65E00	6.95E02	7.31E02	6.60E02	5.40E00	6.48E02	6.72E02
f_7	1.08E03	3.78E01	1.01E03	1.15E03	1.06E03	3.40E01	1.00E03	1.13E03	3.19E03	2.91E02	2.42E03	3.83E03	1.07E03	2.94E01	9.93E02	1.13E03
f_8	1.05E03	2.06E01	1.01E03	1.09E03	1.06E03	1.51E01	1.02E03	1.09E03	1.34E03	2.90E01	1.28E03	1.40E03	1.05E03	1.55E01	1.03E03	1.10E03
f_9	5.54E03	1.00E03	3.58E03	7.98E03	5.17E03	6.77E02	3.28E03	6.74E03	2.20E04	3.26E03	1.36E04	2.93E04	5.10E03	8.09E02	3.72E03	7.08E03
f_{10}	8.11E03	5.25E02	6.53E03	8.93E03	8.10E03	4.61E02	6.84E03	8.82E03	9.08E03	3.20E02	7.92E03	9.57E03	8.07E03	3.82E02	6.88E03	8.69E03
f_{11}	3.19E03	9.12E02	2.06E03	5.87E03	3.29E03	6.67E02	2.00E03	4.66E03	2.28E04	6.89E03	1.17E04	3.73E04	3.57E03	1.20E03	2.13E03	6.84E03
f_{12}	7.77E08	2.98E08	2.75E08	1.33E09	1.01E09	3.55E08	3.77E08	1.99E09	2.20E10	6.24E09	9.69E09	3.16E10	8.29E08	3.48E08	2.87E08	1.67E09
f_{13}	1.80E07	9.75E06	4.25E06	5.10E07	1.89E07	9.93E06	4.91E06	4.85E07	8.34E09	5.93E09	5.78E08	2.08E10	2.00E07	1.15E07	6.51E06	5.17E07
f_{14}	1.35E05	1.12E05	1.39E04	3.84E05	2.76E05	2.74E05	1.80E04	1.17E06	2.13E06	3.05E06	8.41E04	1.29E07	1.58E05	1.12E05	7.61E03	4.33E05
f_{15}	1.37E06	1.13E06	2.09E05	6.76E06	1.13E06	8.47E05	1.78E05	3.53E06	1.32E08	2.26E08	1.84E06	1.17E09	1.30E06	7.37E05	1.49E05	3.83E06
f_{16}	3.31E03	2.73E02	2.72E03	3.80E03	3.27E03	2.07E02	2.72E03	3.60E03	7.52E03	1.84E03	4.95E03	1.43E04	3.18E03	2.39E02	2.72E03	3.55E03
f_{17}	2.18E03	1.26E02	2.00E03	2.49E03	2.21E03	1.62E02	1.92E03	2.55E03	4.36E03	1.19E03	2.66E03	8.19E03	2.18E03	1.18E02	1.96E03	2.44E03
f_{18}	6.62E05	4.75E05	1.48E05	1.98E06	6.19E05	3.83E05	1.30E05	1.84E06	1.04E08	1.28E08	5.71E05	5.38E08	8.79E05	8.66E05	1.63E05	4.66E06
f_{19}	3.35E06	1.96E06	8.50E05	8.56E06	3.42E06	2.43E06	7.65E05	1.06E07	1.03E09	1.13E09	1.59E06	4.05E09	3.54E06	2.09E06	5.62E05	9.40E06
f_{20}	2.58E03	1.14E02	2.38E03	2.84E03	2.53E03	1.20E02	2.35E03	2.80E03	3.07E03	1.84E02	2.67E03	3.39E03	2.52E03	1.15E02	2.36E03	2.86E03
f_{21}	2.52E03	4.76E01	2.33E03	2.57E03	2.52E03	3.97E01	2.38E03	2.58E03	2.88E03	4.66E01	2.72E03	2.96E03	2.53E03	4.14E01	2.34E03	2.58E03
f_{22}	3.19E03	4.25E02	2.68E03	4.90E03	3.41E03	4.66E02	2.76E03	4.82E03	1.03E04	3.44E02	9.15E03	1.08E04	3.42E03	4.51E02	2.77E03	5.13E03
f_{23}	3.01E03	6.04E01	2.81E03	3.09E03	3.09E03	4.45E01	3.02E03	3.20E03	3.80E03	1.27E02	3.50E03	4.01E03	3.04E03	4.73E01	2.94E03	3.14E03
f_{24}	3.13E03	2.90E01	3.07E03	3.19E03	3.19E03	4.26E01	3.13E03	3.30E03	4.06E03	1.49E02	3.71E03	4.28E03	3.16E03	4.85E01	3.08E03	3.27E03
f_{25}	3.15E03	6.21E01	3.05E03	3.32E03	3.14E03	4.54E01	3.05E03	3.24E03	1.55E04	3.67E03	9.10E03	2.70E04	3.13E03	4.83E01	3.04E03	3.24E03
f_{26}	6.03E03	7.90E02	4.48E03	7.34E03	6.00E03	9.04E01	4.29E03	7.69E03	1.60E04	1.62E03	1.26E04	1.87E04	5.83E03	7.22E02	4.26E03	7.23E03
f_{27}	3.47E03	4.33E01	3.39E03	3.55E03	3.61E03	5.43E01	3.50E03	3.74E03	3.20E03	6.16E-5	3.20E03	3.20E03	3.20E03	1.49E-4	3.20E03	3.20E03
f_{28}	3.67E03	1.10E02	3.44E03	3.89E03	3.75E03	9.27E01	3.49E03	3.91E03	3.30E03	7.85E-5	3.30E03	3.30E03	3.30E03	1.28E-4	3.30E03	3.30E03
f_{29}	4.47E03	1.52E02	4.20E03	4.74E03	4.54E03	1.88E02	4.22E03	4.86E03	2.05E04	2.68E04	5.35E03	1.20E05	4.55E03	1.94E02	3.88E03	4.95E03
f_{30}	1.14E07	5.08E06	5.78E06	2.47E07	9.77E06	5.23E06	2.36E06	2.61E07	1.67E09	1.06E09	1.13E08	4.50E09	1.01E07	4.51E06	4.26E06	1.98E07

6 Conclusion

In this original study, the impact of various border strategies on the performance of the firefly algorithm is tested. The topic is actual due to the increasing variety and complexity of optimization problems. As a benchmark for the performance comparisons, the CEC 2017 set was used. It represents the most recent collection of artificial optimization problems that vary in terms of modality and other characteristics of the fitness landscape.

It may be concluded, that according to statistical data, the hard borders and the reflection boundary strategy seem to be favorable over the other two (random with slightly worse performance and hypersphere).

However, it seems that in the most cases, the border strategy does not have a significant impact on the performance of the method (especially in lower dimensions). The hypersphere boundary model stands out, mostly in the negative way. Despite the fact that remaining strategies behave almost identical on most problems, the most favorable strategies could be either the reflection or random position. These two strategies are often picked also for PSO [8].

Despite that, the results of this study are useful as an empirical study for researchers dealing with firefly algorithm. This research will continue in the future with exploring the performance of firefly algorithm with different boundary strategies on different fitness landscape models and real-world problems, especially with a focus on the algorithm setup to achieve the best performance.

Acknowledgements This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO), and Action IC406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet).

References

1. X. S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, UK, (2008).
2. X. S. Yang, Firefly algorithms for multimodal optimization, Proc. 5^{th} Symposium on Stochastic Algorithms, Foundations and Applications, Lecture Notes in Computer Science, 5792: 169-178 (2009).
3. Oosumi R, Tamura K, Yasuda K. Novel single-objective optimization problem and Firefly Algorithm-based optimization method. 9-12 Oct. 2016; IEEE; 2016.
4. Yang X. Multiobjective firefly algorithm for continuous optimization. Engineering with Computers. 2013; 29 (2):175-84.

5. X. S. Yang, Firefly algorithm, Lévy flights and global optimization, in: Research an Development in Intelligent Systems XXVI (Eds M. Bramer, R. Ellis, M. Petridis), Springer London, pp. 209-218 (2010).
6. Coelho LdS, Mariani VC. Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Computers & Mathematics with Applications. 2012 October 1; 64(8):2371-82.
7. Helwig S, Branke J, Mostaghim S M. Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization. TEVC. 2013; 17(2):259-71.
8. Kadavy T, Pluhacek M, Viktorin A, Senkerik R. Comparing Border Strategies for Roaming Particles on Singlea nd Multi-swarm PSO. In: Artificial Intelligence Trends in Intelligent Systems: Proceedings of the 6^{th} Computer Science On-line Conference 2017 (CSOC2017), Vol1. Cham: Springer International Publishing; 2017. p. 528-36.
9. N. H. Awad, et al. Problem Definitions and Evaluation Criteria for CEC 2017 Special Ses-sion and Competition on Single-Objective Real-Parameter Numerical Optimization, 2016.
10. Kadavy T, Pluhacek M, Viktorin A, Senkerik R. Hypersphere Universe Boundary Method Comparison on HCLPSO and PSO. In Hybrid Artificial Intelligent Systems: 12 th International Conference, HAIS 2017, La Rioja, Spain, June 21-23, 2017, Proceedings. Cham: Springer International Publishing; 2017. p. 173-82.
11. Barthelemy, P.: Bertolotti J., Wiersma D. S.,A Lévy flight for light, Nature, 453, 495-498 (2008).

Population Diversity Analysis for the Chaotic based Selection of Individuals in Differential Evolution

Roman Senkerik[1][0000−0002−5839−4263], Adam Viktorin[1][0000−0003−0861−0340], Michal Pluhacek[1][0000−0002−3692−2838], and Tomas Kadavy[1][0000−0002−3341−4336]

Tomas Bata University in Zlin, Faculty of Applied Informatics, T. G. Masaryka 5555, 760 01, Zlin Czech Republic
senkerik@utb.cz, aviktorin@utb.cz, pluhacek@utb.cz, kadavy@utb.cz

Abstract. This research deals with the modern and popular hybridization of chaotic dynamics and evolutionary computation. It is aimed at the influence of chaotic sequences on the population diversity as well as the algorithm performance of the simple parameter adaptive Differential Evolution (DE) strategy: jDE. Experiments are focused on the extensive investigation of the different randomization schemes for the selection of individuals in DE algorithm driven by the nine different two-dimensional discrete chaotic systems, as the chaotic pseudo-random number generators. The population diversity and jDE convergence are recorded on the 15 test functions from the CEC 2015 benchmark.

Keywords: Differential Evolution, Complex dynamics, Deterministic chaos, Population diversity, Chaotic map

1 Introduction

This research deals with the mutual intersection of the two computational intelligence fields, which are the complex sequencing and dynamics given by the selected chaotic systems, and evolutionary computation techniques (ECT's).

Together with this persistent development in above-mentioned mainstream research topics, the popularity of hybridizing of chaos and metaheuristic algorithms is growing every year. Recent research in chaotic approach for metaheuristics uses various chaotic maps in the place of pseudo-random number generators (PRNG).

The initial concept of embedding chaotic dynamics into the evolutionary/swarm algorithms as chaotic pseudo-random number generator (CPRNG) is given in [1]. Firstly, the Particle Swarm Optimization (PSO) algorithm with elements of chaos was introduced as CPSO [2], followed by the introduction of chaos embedded Differential evolution (DE) [3], PSO with inertia weigh strategy [4], and PSO with an ensemble of chaotic systems [5]. Recently the chaos driven heuristic concept has been utilized in several swarm-based algorithms like ABC algorithm [6], Firefly [7] and other metaheuristic algorithms [8] – [11], as well as many applications with DE [12].

The unconventional chaos-based approach is tightly connected with the importance of randomization within heuristics as compensation of a limited amount of search moves as stated in the survey paper [13]. This idea has been carried out in subsequent studies describing different techniques to modify the randomization process [14], [15]

and especially in [16], where the sampling of the points is tested from modified distribution. The importance and influence of randomization operations were also profoundly experimentally tested in simple control parameter adjustment DE strategy [17].

The focus of this research is the deeper insight into the population dynamics of the selected DE strategy (jDE) [18] when the directly embedded CPRNG is driving the indices selection. Currently, DE [19] - [21] is a well-known evolutionary computation technique for continuous optimization purposes solving many difficult and complex optimization problems. Many DE variants have been recently developed with the emphasis on control parameters self-adaptivity. DE has been modified and extended several times using new proposals of versions, and the performances of different DE variants have been widely studied and compared with other ECTs. Over recent decades, DE has won most of the evolutionary algorithm competitions in the leading scientific conferences [22] – [26], as well as being applied to several applications.

The organization of this paper is following: Firstly, the motivation for this research is proposed. The next sections are focused on the description of the concept of chaos driven jDE, and the experiment background. Results and conclusion follow afterward.

2 Motivation and Related Research

Recently, chaos with its properties like ergodicity, stochasticity, self-similarity, and density of periodic orbits became very popular and modern tool for improving the performance of various ECTs. Nevertheless, the questions remain, as to why it works, why it may be beneficial to use the chaotic sequences for pseudo-random numbers driving the selection, mutation, crossover or other processes in particular heuristics.

This research is an extension and continuation of the previous successful experiment with the single/multi-chaos driven PSO [5] and jDE [27], where the positive influence of hidden complex dynamics for the heuristic performance has been experimentally shown. This research is also a follow up to previous initial experiments with different sampling rates applied to the chaotic sequences resulting in keeping, partially/fully removing of traces of chaos [28].

The motivation and the novelty of the research are given by the investigating the influence of chaotic sequences to the population diversity, connected with the algorithm performance of the basic control parameter adjustment DE strategy: jDE. This strategy was selected as a compromise between original simple DE and the most recent Success-History based Adaptive Differential Evolution (SHADE) variants [26], where the influence of chaotic dynamics may be suppressed by the complex adaptive process and operations with the archive.

3 Differential Evolution

This section describes the basics of original DE and jDE strategies. The original DE [19] has four static control parameters – a number of generations G, population size NP, scaling factor F and crossover rate CR. In the evolutionary process of DE, these four parameters remain unchanged and depend on the initial user setting. jDE algorithm, on the other hand, adapts the F and CR parameters during the evolution. The mutation

strategy for jDE is adapted from the original DE. The concept of essential operations in jDE algorithm is shown in following sections, for a detailed description on either original DE refer to [19] or for jDE see [18].

3.1 jDE

In this research, we have used jDE with original DE "rand/1/bin" (1) mutation strategy and binomial crossover (2).

Mutation Strategies and Parent Selection The parent indices (vectors) are selected either by standard PRNG with uniform distribution or by CPRNG in case of chaotic versions. Mutation strategy "rand/1/bin" uses three random parent vectors with indexes *r1*, *r2* and *r3*, where $r1 = U[1, NP]$, $r2 = U[1, NP]$, $r3 = U[1, NP]$ and $r1 \neq r2 \neq r3$. Mutated vector $\boldsymbol{v}_{i,G}$ is obtained from three different vectors $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, $\boldsymbol{x}_{r3}$ from current generation G with the help of scaling factor F_i as follows:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r1,G} + F_i \left(\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}\right) \tag{1}$$

Crossover and Selection The trial vector $\boldsymbol{u}_{i,G}$ which is compared with original vector $\boldsymbol{x}_{i,G}$ is completed by crossover operation (2). CR_i value in jDE algorithm is not static.

$$\boldsymbol{u}_{j,i,G} = \begin{cases} \boldsymbol{v}_{j,i,G} & \text{if } U[0,1] \leq CR_i \text{ or } j = j_{rand} \\ \boldsymbol{x}_{j,i,G} & \text{otherwise} \end{cases} \tag{2}$$

Where j_{rand} is a randomly selected index of a feature, which has to be updated ($j_{rand} = U[1, D]$), D is the dimensionality of the problem.

The vector which will be placed into the next generation $G+1$ is selected by elitism. When the objective function value of the trial vector $\boldsymbol{u}_{i,G}$ is better than that of the original vector $\boldsymbol{x}_{i,G}$, the trial vector will be selected for the next population. Otherwise, the original will survive. (3).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} \tag{3}$$

3.2 Parameter adjustment in jDE

The generated ensemble of two control parameters F_i and CR_i is assigned to each i-th individual of the population and survives with the solution if an individual is transferred to the new generation. The initialization of values of F and CR is designed to be either fully random with uniform distribution for each individual in the population or can be set according to the recommended values in the literature. If the newly generated solution is not successful, i.e., the trial vector has worse fitness than the compared original active individual; the new (possibly) reinitialized control parameters values disappear together with not successful solution. The both aforementioned DE control parameters may be randomly mutated with predefined probabilities τ_1 and τ_2. If the mutation condition happens, a new random value of $CR \in [0, 1]$ is generated, possibly also a new

value of F which is mutated in $[F_l , F_u]$. These new control parameters are after that stored in the new population. Input parameters are typically set to $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = 0.1$, and $\tau_2 = 0.1$ as originally given in [18], [20].

4 Chaotic Systems for CPRNGs

Following nine well known and frequently utilized discrete dissipative chaotic maps were used as the CPRNGs for jDE. With the settings as in Table 1, systems exhibit typical chaotic behavior [29].

Table 1. Definition of chaotic systems used as CPRNGs

Chaotic system	Notation	Parameters values
Arnold Cat map	$X_{n+1} = X_n + Y_n (mod 1)$ $Y_{n+1} = X_n + kY_n (mod 1)$	k = 2.0
Burgers map	$X_{n+1} = aX_n - Y_n^2$ $Y_{n+1} = bY_n + X_n Y_n$	a = 0.75 and b = 1.75
Delayed Logistic	$X_{n+1} = AX_n (1 - Y_n)$ $Y_{n+1} = X_n$	A = 2.27
Dissipative Standard map	$X_{n+1} = X_n + Y_{n+1} (mod 2\pi)$ $Y_{n+1} = bY_n + k \sin X_n (mod 2\pi)$	b = 0.1 and k = 8.8
Hénon map	$X_{n+1} = a - x_n^2 + by_n$ $Y_{n+1} = x_n$	a = 1.4 and b = 0.3
Ikeda map	$X_{n+1} = \gamma + \mu(X_n \cos\phi + Y_n \sin\phi)$ $Y_{n+1} = \mu(X_n \sin\phi + Y_n \cos\phi)$ $\phi = \beta - \alpha / \left(1 + X_n^2 + Y_n^2\right)$	α = 6, β = 0.4, γ = 1 and μ = 0.9
Lozi Map	$X_{n+1} = 1 - a\lvert X_n \rvert + bY_n$ $Y_{n+1} = X_n$	a = 1.7 and b = 0.5
Sinai map	$X_{n+1} = X_n + Y_n + \delta \cos 2\pi Y_n (mod 1)$ $Y_{n+1} = X_n + 2Y_n (mod 1)$	δ = 0.1
Tinkerbell map	$X_{n+1} = X_n^2 - Y_n^2 + aX_n + bY_n$ $Y_{n+1} = 2X_n Y_n + cX_n + dY_n$	a = 0.9, b = -0.6, c = 2 and d = 0.5

5 The Concept of ChaosDE with Discrete Chaotic System as driving CPRNG

The general idea of CPRNG is to replace the default PRNG with the chaotic system. As the chaotic system is a set of equations with a static start position, we created a random start position of the system, to have different start position for different experiments. Thus we are utilizing the typical feature of chaotic systems, which is extreme sensitivity to the initial conditions, popularly known as "butterfly effect." This random position is initialized with the default PRNG, as a one-off randomizer. Once the start position of

the chaotic system has been obtained, the system generates the next sequence using its current position. Used approach is based on the following definition (4):

$$rndreal = \mod\ (\mathrm{abs}\,(rndChaos)\,, 1.0) \tag{4}$$

5.1 Experiment design

For the population diversity analysis and performance comparisons in this research, the CEC 15 benchmark was selected. The dimension D was set to 10. Every instance was repeated 51 times with the maximum number of objective function evaluations set to 100 000 (10,000 × D). The convergence and population diversity were recorded for all tested algorithm – original jDE and nine versions of C_jDE with different CPRNGs. All algorithms used the same set of control parameters: population size $NP = 50$ and initial settings $F = 0.5$, $CR = 0.8$. Experiments were performed in the environment of *Java*; jDE, therefore, has used the built-in *Java linear congruential pseudorandom number generator* representing traditional pseudorandom number generator in comparisons.
The Population Diversity (PD) measure used in this paper was described in [30] and is based on the sum of deviations (6) of individual's components from their corresponding means (5).

$$\overline{x_j} = \frac{1}{NP}\sum_{i=1}^{NP} x_{ij} \tag{5}$$

$$PD = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}\left(x_{ij} - \overline{x_j}\right)^2} \tag{6}$$

Where i is the population member iterator and j is the vector component iterator.

6 Results

Statistical results for the comparisons are shown in comprehensive Tables 2 and 3. Table 2 shows the mean results, with the highlighting based on the *Wilcoxon sum-rank test* with the significance level of 0.05; performed for each pair of original jDE and C_jDE. Ranking of the algorithms given in Figure 1 was evaluated based on the Friedman test with Nemenyi post hoc test. Figures 2 – 5 depict the graphical comparisons of the convergence plots and corresponding population diversity plots provided for the selected five benchmark functions. The Fig. 6 shows the detailed comparisons of population diversity plots (with confidence intervals) for the selected pair of jDE and C_jDE where the performance is different. The results discussion is in the next section.

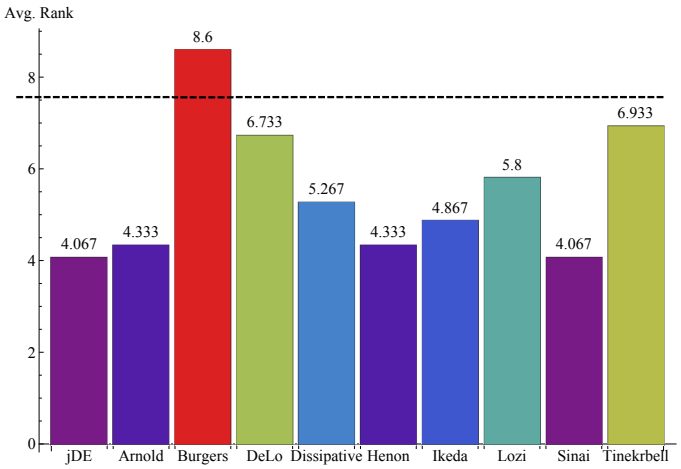


Fig. 1. Ranking of the all algorithms based on the 51 runs and 15 functions of CEC2015 benchmark in 10D. Dashed line represents the Nemenyi Critical Distance.

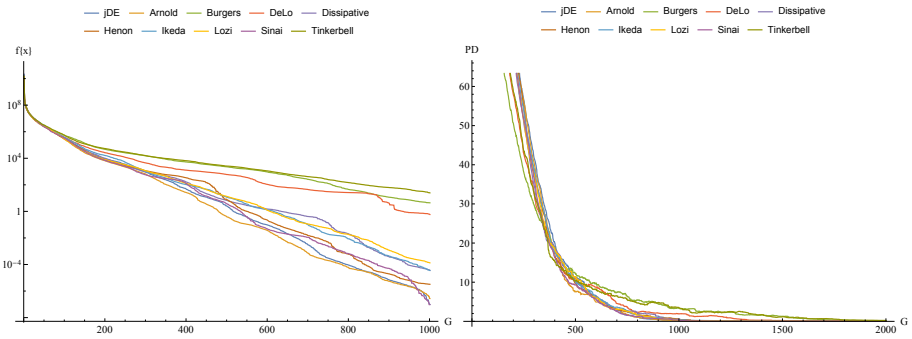


Fig. 2. Convergence plot (left) and population diversity plot (right) of CEC2015 *f1* in 10*D*.

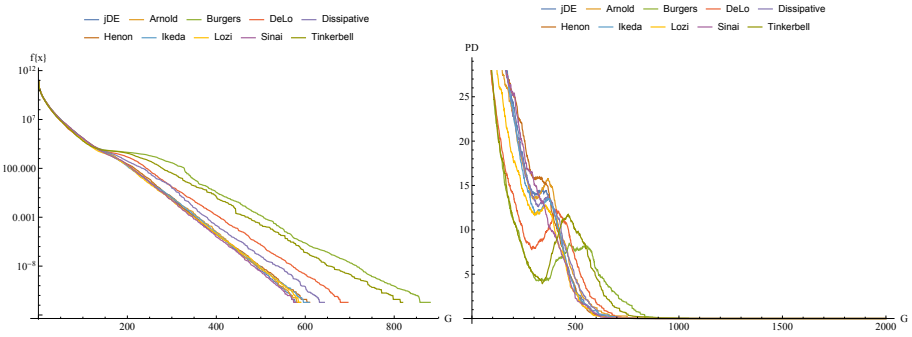


Fig. 3. Convergence plot (left) and population diversity plot (right) of CEC2015 *f2* in 10*D*.

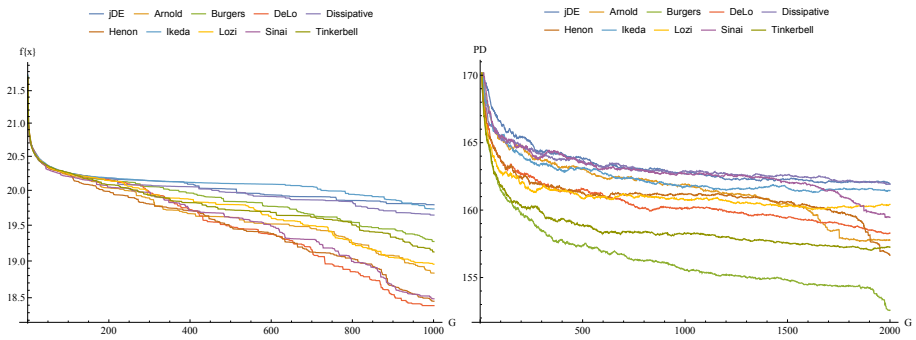


Fig. 4. Convergence plot (left) and population diversity plot (right) of CEC2015 *f3* in 10*D*.

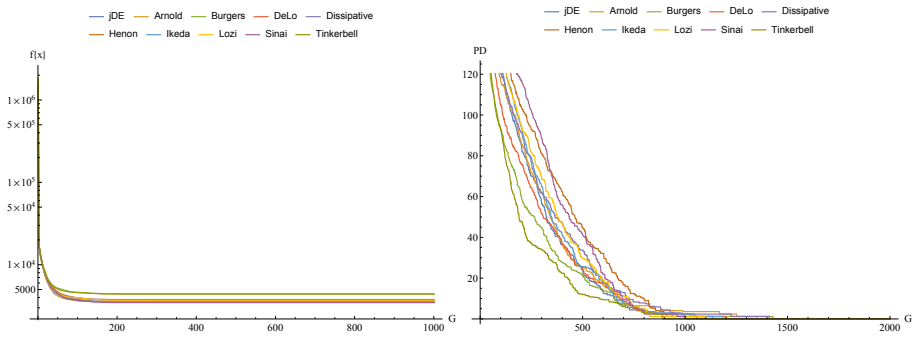


Fig. 5. Convergence plot (left) and population diversity plot (right) of CEC2015 *f14* in 10*D*.

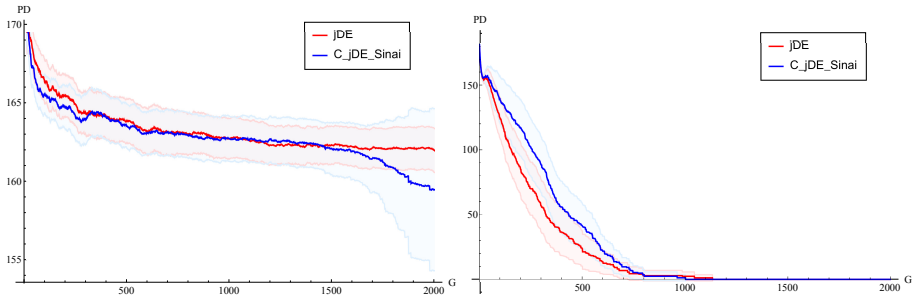


Fig. 6. Detailed population diversity plots for the selected pair of jDE and C_jDE driven by Sinai chaotic system (left *f3*) and (right *f14*), CEC2015 in 10*D*.

Table 2. Results comparisons for the mean results of jDE and C_jDE; CEC 2015 Benchmark set, 10*D*, 51 runs

system\f	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
jDE	9.79E-08	**0.**	19.7916	4.1515	**126.70**	6.1719	0.1724	**0.2298**	100.207	**218.29**	**101.59**	101.839	27.96	3613.67	**100.**
Arnold C_jDE	2.74E-07	**0.**	18.8335	4.2794	145.24	9.4381	0.1567	0.5323	100.209	219.42	107.41	101.839	27.62	3544.63	**100.**
Burgers C_jDE	*4.4106* †	**0.**	19.2706	4.0942	*173.63* †	*54.0442* †	0.2997	*10.2154* †	*100.226* †	*243.52* †	*148.95* †	101.995	28.56	*4406.91* †	**100.**
DeLo C_jDE	*0.5832* †	**0.**	**18.3945**	**3.8806**	*166.44* †	*33.1935* †	0.1855	1.9563	**100.204**	*233.58* †	*154.45* †	101.884	28.68	3777.18	**100.**
Dissipative C_jDE	*3.62E-05* †	**0.**	19.6439	4.0253	139.11	9.7203	0.1662	0.5663	100.205	219.95	136.56	101.815	27.91	3756.54	**100.**
Henon C_jDE	*3.22E-06* †	**0.**	18.4531	4.1839	126.96	**2.4113**	0.1728	*0.9548* †	*100.218* †	219.42	124.95	101.851	**27.53**	3488.54	**100.**
Ikeda C_jDE	*3.55E-05* †	**0.**	19.7317	3.9634	137.59	6.1798	0.1592	1.5255	100.210	219.03	*154.26* †	101.820	27.71	3557.67	**100.**
Lozi C_jDE	*1.32E-04* †	**0.**	18.9552	4.1586	134.54	6.9613	**0.1561**	1.3641	100.215	219.59	142.43	101.859	28.09	3688.73	**100.**
Sinai C_jDE	**9.54E-08**	**0.**	*18.4859* ‡	3.9828	127.20	10.3090	0.1569	1.3596	100.208	219.34	130.81	101.880	27.73	**3466.09**	**100.**
Tinkerbell C_jDE	*24.6343* †	**0.**	19.1244	3.9863	148.39	*39.4389* †	0.2460	*8.5815* †	100.215	*231.44* †	*125.99* †	**101.798**	28.06	*4417.27* †	**100.**

The bold values in Table 2 depict the best-obtained results (based on the mean values); italic values are considered to be significantly different (according to the Wilcoxon sum-rank test with the significance level of 0.05; performed for each pair of original jDE and C_jDE; † - performance of C_JDE was significantly worse, ‡- significantly better).

Table 3. The best (minimum found) results for jDE and C_jDE; CEC 2015 Benchmark set, 10*D*, 51 runs

system\f	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
jDE	9.79E-08	**0.**	19.7916	4.1515	126.703	6.1719	0.1724	0.2298	100.207	218.286	101.59	101.839	27.961	3613.67	**100.**	2
Arnold C_jDE	**0**.	**0.**	**2.50E-09**	1.2559	37.1301	**0**.	0.0369	7.11E-07	100.152	**216.537**	0.4638	101.245	24.006	2935.54	**100.**	6
Burgers C_jDE	7.01E-04	**0.**	0.1154	**0.**	**10.3074**	2.14E-03	0.0197	6.37E-06	100.154	216.556	1.6532	101.308	**21.749**	2935.54	**100.**	5
DeLo C_jDE	4.92E-07	**0.**	5.5139	**0.**	25.7866	0.2081	0.0310	3.69E-06	**100.121**	**216.537**	1.0906	100.994	23.384	2935.54	**100.**	5
Dissipative C_jDE	**0.**	**0.**	6.5179	1.1008	37.3398	**0**.	0.0464	1.69E-05	100.108	**216.537**	0.7982	100.948	23.818	2935.54	**100.**	5
Henon C_jDE	**0.**	**0.**	2.39E-04	1.2774	14.2125	**0.**	0.0549	9.69E-05	100.134	**216.537**	0.9954	100.928	23.614	**100.**	**100.**	6
Ikeda C_jDE	**0.**	**0.**	11.9019	1.8199	40.4215	**0.**	0.0381	3.29E-06	**100.121**	**216.537**	0.8272	100.801	24.465	**100.**	**100.**	7
Lozi C_jDE	**0.**	**0.**	6.4830	1.5781	33.1746	**0.**	0.0497	3.38E-04	100.129	**216.537**	**0.4429**	101.238	25.882	2935.54	**100.**	6
Sinai C_jDE	**0.**	**0.**	3.24E-04	1.1066	23.3597	**0.**	0.0333	**3.38E-07**	100.126	**216.537**	0.5749	101.067	23.732	2935.54	**100.**	6
Tinkerbell C_jDE	3.15E-04	**0.**	7.3022	1.0075	33.8032	**0.**	**0.0093**	1.30E-04	100.148	216.539	1.1523	**100.409**	23.073	**100.**	**100.**	6

The bold values in Table 3 depict the best-obtained results (based on the min. values).

7 Conclusions

The primary aim of this original work is to provide a more in-depth insight into the inner dynamics of indices selection in DE. The focus is to experimentally investigate the influence of different types of unconventional non-random (chaotic) sequences to the population diversity as well as to the performance of the simple parameter adjustment DE strategy, which is jDE. The findings can be summarized as:

- Obtained graphical comparisons and data in Tables 2 and 3 support the claim that jDE is sensitive to the chaotic dynamics driving the selection (mutation) process through CPRNG. At the same time, it is clear that (selection of) the best CPRNGs are problem-dependent. By using the CPRNG inside the heuristic, its performance is (significantly) different: either better or worse against other compared versions.
- The performance comparisons presented in Tables 2 and 3 reveal the fact that only in one case the performance of C_jDE is statistically significantly better (*f3* and Sinai map). Mostly the performance of compared pairs of jDE and C_jDE is similar, or in some cases, the chaotic versions performed significantly worse. Such a worse performance was repeatedly observed for two chaotic maps: Burgers and Tinkerbell. On the other hand, these two maps usually secured robust progress towards function extreme (local) followed by premature population stagnation phase, thus repeatedly secured finding of minimum values. Overall, C_jDE versions seem to be very effective regarding finding min. values of the objective function (See Table 3).
- The population diversity plots in Figures 2 - 5 supports the above-mentioned facts. It is possible to identify 3 groups of population diversity behavior in comparison with original j_DE: less decreasing (Sinai, Henon, Ikeda maps), more decreasing (Lozi, Arnold, Dissipative maps) and significantly more decreasing (Delayed Logistic, Tinkerbell, Burgers maps).
- The selected paired diversity plots in Fig 6 show that the diversity of the population is maintained higher for a longer period. Therefore the exploration phase supported by Sinai map based CPRNG is longer. This in return is beneficial for the result of the optimization.
- The population diversity analysis supports the theory, that unique features of the chaos transformed into the sequencing of CPRNG values may create the subpopulations (or inner neighborhood selection schemes, i.e., lower population diversity). Thus the metaheuristic can benefit from the searching within those sub-populations and quasi-periodic exchanges of information between individuals (see Fig 3 for the sudden increase of diversity – the new search region was explored and attracted some (group) of individuals). However, lot of analyses and different scenarios (dimensional settings, etc.) are required in the future.

The research of randomization issues and insights into the inner dynamic of metaheuristic algorithms was many times addressed as essential and beneficial. The results presented here support the approach for multi-chaotic generators [31] or ensemble systems, where we can profit from the combined/selective population diversity (i.e. exploration/exploitation) tendencies, sequencing-based either stronger or moderate

progress towards the function extreme, all given by the smart combination of multi-randomization schemes.

Acknowledgements This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO), and Action IC406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet).

References

1. Caponetto R, Fortuna L, Fazzino S, Xibilia MG (2003) Chaotic sequences to improve the performance of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7 (3): 289-304.
2. Coelho LdS, Mariani VC (2009) A novel chaotic particle swarm optimization approach using Hénon map and implicit filtering local search for economic load dispatch. Chaos, Solitons & Fractals 39 (2): 510-518.
3. Davendra, D., Zelinka, I., Senkerik, R.: Chaos driven evolutionary algorithms for the task of PID control. Computers & Mathematics with Applications 60 (4): 1088-1104 (2010).
4. Pluhacek M, Senkerik R, Davendra D, Kominkova Oplatkova Z, Zelinka I (2013) On the behavior and performance of chaos driven PSO algorithm with inertia weight. Computers & Mathematics with Applications 66 (2):122-134.
5. Pluhacek, M., Senkerik, R., Davendra, D.: Chaos particle swarm optimization with Eensemble of chaotic systems. Swarm and Evolutionary Computation 25: 29-35 (2015).
6. Metlicka, M., Davendra, D.: Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems. Swarm and Evolutionary Computation 25: 15-28 (2015).
7. Gandomi, A. H., Yang, X. S., Talatahari, S., & Alavi, A. H. (2013). Firefly algorithm with chaos. Communications in Nonlinear Science and Numerical Simulation, 18 (1): 89-98.
8. Wang, G. G., Guo, L., Gandomi, A. H., Hao, G. S., & Wang, H. (2014). Chaotic krill herd algorithm. Information Sciences, 274, 17-34.
9. Zhang, C., Cui, G., & Peng, F. (2016). A novel hybrid chaotic ant swarm algorithm for heat exchanger networks synthesis. Applied Thermal Engineering, 104, 707-719.
10. Jordehi, A. R. (2015). Chaotic bat swarm optimisation (CBSO). Applied Soft Computing, 26, 523-530.
11. Wang, G. G., Deb, S., Gandomi, A. H., Zhang, Z., & Alavi, A. H. (2016). Chaotic cuckoo search. Soft Computing, 20 (9): 3349-3362.
12. Coelho, L.d.S., Ayala, H.V.H., Mariani, V.C. (2014). A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. Applied Mathematics and Computation 234 (0): 452-459.
13. Neri, F., Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33 (1-2): 61-106.
14. Weber, M., Neri, F., Tirronen, V. (2011). A study on scale factor in distributed differential evolution. Information Sciences 181(12): 2488-2511.

15. Neri, F., Iacca, G., Mininno, E. (2011). Disturbed Exploitation compact Differential Evolution for limited memory optimization problems. Information Sciences 181 (12), 2469-2487.
16. Iacca, G., Caraffini, F., Neri, F. (2012). Compact Differential Evolution Light: High Performance Despite Limited Memory Requirement and Modest Computational Overhead. J. Comput. Sci. Technol. 27 (5): 1056-1076.
17. Zamuda, A., Brest, J. (2015) Self-adaptive control parameters? randomization frequency and propagations in differential evolution. Swarm and Evolutionary Computation 25: 72-99.
18. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V. (2006). Self- Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, IEEE Transactions on Evolutionary Computation, 10 (6): 646–657.
19. Price, K. V., Storn, R. M.,Lampinen, J. A. (2005). Differential Evolution: A Practical Approach to Global Optimization, ser. Natural Computing Series. Berlin, Germany: Springer-Verlag.
20. Neri, F., Tirronen,V. (2010). Recent Advances in Differential Evolution: A Survey and Experimental Analysis, Artificial Intelligence Review, 33 (1–2), 61–106.
21. Das, S., Mullick, S. S., Suganthan, P. (2016). Recent advances in differential evolution – An updated survey, Swarm and Evolutionary Computation, 27, 1–30.
22. Das, S., Abraham, A., Chakraborty, U., Konar, A. (2009). Differential Evolution Using a Neighborhood-based Mutation Operator, IEEE Transactions on Evolutionary Computation, 13 (3): 526–553.
23. Mininno, E., Neri, F., Cupertino, F., Naso, D. (2011). Compact Differential Evolution, IEEE Transactions on Evolutionary Computation, 15 (1), 32–54.
24. Mallipeddi, R., Suganthan, P. N., Pan, Q. K., Tasgetiren, M. F. (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing, 11 (2), 1679– 1696.
25. Brest, J., Korosec, P., Silc, J., Zamuda, A., Boskovic, B., Maucec, M. S. (2013) Differential evolution and differential ant-stigmergy on dynamic optimisation problems, International Journal of Systems Science, 44 (4), 663–679.
26. Tanabe, R., Fukunaga, A. S. (2014). Improving the search performance of SHADE using linear population size reduction, in 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, 1658–1665.
27. Senkerik, R., Pluhacek, M., Zelinka, I., Viktorin, A., Oplatkova, Z. K. (2016, June). Hybridization of Multi-chaotic Dynamics and Adaptive Control Parameter Adjusting jDE Strategy. In International Conference on Soft Computing-MENDEL). Springer, 77-87.
28. Senkerik R, Pluhacek M, Zelinka I, Davendra D, Janostik J (2016) Preliminary Study on the Randomization and Sequencing for the Chaos Embedded Heuristic. In: Abraham A, Wegrzyn-Wolska K, Hassanien EA, Snasel V, Alimi MA (eds) Proceedings of the Second International Afro-European Conference for Industrial Advancement AECIA 2015. Springer International Publishing, 591-601.
29. Sprott JC (2003) Chaos and Time-Series Analysis. Oxford University Press.
30. Polakova, R., Tvrdik, J., Bujok, P., Matousek, R. (2016). Population-size adaptation through diversity-control mechanism for differential evolution. In MENDEL, 22th International Conference on Soft Computing, 49-56.
31. Viktorin, A., Pluhacek, M., Senkerik, R. (2016, July). Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set. In Evolutionary Computation (CEC), 2016 IEEE Congress on. 4797-4803.

Tuning Multi-Objective Optimization Algorithms for the Integration and Testing Order Problem

Miha Ravber[1], Matej Črepinšek[1], Marjan Mernik[1], and Tomaž Kosar[1]

Faculty of Electrical Engineering and Computer Science,
University of Maribor, Slovenia

Abstract. Multi-Objective Evolutionary Algorithms (MOEAs) are one of the most used search techniques in Search-Based Software Engineering (SBSE). However, MOEAs have many control parameters which must be configured for the problem at hand. This can be a very challenging task by itself. To make matters worse, in Multi-Objective Optimization (MOO) different aspects of quality of the obtained Pareto front need to be taken in to account. A novel method called MOCRS-Tuning is proposed to address this problem. MOCRS-Tuning is a meta-evolutionary algorithm which uses a chess rating system with quality indicator ensemble. The chess rating system enables us to determine the performance of an MOEA on different problems easily. The ensemble of quality indicators ensures that different aspects of quality are considered. The tuning was carried out on five different MOEAs on the Integration and Test Order Problem (ITO). The experimental results show significant improvement after tuning of all five MOEAs used in the experiment.

Keywords: Multi-objective optimization, Evolutionary algorithms, Parameter tuning, Search-based software engineering, Class integration and testing order, Chess rating system.

1 Introduction

Search-Based Software Engineering (SBSE) is an approach where search-based optimization algorithms are used to solve problems in software engineering [1]. One of the many areas that SBSE tackles is software testing [2]. Software testing plays an important role in the software development life cycle, since it has a direct impact on the quality of the software. However, generating tests is a very difficult and costly task [3]. Since software testing is so complex, and exact solutions cannot be found in reasonable time using deterministic methods, it is no surprise that SBSE algorithms were applied in industrial cases [4,5]. One of the most popular methods used in SBSE are Multi-Objective Evolutionary Algorithms (MOEAs), which return a Pareto fronts. This enables the users to choose a solution with the best trade-off between different objectives. However, evolutionary algorithms have different control parameters. The choice of control parameters has a great impact on the performance of an evolutionary algorithm

[6]. Setting control parameters can be very challenging, and is known as a parameter tuning problem [7]. Tuning of algorithms is very important. It can find good control parameters which improve the algorithms performance. Also, with tuned algorithms we can perform a fair comparison [8]. Algorithms with default parameters perform well on benchmark problems, but this is usually not the case for real-world problems, since they are not studied in literature and we have little knowledge about them.

To tackle this problem, we propose Tuning with a Chess Rating System (CRSTuning) [8], adapted for Multi-Objective Optimization called MOCRS-Tuning. The tuning process is guided by a self-adaptive Differential Evolution (jDE) [9], which searches for optimal control parameter. By using a self-adaptive algorithm, we removed the additional parameters needed for the tuning process. The solutions in the population are evaluated with a Chess Rating System with a Quality Indicator Ensemble (CRS4MOEA/QIE) [10]. The Quality Indicator Ensemble ensures that the outcome of each candidate solution is evaluated with different Quality Indicators (QIs), making sure that different aspects of quality are taken into account [11]. We know by the No Free Lunch (NFL) theorem [12] that it is not possible to find optimal parameter settings, but this holds only if all possible search problems are considered. In our experiments, we limited ourselves to the Integration and Testing Order (ITO) problem [13], which has been shown that MOEAs can solve efficiently [14]. The ITO problem is concerned with the order in which software components are to be integrated and tested, such that the stubbing cost is minimised [15].

In our experiments, we applied the novel MOCRS-Tuning method to five different MOEAs. The tuning was conducted on 8 real-world object-oriented and aspect-oriented systems [4,13,14]. The comparison was conducted using the Evolutionary Algorithms Rating System (EARS) framework [16]. The EARS framework uses a chess rating system to rank and compare evolutionary algorithms. The results show significant improvement of all five MOEAs with tuned control parameters compared with the non-tuned (default) versions.

The remainder of the paper is organised as follows. A brief description of the ITO problem is given in Section 2. The chess rating system for evolutionary algorithms is described in Section 3. Section 4 describes the proposed tuning method. The execution of the experiment and results are presented in Section 5. Finally, the paper concludes in Section 6.

2 Integration and Testing Order

When performing a unit test in order to detect interaction problems between units, they need to be integrated and tested in order. If a unit is required by other units but is not yet available, it has to be emulated. An emulated unit is called a stub, and it imitates some or all functions of the actual unit [15]. A stub must be created for each unit that is not available during the integration process. Stubs are not desired for three reasons. First, stubs can be more complex than the code they simulate. Second, since they require understanding of the semantics of

the simulated functions their generation cannot be fully automated. Third, some stubs may be more error prone than their real counterparts [17]. Therefore, to reduce the stubbing cost, such a sequence must be determined that it minimises the stubbing cost [14]. This, however, is not a trivial task, and is known as the Integration and Testing Order problem [13]. Since different factors (objectives) influence the stubbing process, the problem should be treated as multi-objective [14]. This makes it very suitable to be solved by MOEAs [13]. In our experiments we used two objectives: Number of attributes and number of methods, which have to be emulated in the stub if the dependencies between two modules are to be broken. We used eight real systems in our experiments: MyBatis, AJHSQLDB (HyperSQL DataBase), BCEL (Byte Code Engineering Library), JHotDraw, HealthWatcher, JBoss, AJHotDraw, TollSystems. Information about the systems such as number of dependencies, classes, aspects, and Lines of Code (LOC) is given in Table 1.

Table 1. Details of the systems used for the ITO problem in the experiments.

Name	**Dependencies**	**Classes**	**Aspects**	**LOC**
AJHotDraw	1592	290	31	18586
AJHSQLDB	1338	276	15	68550
MyBatis	1271	331	-	23535
JHotDraw	809	197	-	20273
JBoss	367	150	-	8434
HealthWatcher	289	95	22	5479
BCEL	289	45	-	2999
TollSystems	188	53	24	2496

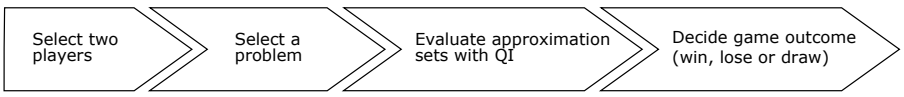


Fig. 1. A single game in CRS4MOEA/QIE.

3 Chess rating system for evolutionary algorithms

For comparing and evaluating MOEAs we used a novel method called Chess Rating System with a Quality Indicator Ensemble (CRS4MOEA/QIE) [10]. CRS4MOEA/QIE uses the Glicko-2 system [18] to rate and rank players. A chess rating system is used to estimate a chess player's skill level. Although it was initially intended to rank chess players, it can be applied to any competitor-versus-competitor game. In our case, players are MOEAs. Figure 1 shows a single game between two players (MOEAs). Each MOEA returns an approximation set for the given problem. The two approximation sets are evaluated with a QI from

the ensemble, and the outcome of the game is decided. In a tournament players play multiple games against all participating players for each given problem. A tournament can have multiple independent runs. The outcomes of the games are used to update each player's rating R and rating deviation RD [18]. Each unrated player has his rating set to 1500 and RD to 350 before the tournament starts. The rating represents a player's skill; the higher the rating, the higher the skill. If the player performs better than expected his rating increases, and decreases if they perform worse than expected. The rating deviation indicates how reliable a player's rating is. A small RD means a player plays often and has a reliable rating. In contrast, if the RD is high, his rating is unreliable. The chess rating system was used for comparison of MOEAs and for their evaluation in the tuning process.

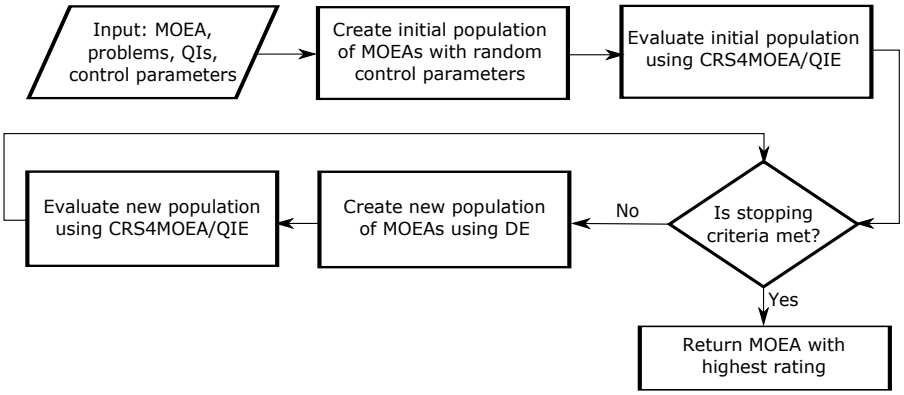


Fig. 2. MOCRS-Tuning flowchart.

4 MOCRS-Tunning method

Proposed method (MOCRS-Tunning) uses a meta-evolutionary approach in order to tune the control parameters of MOEAs. Figure 2 shows a simple flowchart of the tuning process. As input, we give it an MOEA, the problems for which it will be tuned, QIs for evaluation of results, and control parameters with their ranges to be tuned. First an initial population is created where the control parameters are generated randomly. The initial population is then evaluated using CRS4MOEA/QIE. In the tournament, multiple versions of the given MOEA with different control parameters are competing with each other on the given problems. At the end of the tournament, each version of MOEA receives its rating R, which reflects its performance (fitness). In the main loop, the tuning process takes place and is guided by jDE and CRS4MOEA/QIE. jDE is used to produce new solutions (MOEAs with different control parameters) and CRS4MOEA/QIE to evaluate the newly produced solutions. In order to evaluate the new solution it plays in a tournament with the old population. If the new

solution has a higher rating compared to its current version, it will be added to the new population, otherwise the current solution is added. The new population needs to be evaluated, since the rating of a player depends on its current opponents. When the stopping criteria is meet the MOEA with the highest rating in the population is returned.

5 Experiment

The experiment consists of three parts: Comparing MOEAs with their 'default' control parameters, tuning of MOEAs, and comparing MOEAs with tuned control parameters against MOEAs with default parameters. The experiment was performed on five MOEAs: $IBEA$ [19], $MOEAD$ [20], $NSGA-II$ [21], $PESA-II$ [22] and $SPEA2$ [23]. The Quality Indicator ensemble contained five different QIs in all experiments: IGD^+ [24], HV [25], $R2$ [26], MS [27] and $I_{\epsilon+}$ [28]. The diversity of QIs ensures that all the aspects of quality are covered [10]. In all experiments, MOEAs solved the ITO problem on eight previously mentioned systems for which the stopping criteria was set to 300,000 evaluations for each system. In the first and last parts we conducted a tournament for the comparison using CRS4MOEA/QIE incorporated in EARS. The tournament in the first and third parts contained the same problems and QIs as the tuning process. For the comparison of MOEAs before and after tuning, the number of independent runs in the tournament was set to 15. At the end of the tournament we plotted Rating Intervals (RI) of each MOEA using their rating and RD. Using RIs, we are 95% confident that the player's rating R is within an interval $[R-2RD, R+2RD]$. If the rating intervals of two MOEAs do not overlap, then they are significantly different, whereas, conversely, it is not necessarily true

5.1 Comparing MOEAs with default control parameters

Finding default parameters can be challenging, since they depend on the type of problem. The default values are commonly provided by the author. However they are usually available only for continuous types of problems, and rarely for combinatorial types such as the ITO problem. Therefore, we set the default values of control parameters for all MOEAs based on the source code of combinatorial operators found in the jMetal framework [29]. We limited the tuning process to three control parameters: Population size μ, crossover probability η_c and mutation probability η_m. Based on the jMetal framework, the default values of the parameters are 100 for μ, 1.0 for η_c and 0.2 for η_m. The results of the tournament are displayed in the form of Rating Intervals. The RD value of all participating MOEAs reached its minimum value (50) [30]. The Rating Intervals of all MOEAs with default control parameters are shown in Figure 3. The algorithms are ranked based on the rating, where the algorithm with the highest rating is first. From the results, we can observe that $NSGAII$ performed the best, and is significantly better (Rating Intervals do not overlap) than $MOEAD$. $PESAII$, $IBEA$ and $SPEA2$ have a very similar rating and, consequently, their intervals

overlap almost entirely. There is a high probability that the order of these three MOEAs would change if the tournament were repeated.

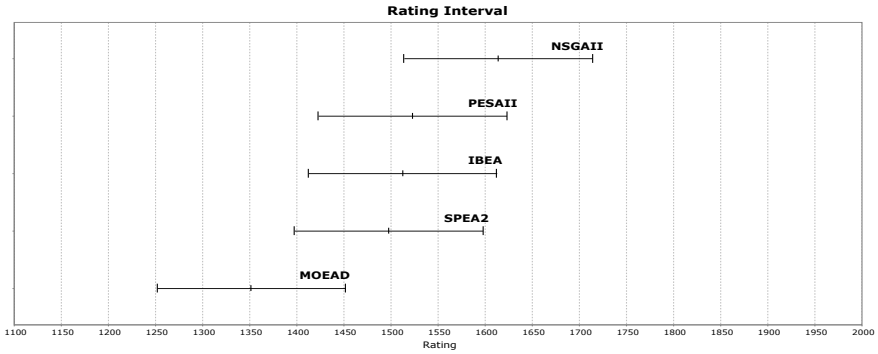


Fig. 3. 95% rating intervals of all MOEAs with default parameters.

Table 2. Control parameters of all five MOEAs after tuning.

	μ	η_c	η_m
$IBEA$	173	0.63	0.9
$MOEAD$	192	0.50	0.98
$NSGAII$	108	1.00	0.86
$PESAII$	110	0.37	1.0
$SPEA2$	190	0.74	0.87

5.2 Tuning MOEAs

In the tuning process we limited the search space of control parameters. For population size the lower bound is set to 10 and the upper bound to 200. Mutation and crossover probability have the same lower bound 0.1 and upper bound 1.0. The population size (number of MOEAs) for jDE was set to 20, and the stopping criteria was set to 20 generations. Table 2 shows the parameters of each MOEA after the tuning process. We can observe that different MOEAs have very different control parameters, except for mutation η_m. All MOEAs seem to prefer a higher mutation probability, meaning that higher exploitation is required for the given problems.

5.3 Comparing MOEAs with tuned control parameters

After all MOEAs underwent tuning, we repeated the tournament from the first part of the experiment for each MOEA. In each tournament, an MOEA with tuned control parameters played against MOEAs which had their default parameters. This enabled us to detect performance improvement of each MOEA

easily. The resulting Rating Intervals are displayed in Figures 4 to 8. The first Figure (Figure 4), shows the improvement of $IBEA$ with tuned parameters. By comparing the results to Figure 3, we can see that $IBEA$ jumped from third to first place whilst the other MOEAs are almost unchanged. The tuning had a big impact on $IBEA$'s performance, since it is significantly better than all other MOEAs. Figure 5 shows the comparison of tuned $MOEAD$ against MOEAs with default parameters. Compared to Figure 3, $MOEAD$ jumped from last to first place meaning, it is no longer significantly worse than $NSGAII$. Even though $MOEAD$ is first, it is not significantly better than any other MOEA. Since Rating Intervals of all MOEAs overlap no claims about one MOEA outperforming another can be made. Figure 6 shows the performance of the tuned $NSGAII$. With default parameters it was already first, and significantly better than $MOEAD$. As we can see from the results, this does not mean it cannot be improved. With tuned parameters it performs significantly better than the rest. As we can see from Figure 7, tuning also had a positive effect on $PESAII$. It took the first place from $NSGAII$, and is significantly better than the other MOEAs. The tuning of $PESAII$ also had an effect on the rating of $MOEAD$. It is no surprise that $MOEAD$ is outperformed by $PESAII$, since it is tuned and it was already outperformed by $NSGAII$. However, now it is also outperformed by $IBEA$, and the Rating Intervals are barely overlapping with $SPEA2$. This means that a bigger portion of $MOEAD$s rating can be attributed to victories against $PESAII$. Figure 8 show the performance improvement of tuned $SPEA2$. As with all other MOEAs, it also jumped to the first place. With tuned parameters it outperforms every MOEA except $NSGAII$. The improvement of $SPEA2$ also had a bigger impact on the rating of $MOEAD$. Its Rating Interval shifted to the left, but not as much when compared with the tuned version of $SPEA2$. For a better comparison we performed a tournament where all tuned versions of MOEAs played against each other. The resulting Rating intervals are displayed in Figure 9. We can see that tuning had the biggest impact on $MOEAD$. It is on par $NSGAII$, whereas with default parameters it performed significantly worse (Figure 3). The remaining MOEAs improved almost identically. The order of $SPEA2$, $PESAII$ and $IBEA$ has switched, but their intervals overlap like they do with default parameters. However, the end user is most interested in the obtained approximation set. Therefore, we plotted the approximation set for system AJHSQLDB obtained by $MOEAD$ with default and tuned parameters on Figure 10. The results show that the approximation set obtained with tuned $MOEAD$ has both better convergence as spread. Overall, we can observe that MOCRS-Tuning was very successful at improving the performance of MOEAs, which was reflected in their rating and obtained approximation sets. The drastic changes in the performance after tuning implies that control parameters play a very important role in their execution, giving us a good reason as to why tuning is important, not only in real-world applications, but also when conducting comparisons amongst MOEAs.

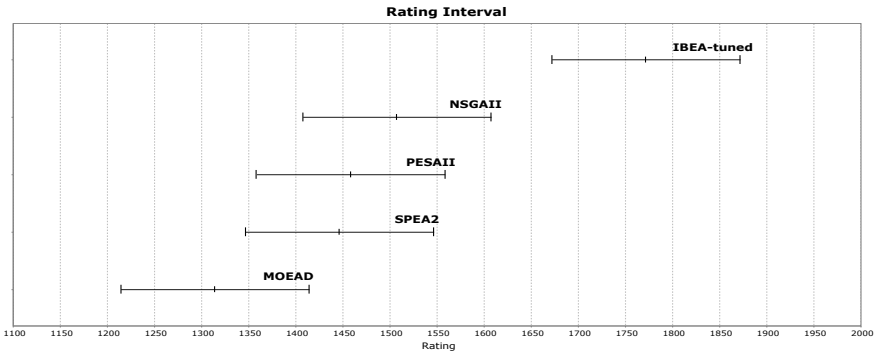


Fig. 4. Comparing $IBEA$ with tuned parameters to MOEAs with default parameters.

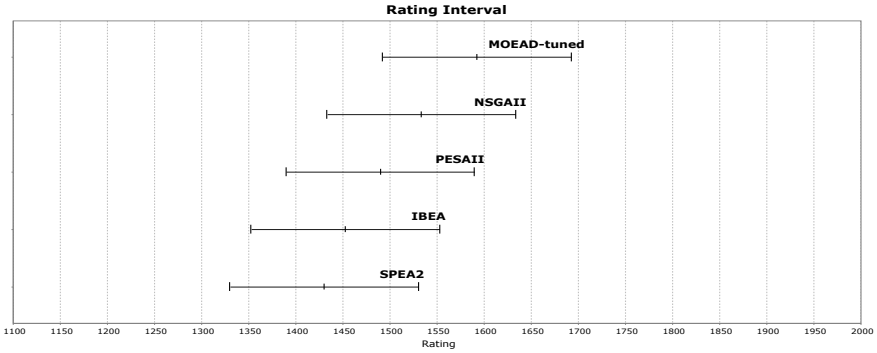


Fig. 5. Comparing $MOEAD$ with tuned parameters to MOEAs with default parameters.

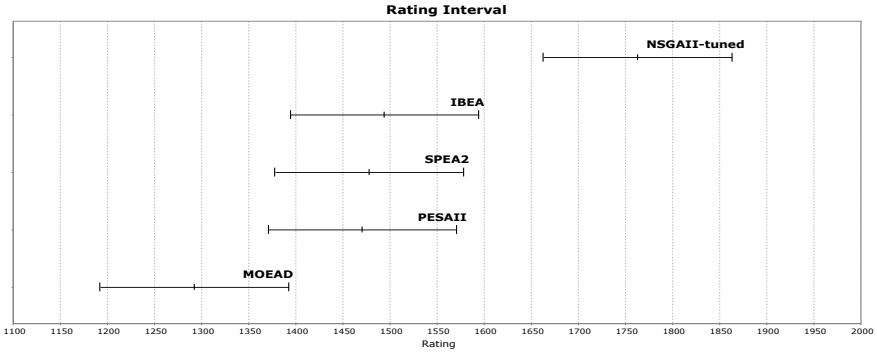


Fig. 6. Comparing $NSGAII$ with tuned parameters to MOEAs with default parameters.

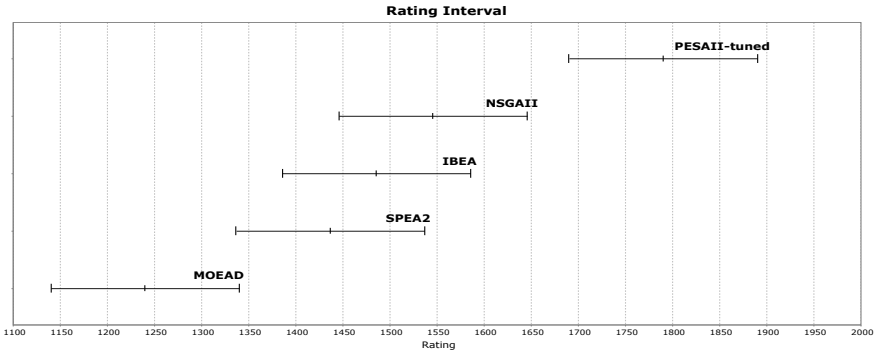


Fig. 7. Comparing $PESAII$ with tuned parameters to MOEAs with default parameters.

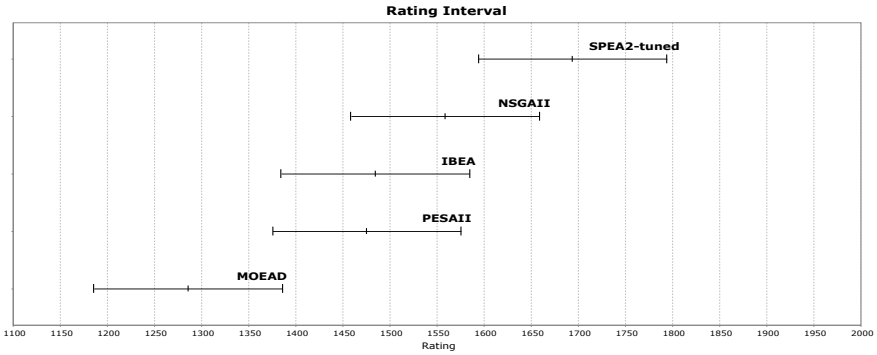


Fig. 8. Comparing $SPEA2$ with tuned parameters to MOEAs with default parameters.

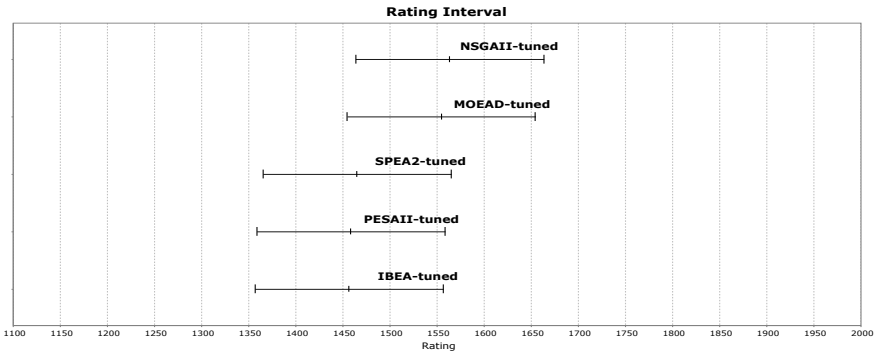


Fig. 9. Comparing all tuned versions of MOEAs.

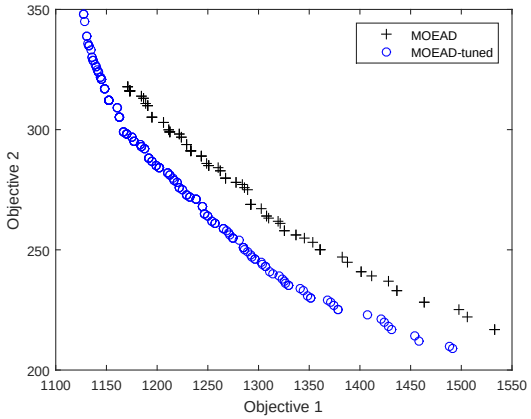


Fig. 10. Comparing approximation fronts of $MOEAD$ with $MOEAD - tuned$ for system AJHSQLDB.

6 Conclusion

In this paper we presented a novel tuning method for multi-objective algorithms called MOCRS-Tuning. The method uses a jDE for the search of optimal parameters and a chess rating system with a Quality Indicator Ensemble for evaluation of MOEAs. The tuning was performed on five different MOEAs. All MOEAs were tuned for the real-world ITO problem on eight systems. All experiments were conducted in the EARS framework. For the comparison between MOEAs before and after tuning, and for the evaluation of solutions in the tuning process, we used CRS4MOEA/QIE. The results have shown that tuned versions of MOEAs have improved significantly compared to their versions with default parameters. This has proven that every MOEA has a benefit when tuning is performed, and that more emphasis needs to be given to control parameters when using MOEAs. Since the tuning process incorporated a Quality Indicator Ensemble, we have assured that different aspects of quality were considered in the tuning process. Because we used eight systems in the tuning process, it was extremely time-consuming. Therefore, each algorithm was tuned only once. The tuning process for one MOEA takes approximately 23 hours on a computer with an Intel(R) Core(TM) i7-4790 3.60 GHz CPU and 16 GB of RAM.

For future work, we would like to speed up the tuning process, which would enable additional tuning runs. In order to know with greater certainty whether the parameters are suitable for a wider set of problems, tuned MOEAs need to be run on additional systems which were not included in the tuning process. We also intend to compare the results of tuned MOEAs with a state-of-the-art method called Hyper-heuristic for the Integration and Test Order Problem (HITO) [4]. This would show us if a well tuned MOEA can compete with a state-of-the-art algorithm. Additionally, we would like to tune MOEAs on problems with higher

numbers of objectives, to see how the number of objectives affects the tuning process of control parameters.

Acknowledgement

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0041).

References

1. M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 11, 2012.
2. W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology*, vol. 51, no. 6, pp. 957–976, 2009.
3. P. McMinn, "Search-based software test data generation: a survey," *Software testing, Verification and reliability*, vol. 14, no. 2, pp. 105–156, 2004.
4. G. Guizzo, S. R. Vergilio, A. T. Pozo, and G. M. Fritsche, "A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem," *Applied Soft Computing*, vol. 56, pp. 331–344, 2017.
5. T. E. Vos, A. I. Baars, F. F. Lindlar, P. M. Kruse, A. Windisch, and J. Wegener, "Industrial scaled automated structural testing with the evolutionary testing tool," in *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, pp. 175–184, IEEE, 2010.
6. A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.
7. G. Karafotias, M. Hoogendoorn, and Á. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.
8. N. Veček, M. Mernik, B. Filipič, and M. Črepinšek, "Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms," *Information Sciences*, vol. 372, pp. 446–469, 2016.
9. J. Brest, V. Zumer, and M. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 215–222, IEEE.
10. M. Ravber, M. Mernik, and M. Črepinšek, "Ranking multi-objective evolutionary algorithms using a chess rating system with quality indicator ensemble," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pp. 1503–1510, IEEE, 2017.
11. M. Ravber, M. Mernik, and M. Črepinšek, "The impact of quality indicators on the rating of multi-objective evolutionary algorithms," in *7th International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2016)*, pp. 119–130, 2016.
12. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

13. W. K. G. Assunção, T. E. Colanzi, S. R. Vergilio, and A. Pozo, "A multi-objective optimization approach for the integration and test order problem," *Information Sciences*, vol. 267, pp. 119–139, 2014.
14. W. K. G. Assunção, T. E. Colanzi, A. T. R. Pozo, and S. R. Vergilio, "Establishing integration test orders of classes with several coupling measures," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 1867–1874, ACM, 2011.
15. B. Beizer, *Software testing techniques*. Dreamtech Press, 2003.
16. "EARS - evolutionary algorithms rating system (github), available at https://github.com/UM-LPM/EARS," 2016.
17. N. L. Hashim, H. W. Schmidt, and S. Ramakrishnan, "Test order for class-based integration testing of java applications," in *Fifth International Conference on Quality Software (QSIC 2005)*, pp. 11–18, IEEE, 2005.
18. M. E. Glickman, "Example of the Glicko-2 system," *Boston University*, 2012.
19. E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature-PPSN VIII*, pp. 832–842, Springer, 2004.
20. Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
21. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
22. D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, *et al.*, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pp. 124–130, 2001.
23. E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," 2001.
24. H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems," in *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pp. 170–177, IEEE, 2014.
25. E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on evolutionary computation*, vol. 3, no. 4, pp. 257–271, 1999.
26. M. P. Hansen and A. Jaszkiewicz, *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical University of Denmark, 1998.
27. G. G. Yen and Z. He, "Performance metric ensemble for multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 131–144, 2014.
28. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.
29. J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.
30. N. Veček, M. Mernik, and M. Črepinšek, "A chess rating system for evolutionary algorithms: a new method for the comparison and ranking of evolutionary algorithms," *Information Sciences*, vol. 277, pp. 656–679, 2014.

Robust Design with Surrogate-Assisted Evolutionary Algorithm: Does it work?

Rodrigo C. P. Silva[1], Min Li[1], Vahid Ghorbanian[1], Frederico G. Guimarães[2], and David A. Lowther[1]

[1] McGill University, Montreal, Quebec, Canada
rodrigo.silva@mail.mcgill.ca
[2] Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil

Abstract. Recently, the use of surrogate models for robustness assessment has become popular in various research fields. In this paper, we investigate whether it is advantageous to use the sample data to build a model instead of computing the robustness measures directly. The results suggest that if the quality of the surrogate model cannot be guaranteed, their use can be harmful to the optimization process.

Keywords: Robust optimization, surrogate models, SPM motor

1 Introduction

The goal of robust engineering design is to optimize performance criteria while minimizing the effect of manufacturing and operational uncertainties, i.e. finding a solution that is robust to uncertain conditions [1]. Often in computer-aided automated design based on finite element analysis (FEA), one solution evaluation may take from a few seconds to several hours of computation, and conventional robustness analysis (e.g. Monte-Carlo simulation) can require a large number of evaluations. For this reason, most robust design schemes are considered unrealistic for practical applications.

In this context, surrogate models have been used in lieu of expensive simulation code to estimate statistical measures of robustness [2–5]. The estimation of statistical quantities, e.g. mean and variance, is not trivial. For instance, [5] suggests that, in some circumstances, global surrogate models are not able to provide accurate estimates of the required quantities. Thus, in [2] and [5], instead of using a global surrogate model, local surrogates, fitted with samples in the neighborhood of the point of interest, are used to estimate robustness.

The use of surrogate models implies the existence of sample data. Besides, the use of local surrogate models implies the existence of data in the neighborhood of the point of interest. If that is the case, one has samples to estimate the robustness directly which in turn brings us to the following question: Do we need surrogate models at all?

In order to investigate this question, we test the framework introduced in [2] and [5] with 4 different types of surrogates plus a surrogate-less (direct calculation) version in a set of benchmark analytical problems. In the tests, in addition

to the type of surrogate, we also vary the number of variables and the number of samples selected for surrogate construction. We compare the surrogate models in terms of the accuracy of the provided estimates, the insensitivity to the number of samples and their distribution, and also in terms of the optimization algorithm effectiveness. Finally, we use the surrogate-assisted robust design method in a bi-objective robust optimization problem related to the design of a Surface-mounted Permanent Magnet (SPM) motor.

2 Robust optimization

In mathematical terms a general optimization problem can be stated as:

$$\min f(\mathbf{x}) \qquad \text{s. t. } \mathbf{x} \in \mathcal{F} \tag{1}$$

where, $f(\mathbf{x})$ is the objective function, $\mathbf{x}$ is the vector of design variables and $\mathcal{F}$ represents the feasible region. The formulation shown in Eq. (1) does not take into account the effect of the uncertainties that often arise in real-world optimization problems. Thus, a more general definition for the objective function $f(\mathbf{x})$ would be: $f = f(\mathbf{x} + \delta, \alpha)$ where δ represents perturbations to the design variables which arise from production tolerances and α are the uncontrollable factors that arise from environmental and material uncertainties.

Robust optimization is a family of optimization approaches that tries to account for uncertainties as the ones defined above. The main goal is to find the, so called, robust solutions which present good performance and small variability with respect to the sources of uncertainty.

This loose definition of robustness can be translated to different formulations of the robust optimization problem such as, for instance, the minimization of the worst-case scenario. In some design problems, however, the worst-case approach can be regarded as too conservative, especially when the worst-cases are very unlikely to happen. Therefore, in this paper, we are going to focus on statistical measures of robustness.

The most commonly used statistical measure of the robustness of a given design is the expected value (mean) given by:

$$\mu_f(\mathbf{x}) = E[f|\mathbf{x}] = \int_{\mathcal{U}(\mathbf{x})} f(\mathbf{x} + \delta, \alpha) p(\delta, \alpha) d\delta d\alpha \tag{2}$$

where, $p(\delta, \alpha)$ is the joint probability distribution of the uncertainties and $\mathcal{U}(\mathbf{x})$ is an uncertainty set and defines the domain of δ and α for each $\mathbf{x}$.

Although the mean gives a more realistic measure of the expected performance, sometimes, it is also important to know the performance variability [4]. Therefore, in order to obtain robust solutions (designs) some measure of dispersion, such as the standard deviation σ, may also be incorporated in the problem formulation. σ is defined by:

$$\sigma_f(\mathbf{x}) = \sqrt{\int_{\mathcal{U}(\mathbf{x})} \left(f(\mathbf{x} + \delta, \alpha) - E[f|\mathbf{x}]\right)^2 p(\delta, \alpha) d\delta d\alpha} \tag{3}$$

3 Surrogate-assisted robust optimization

An important concern in robust optimization is the computational cost related to the robustness estimation, which normally involves the use of Monte-Carlo sampling (MCS). When the sampling involves complex computational models, such as, finite element analysis (FEA), the estimation of robustness may become impractical. To reduce the computational cost, the framework described in Fig. 1 which uses an evolutionary algorithm (EA) as a search mechanism and surrogate models for robustness estimation was proposed in [2].

This algorithm keeps an archive with all the solutions ever evaluated. In step (6), a Latin-hypercube sampling plan (LHS) is generated in the uncertainty set, $\mathcal{U}$, of each offspring for robust assessment. The algorithm searches the archive for the closest neighbor of each point in the LHS. If two points have the same closest neighbor, the farther point from the point of interest in the archive is evaluated with expensive simulation code. Thus, the algorithm takes advantage of previously evaluated solutions and guarantees a reasonably well distributed set of samples. In the next sections a set of possible surrogate models is presented.

```
1.  initialize parent population
2.  initialize archive
3.  while not terminate do
4.      generate offspring
5.      for each offspring do
6.          select archive points for surrogate construction
7.          if no representative set of samples available then
8.              get extra sample points
9.              evaluate the extra sample points
10.             add extra points to the archive
11.         end if
12.         construct local surrogate
13.         evaluate robustness using surrogate
14.     end for
15.     select best offspring as new parent population
16. end while
```

Fig. 1. Surrogate-assisted algorithm for robust optimization

3.1 Polynomial models

Given a $n \times 1$ vector of responses, $\mathbf{y}$, and a $n \times d$ matrix of observed variables, $\mathbf{X}$, where n is the number of samples, the relationship between $\mathbf{y}$ and $\mathbf{X}$ can be described as:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \tag{4}$$

where, β is the vector of regression coefficients, and ϵ the error vector.

The model "training" consists of finding the least squares estimators, $\mathbf{b}$, that minimize the loss function defined in Eq. (5).

$$L = \sum_{i=1}^{n} \epsilon_i^2 = \epsilon'\epsilon = (\mathbf{y} - \mathbf{Xb})^T(\mathbf{y} - \mathbf{Xb}) \tag{5}$$

By taking the derivatives of Eq. (5) with respect to the regression coefficients it is possible to find (see [6] for the detailed derivation) that the least squares estimators of β are given by:

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{6}$$

Thus, a prediction at an unseen point $\mathbf{x}$ is $\hat{y}(\mathbf{x}) = \mathbf{xb}$.

3.2 Kriging

Kriging [7] is an interpolation method that expresses the sought, unknown, function $y(\mathbf{x})$ as a combination of a global model β with local deviations $Z(\mathbf{x})$:

$$y(\mathbf{x}) = \beta + Z(\mathbf{x}) \tag{7}$$

where, β approximates the global trend of the original function while $Z(\mathbf{x})$ creates local deviations in order to approximate a possible multimodal behavior.

Mathematically, $Z(\mathbf{x})$ is the realization of a stochastic process with zero mean, variance σ^2 and covariance given by:

$$Cov[Z(\mathbf{x}_i), Z(\mathbf{x}_j)] = \sigma^2\mathbf{R} \tag{8}$$

$\mathbf{R}$ is the correlation matrix of all the observed data defined as:

$$\mathbf{R} = \begin{bmatrix} R(\mathbf{x}_1, \mathbf{x}_2) & \cdots & R(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ R(\mathbf{x}_1, \mathbf{x}_n) & \cdots & R(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \tag{9}$$

where, $R(\mathbf{x}_i, \mathbf{x}_j)$ is the correlation function defined as:

$$R(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{l=1}^{k} \theta_l |x_{i_l} - x_{j_l}|^2\right) \tag{10}$$

Predicted values at new points are given by:

$$\hat{y}(\mathbf{x}) = \hat{\beta} + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\beta}) \tag{11}$$

where $\mathbf{1}$ is the unit vector, $\hat{\beta}$ is the estimated value of β given by Eq. (14), $\mathbf{y}$ contains the response values of the sample points and $\mathbf{r}^T$ is the correlation vector between an untried point $\mathbf{x}$ and the sampled data points $\mathbf{x}_i$, $i = 1, ..., n$.

$$\mathbf{r}(\mathbf{x})^T = [R(\mathbf{x}, \mathbf{x}_1), R(\mathbf{x}, \mathbf{x}_2), ..., R(\mathbf{x}, \mathbf{x}_n)] \tag{12}$$

Training the kriging model consists of maximizing the likelihood function, given by Eq. (13), in order to find the unknown parameters θ_l.

$$\begin{array}{c}\ln(L(\theta)) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{n}{2}\ln(|\mathbf{R}(\theta)|) \\ -\frac{(\mathbf{y}-\mathbf{1}\hat{\beta})^T\mathbf{R}(\theta)^{-1}(\mathbf{y}-\mathbf{1}\hat{\beta})}{2\sigma^2}\end{array} \tag{13}$$

$$\hat{\beta} = (\mathbf{1}^T\mathbf{R}(\theta)^{-1}\mathbf{y})/(\mathbf{1}^T\mathbf{R}(\theta)^{-1}\mathbf{1}) \tag{14}$$

$$\hat{\sigma}^2 = ((\mathbf{y} - \mathbf{1}\hat{\beta})^T\mathbf{R}(\theta)^{-1}(\mathbf{y} - \mathbf{1}\hat{\beta}))/n \tag{15}$$

3.3 Radial basis functions neural networks

Radial basis functions neural networks (RBFNN) can be implemented in numerous ways. Here, the RBFNN with Gaussian basis functions [3] will be used. This RBFNN has one hidden layer with n neurons, where n is also the sample size, and one output layer. Each neuron in the hidden layer has the following form:

$$\phi_k(\mathbf{x}, \mathbf{c}_k) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_k\|_2^2}{\sigma_k^2}\right) \; 1 \leq k \leq N \tag{16}$$

where $\mathbf{x}$ is some input vector, $\mathbf{c}_k$ is the k^{th} training point which is also the center of the basis function $\phi_k(\cdot)$, and σ_k^2 controls the basis function width.

The output layer is the weighted sum of the hidden layer outputs. Given a set of width parameters σ_k and training points T, consisting of input vectors $\mathbf{c}_i$ and targets $\mathbf{y}_i$, the RBFNN can be concisely expressed in matrix form as:

$$\boldsymbol{\Phi}\mathbf{w} = \mathbf{Y} = [y_1, y_2, \cdots, y_N]^T \tag{17}$$

where,

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(\mathbf{c}_1, \mathbf{c}_2) & \cdots & \phi(\mathbf{c}_1, \mathbf{c}_n) \\ \vdots & \ddots & \vdots \\ \phi(\mathbf{c}_1, \mathbf{c}_n) & \cdots & \phi(\mathbf{c}_n, \mathbf{c}_n) \end{bmatrix} \tag{18}$$

If all the centers are pairwise different, the matrix $\boldsymbol{\Phi}$ is positive-definite and invertible [3]. Thus, the weight vector $\mathbf{w}$ can be computed through Eq. (17). Once the weights are computed, predictions at untried points $\mathbf{x}$ are given by:

$$\begin{array}{l}\hat{y}(\mathbf{x}) = \mathbf{r}(\mathbf{x}) \cdot \mathbf{w} \\ \mathbf{r}(\mathbf{x}) = [\phi(\mathbf{x}, \mathbf{c}_1), \phi(\mathbf{x}, \mathbf{c}_2), \cdots, \phi(\mathbf{x}, \mathbf{c}_N)]^T \\ \mathbf{w} = [w_1, w_2, \cdots, w_N]^T\end{array} \tag{19}$$

3.4 Generalized regression neural network

The Generalized Regression Neural Network (GRNN) is a simple yet powerful regression model proposed in [8]. Different from other artificial neural networks (ANNs), GRNNs do not use backpropagation for training. Instead, the predictions are directly derived from the sampled data using the following formula:

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{n} y_i \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_i)^T(\mathbf{x}-\mathbf{x}_i)}{2\sigma^2}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_i)^T(\mathbf{x}-\mathbf{x}_i)}{2\sigma^2}\right)} \tag{20}$$

where, $\mathbf{x_i}$s and y_is are the sampled input vector and the respective response values. σ is known as the smoothing parameter and controls the width of the Gaussian functions.

4 Computational experiments

In order to test the described surrogates and the optimization framework described in Section 3, three benchmark functions proposed in [9] for robust optimization have been used. They are defined as follows:

$$\begin{aligned} f7a(\mathbf{x}) &= H(x_2) \times \left(\textstyle\sum_{i=3}^{D} 50x_i^2 - S(\mathbf{x})\right) + 1.5 \\ H(x) &= \tfrac{1}{\sqrt{2\pi}}\pi e^{-0.5(\frac{x-1.5}{0.5})^2} + \tfrac{2}{\sqrt{2\pi}}\pi e^{-0.5(\frac{x-1.5}{0.1})^2} \\ S(\mathbf{x}) &= \begin{cases} -x_1^{1.5} & \text{, if } x_2 < 0.8 \\ -x_1 & \text{, if } x_2 \geq 0.8 \end{cases} \\ & x_i \in [0.2, 1.8] \end{aligned} \tag{21}$$

$$\begin{aligned} f10a(\mathbf{x}) &= H(x_2) \times \left(\textstyle\sum_{i=3}^{D} 50x_i^2 - x_1^{0.5}\right) + 1 \\ H(x) &= \tfrac{e^{-x^2}\cos(6\pi x) - x}{4} + 0.5 \\ & x_i \in [0.2, 0.8] \end{aligned} \tag{22}$$

$$\begin{aligned} f16a(\mathbf{x}) = \; & G(x) \\ & \times \left(\left(1 - \sqrt{\tfrac{x_1}{G(x)}} - \tfrac{x_1}{G(x)}\sin(4\pi x_1)\right) + H(x_1)\right) \\ & \times \left(\left(1 - \sqrt{\tfrac{x_2}{G(x)}} - \tfrac{x_2}{G(x)}\sin(4\pi x_2)\right) + H(x_2)\right) + 0.5 \\ H(x) = \; & \tfrac{e^{-2x^2}\sin(12\pi(x+\frac{\pi}{24})) - x}{3} + 0.5 \\ G(\mathbf{x}) = \; & 1 + 10\tfrac{\sum_{i=2}^{D} x_i}{D} \\ & x_i \in [0.2, 0.8] \end{aligned} \tag{23}$$

The uncertainty set is defined as $U(\mathbf{x}) = [\mathbf{x} - 0.2, \mathbf{x} + 0.2]$ for the three problems.

4.1 Surrogate models for robustness assessment

The problem of estimating robustness using sampling is that the computed estimates will depend on the sample data. Hence, different results may be obtained for repeated evaluations at the same design parameter values. Such a noisy objective function may cause the following undesirable behavior [10]: (i) A superior candidate solution may be believed to be inferior and get eliminated; (ii) an inferior candidate may be believed to be superior and get selected for survival and reproduction; and (iii) the objective values may not monotonically improve over the generations.

Thus, a good methodology for robustness assessment should not only provide accurate estimates of the sought measures but also present small variability with respect to the number and the distribution of the samples.

With this in mind, in this section, the described surrogate models are evaluated in the estimation of statistical measures of robustness, more specifically the mean, μ_f, and the standard deviation, σ_f. In order to assess the quality of the surrogates, the average relative error, Eq. (24), is used to estimate the accuracy, and the average coefficient of variation, Eq. (25), is used to estimate the noise caused by the limited number of samples. These metrics are defined below.

$$e = \frac{\sum_{i=0}^{N} |\hat{s}_i - s_{ref}|/s_{ref}}{N} \tag{24}$$

where, $\hat{s}_i$ is the estimated value, s_{ref} is the reference value computed with a MCS of 10^6 samples and N is the number of experiments.

$$cv = \frac{\sum_{i=0}^{N} \sigma(\hat{\mathbf{s}}_i)/\mu(\hat{\mathbf{s}}_i)}{N} \tag{25}$$

where, $\hat{\mathbf{s}}_i$ is the collection of all estimates computed for $\mathbf{x}_i$. The experiment was designed as follows:

1. For each test function, 20 points were randomly selected in the design space;
2. For each selected point $\mathbf{x}_i$ a random sample of k points in $\mathcal{U}(\mathbf{x}_i)$ was generated and evaluated;
3. If surrogate models are used, the k samples are used to fit the surrogate (the values of σ used by GRNNs and RBFNNs were set to 1.201 as suggested in [11]). μ_f and σ_f are computed with a MCS of 10^6 samples evaluated with the surrogate;
4. If surrogate models are not used, μ_f and σ_f are computed directly from the k samples (DIRECT estimates);
5. This experiment was repeated 30 times for each $\mathbf{x}_i$.

Figure 2 illustrates this experiment for $f(x) = x \times sin(x) + 12$, where $\mathcal{U}(x) = [x - 2, x + 2]$. As mentioned before, the average relative error measures how accurate the estimates are. The average coefficient of variation measures, in some sense, the width of the shaded area ($max - min$ estimates of each point) which represents the noise.

Tables 1 and 2 show the average error and the coefficient of variation (in parenthesis) obtained in the estimation of μ_f and σ_f, respectively. Dunn's Test of Multiple Comparisons [12] was used for the statistical analysis. Surrogate-based estimates with average error significantly different ($\alpha = 95\%$) and better than *Direct* are shown in blue. Significantly different ($\alpha = 95\%$) and worse than *Direct* are shown in red. The column *Problem* consists of: *problem name / number of variables / number of samples (k).*

Table 1 shows that, in some of the tested problems, the use of Kriging improved the accuracy of the estimates of μ_f. It has also reduced the noise throughout independent executions when compared to the other models. GRNN

presented results close to the ones obtained when μ_f is directly estimated with the samples ("Direct"). When Polynomials and RBFNN were used, the accuracy decreased and the noise increased when compared with the other approaches.

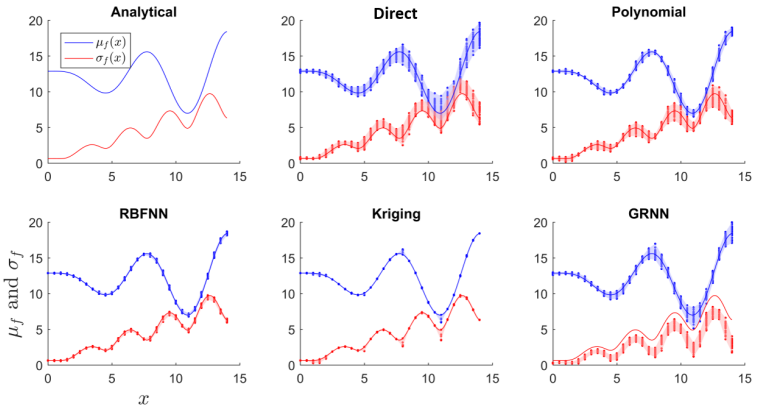


Fig. 2. Surrogate models for robustness estimation.

	Surrogate Model				
Problem	Direct	Kriging	RBFNN	GRNN	2^{nd}-order Polynomial
f7a/5/10	$0.095_{(0.118)}$	$0.044_{(0.060)}$	$0.092_{(0.121)}$	$0.102_{(0.125)}$	$0.083_{(0.106)}$
f7a/5/20	$0.069_{(0.084)}$	$0.008_{(0.011)}$	$0.068_{(0.096)}$	$0.068_{(0.084)}$	$0.124_{(0.244)}$
f7a/5/40	$0.050_{(0.061)}$	$0.003_{(0.003)}$	$0.049_{(0.068)}$	$0.047_{(0.059)}$	$0.016_{(0.019)}$
f7a/10/10	$0.096_{(0.117)}$	$0.090_{(0.114)}$	$0.129_{(0.189)}$	$0.092_{(0.113)}$	$0.179_{(0.257)}$
f7a/10/20	$0.066_{(0.081)}$	$0.039_{(0.058)}$	$0.064_{(0.088)}$	$0.064_{(0.078)}$	$0.066_{(0.081)}$
f7a/10/40	$0.050_{(0.061)}$	$0.008_{(0.015)}$	$0.042_{(0.054)}$	$0.043_{(0.053)}$	$0.053_{(0.070)}$
f10a/5/10	$0.150_{(0.189)}$	$0.130_{(0.161)}$	$0.458_{(0.676)}$	$0.144_{(0.180)}$	$1.243_{(5.228)}$
f10a/5/20	$0.109_{(0.133)}$	$0.052_{(0.072)}$	$0.571_{(0.993)}$	$0.104_{(0.131)}$	$0.328_{(0.446)}$
f10a/5/40	$0.076_{(0.095)}$	$0.015_{(0.019)}$	$0.541_{(0.783)}$	$0.073_{(0.090)}$	$0.102_{(0.130)}$
f10a/10/10	$0.141_{(0.175)}$	$0.140_{(0.173)}$	$0.405_{(0.627)}$	$0.150_{(0.189)}$	$0.427_{(0.583)}$
f10a/10/20	$0.101_{(0.122)}$	$0.087_{(0.114)}$	$0.242_{(0.315)}$	$0.097_{(0.121)}$	$0.249_{(0.330)}$
f10a/10/40	$0.072_{(0.090)}$	$0.032_{(0.048)}$	$0.207_{(0.277)}$	$0.073_{(0.091)}$	$0.209_{(0.268)}$
f16a/5/10	$0.057_{(0.072)}$	$0.063_{(0.078)}$	$0.196_{(0.320)}$	$0.058_{(0.073)}$	$0.140_{(0.197)}$
f16a/5/20	$0.042_{(0.052)}$	$0.041_{(0.050)}$	$0.223_{(0.334)}$	$0.044_{(0.055)}$	$0.362_{(0.849)}$
f16a/5/40	$0.030_{(0.037)}$	$0.025_{(0.031)}$	$0.325_{(0.515)}$	$0.029_{(0.037)}$	$0.043_{(0.055)}$
f16a/10/10	$0.057_{(0.071)}$	$0.059_{(0.073)}$	$0.155_{(0.247)}$	$0.056_{(0.071)}$	$0.161_{(0.230)}$
f16a/10/20	$0.039_{(0.049)}$	$0.042_{(0.052)}$	$0.092_{(0.121)}$	$0.042_{(0.050)}$	$0.089_{(0.115)}$
f16a/10/40	$0.029_{(0.036)}$	$0.027_{(0.033)}$	$0.079_{(0.110)}$	$0.030_{(0.037)}$	$0.086_{(0.110)}$

Table 1. Using surrogates for the estimation of μ_f.

	Surrogate Model				
Problem	Direct	Kriging	RBFNN	GRNN	2^{nd}-order Polynomial
f7a/5/10	$0.173_{(0.215)}$	$0.206_{(0.314)}$	$0.184_{(0.223)}$	$0.982_{(0.356)}$	$0.252_{(0.212)}$
f7a/5/20	$0.119_{(0.149)}$	$0.039_{(0.041)}$	$0.186_{(0.163)}$	$0.982_{(0.253)}$	$0.700_{(0.414)}$
f7a/5/40	$0.079_{(0.098)}$	$0.013_{(0.012)}$	$0.120_{(0.103)}$	$0.983_{(0.188)}$	$0.035_{(0.038)}$
f7a/10/10	$0.160_{(0.207)}$	$0.915_{(3.112)}$	$0.238_{(0.299)}$	$0.978_{(0.304)}$	$0.723_{(0.469)}$
f7a/10/20	$0.120_{(0.148)}$	$0.570_{(1.078)}$	$0.163_{(0.189)}$	$0.981_{(0.249)}$	$0.290_{(0.182)}$
f7a/10/40	$0.080_{(0.101)}$	$0.122_{(0.282)}$	$0.088_{(0.102)}$	$0.982_{(0.180)}$	$0.416_{(0.170)}$
f10a/5/10	$0.177_{(0.228)}$	$0.461_{(0.621)}$	$1.128_{(0.682)}$	$0.986_{(0.405)}$	$7.103_{(1.262)}$
f10a/5/20	$0.117_{(0.149)}$	$0.179_{(0.254)}$	$2.130_{(0.682)}$	$0.989_{(0.367)}$	$1.153_{(0.484)}$
f10a/5/40	$0.085_{(0.106)}$	$0.043_{(0.050)}$	$2.234_{(0.575)}$	$0.990_{(0.302)}$	$0.229_{(0.229)}$
f10a/10/10	$0.157_{(0.202)}$	$0.947_{(3.786)}$	$0.787_{(0.608)}$	$0.983_{(0.306)}$	$1.695_{(0.572)}$
f10a/10/20	$0.101_{(0.125)}$	$0.743_{(1.569)}$	$0.557_{(0.406)}$	$0.986_{(0.261)}$	$1.114_{(0.324)}$
f10a/10/40	$0.070_{(0.086)}$	$0.294_{(0.555)}$	$0.586_{(0.338)}$	$0.989_{(0.241)}$	$2.109_{(0.321)}$
f16a/5/10	$0.190_{(0.235)}$	$0.573_{(0.703)}$	$1.270_{(0.814)}$	$0.986_{(0.416)}$	$1.120_{(0.526)}$
f16a/5/20	$0.137_{(0.168)}$	$0.401_{(0.375)}$	$2.153_{(0.619)}$	$0.988_{(0.339)}$	$6.191_{(0.995)}$
f16a/5/40	$0.093_{(0.115)}$	$0.291_{(0.222)}$	$3.186_{(0.795)}$	$0.989_{(0.267)}$	$0.215_{(0.238)}$
f16a/10/10	$0.189_{(0.238)}$	$0.952_{(3.196)}$	$0.698_{(0.607)}$	$0.982_{(0.347)}$	$1.607_{(0.630)}$
f16a/10/20	$0.133_{(0.166)}$	$0.795_{(1.527)}$	$0.581_{(0.505)}$	$0.985_{(0.286)}$	$0.999_{(0.334)}$
f16a/10/40	$0.090_{(0.113)}$	$0.521_{(0.673)}$	$0.559_{(0.359)}$	$0.988_{(0.250)}$	$1.869_{(0.310)}$

Table 2. Using surrogates for the estimation of σ_f.

As can be seen in Table 2, the estimation of σ_f with surrogates is more complicated than it is for μ_f. In the majority of the tested scenarios, there was no advantage in using surrogate models. Their use led, in most of the cases, to higher levels of noise and less accurate estimates. The exception was the problem $f7a$ for which Kriging presented some improvement when compared with Direct.

4.2 Surrogate-assisted robust optimization

In this section, we compare the aforementioned robustness estimation schemes within the optimization framework presented in section 3. Matlab's genetic algorithm (GA) [11] was used as search engine. Each version of the algorithm is used to minimize μ_f or σ_f for the 5-variable versions of $f7a$, $f10a$ and $f16a$. k is set to 20, the algorithm stops when the number of generations reaches 200 and 20 independent runs are performed for each version.

Fig. 3 presents the convergence curves of the minimization problems regarding μ_f and σ_f for $f10a$ and $f16a$. To generate these curves, the best individual at each generation is evaluated using a Monte-Carlo simulation with 10000 samples. Given the 20 independent runs, the dots represent the average and the bars represent the range of the best individuals' fitness at that generation.

For the μ_f minimization problems, the framework versions with Kriging, GRNNs and direct estimates (without surrogate models) presented the best results which, in turn, can be explained by the error and coefficient of variation

values in Table 1. For the σ_f minimization problems, as expected from the results in Table 2, the version with direct estimates outperformed the others. Interestingly, RBFNNs also presented good performance, despite the high prediction errors. This indicates that, for the tested problems, the prediction errors may have had minimal influence on the ranks of the candidate solutions in the GA. Overall, as can be seen in Fig. 3, the results presented by the framework with direct estimates were more consistent throughout the tested problems.

Fig. 4(a) shows an example of a 3-phase 4-pole surface mounted permanent magnet (SPM) motor. SPM motors have been widely used in applications such as hybrid vehicles and robotics, and their performance highly depends on the produced average torque (T_{avg}) and cogging torque levels (T_{cog}).

It has been shown in [13] that an electrical machine's performance can be significantly affected by small tolerances in the manufacturing process. In this context, instead of optimizing for the nominal values, we optimize the expected

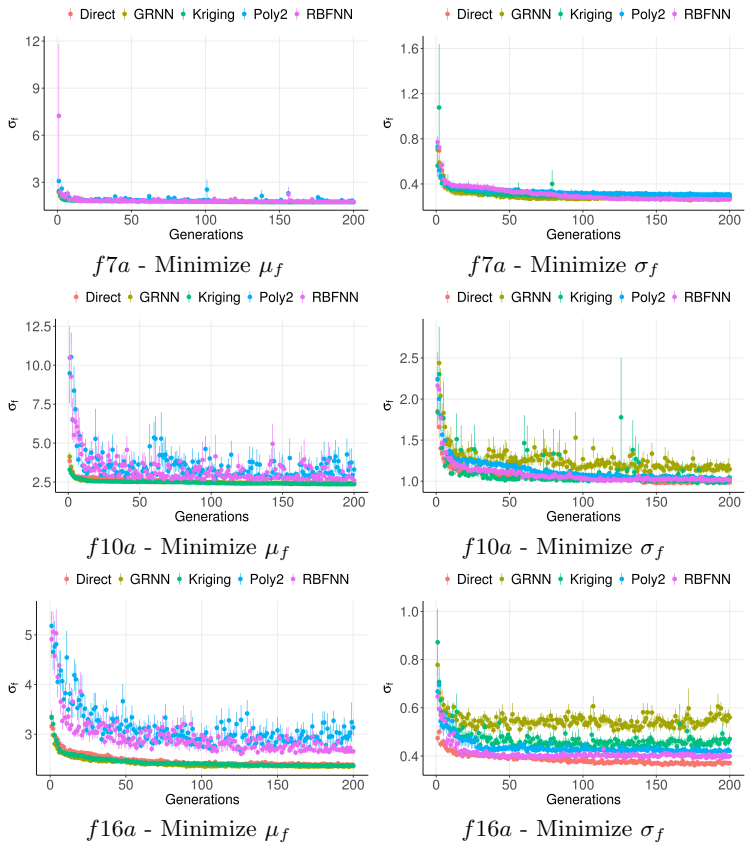
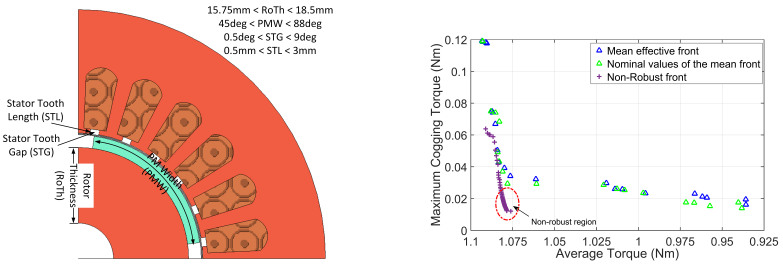


Fig. 3. Convergence curves

(or mean) performance. In this formulation, the tolerances are set to 2% of the lower bound values, that is, $U(\mathbf{x}) = [\mathbf{x} - \mathbf{d}, \mathbf{x} + \mathbf{d}]$ where $d_i = 0.02 \times l_i$. The four design variables are indicated in Figure 4(a).

To solve this problem, Kriging was combined with the Matlab multi-objective genetic algorithm (MOGA) [11]. Given the results presented in the previous section, Kriging seems to be the obvious choice for this kind of problem. For the sake of simplicity, from here on, this method is going to be called as robust multi-objective genetic algorithm (RMOGA).

Fig. 4(b) shows the non-dominated solutions obtained using MOGA to solve the optimization problem without uncertainties (non-robust front) and RMOGA. The non-robust region is highlighted in the figure. It was observed that those solutions are clustered in a small region of the design space which means that a small change in the design variables is causing a large variation of the performance. In the robust front, on the other hand, the solutions are well distributed over the design space, representing the trade-offs between the two objectives. It is worth noting that the front of the optimal designs moves toward lower average torque and higher cogging torque levels if the robust approach is used. This demonstrates the trade-off between performance and robustness in this problem.



(a) SPM motor (b) Pareto-fronts

Fig. 4. SPM motor layout and solutions

5 Conclusion

In this paper, the use of surrogate models for robustness assessment has been investigated. Although it has been shown that surrogate models may improve the accuracy and reduce the noise caused by the small sample sizes, that does not seem to happen very often. Even when the robustness estimation process was improved, the optimization results did not seem to be affected. In fact, in all the tested optimization problems, the surrogate-less method was either the best or among the best performing methods.

Although Kriging presented good performance for the mean related problems, for the general case, where no guarantees about the surrogate model's accuracy

can be provided, it is safer to rely directly on the data acquired from the original problem set-up. If the surrogate model cannot be carefully constructed, it may end up introducing error and noise to the objective function instead of removing them. Although experiments in a larger set of problems with other types of surrogate models may be required to provide a definitive answer to the question proposed in the title, the results displayed here present a fair amount of evidence showing that, in general, the use of surrogate models is not advantageous.

Acknowledgements

Frederico G. Guimarães would like to thank the Minas Gerais State Agency for Research and Development (FAPEMIG).

References

1. Yoon, S.B., Jung, I.S., Hyun, D.S., Hong, J.P., Kim, Y.J.: Robust shape optimization of electromechanical devices. IEEE Transactions on Magnetics **35**(3) (May 1999) 1710–1713
2. Kruisselbrink, J., Emmerich, M., Deutz, A., Back, T.: A robust optimization approach using kriging metamodels for robustness approximation in the cma-es. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. (July 2010) 1–8
3. Yao, W., Chen, X., Huang, Y., van Tooren, M.: A surrogate-based optimization method with rbf neural network enhanced by linear interpolation and hybrid infill strategy. Optimization Methods and Software **29**(2) (2014) 406–429
4. Xiao, S., Li, Y., Rotaru, M., Sykulski, J.K.: Six sigma quality approach to robust optimization. IEEE Transactions on Magnetics **51**(3) (March 2015) 1–4
5. Li, M., Silva, R., Lowther, D.: Global and local meta-models for the robust design of electrical machines. International Journal of Applied Electromagnetics and Mechanics **51**(s1) (2016) 89–95
6. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product in Optimization Using Designed Experiments. 2nd edn. John Wiley & Sons, Inc., New York, NY, USA (2002)
7. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. of Global Optimization **21**(4) (2001) 345–383
8. Specht, D.F.: A general regression neural network. IEEE Transactions on Neural Networks **2**(6) (Nov 1991) 568–576
9. Mirjalili, S., Lewis, A.: Novel frameworks for creating robust multi-objective benchmark problems. Information Sciences **300** (2015) 158 – 192
10. Rana, S., Whitley, L.D., Cogswell, R. In: Searching in the presence of noise. Springer Berlin Heidelberg, Berlin, Heidelberg (1996) 198–207
11. MATLAB: version (R2015a). The MathWorks Inc., Natick, Massachusetts (2015)
12. Dinno, A.: Dunn's test of multiple comparisons using rank sums. Technical report (2017)
13. Lei, G., Wang, T., Zhu, J., Guo, Y., Wang, S.: System-level design optimization method for electrical drive systems - robust approach. IEEE Transactions on Industrial Electronics **62**(8) (Aug 2015) 4702–4713

Author Index