The Complexity of Designing and Implementing Metaheuristics

Ricardo Soto[1,2,3(✉)], Broderick Crawford[1,4,5], Rodrigo Olivares[1], Cristian Galleguillos[1], Kathleen Crawford[1], Franklin Johnson[6], and Fernando Paredes[7]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{ricardo.soto,broderick.crawford}@ucv.cl, rodrigo.olivares@uv.cl, cgalleguillosm@ieee.org, kathleen.crawford.a@mail.ucv.cl
[2] Universidad Autónoma de Chile, Santiago, Chile
[3] Universidad Cientifica del Sur, Lima, Peru
[4] Universidad Central de Chile, Santiago, Chile
[5] Universidad San Sebastián, Santiago, Chile
[6] Universidad de Playa Ancha, Valparaíso, Chile
franklin.johnson@upla.cl
[7] Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago, Chile
fernando.paredes@udp.cl

Abstract. Optimization problems can be found in several real application domains such as engineering, medicine, mathematics, mechanics, physics, mining, games, design, and biology, among others. There exist several techniques to the efficient solving of these problems, which can be organized in two groups: exact and approximate methods. Metaheuristics are one of the most famous and widely used approximate methods for solving optimization problems. Most of them are known for being inspired on interesting behaviors that can be found on the nature, such as the way in which ants, bees and fishes found food, or the way in which fireflies and bats move on the environment. However, solving optimization problems via metaheuristics is not always a simple trip. In this paper, we analyze and discuss from an usability standpoint how the effort needed to design and implement efficient and robust metaheuristics can be conveniently managed and reduced.

Keywords: Optimization problems · Metaheuristics · Local solution · Optimal solution

1 Introduction

Optimization problems can be found in several real application domains such as engineering, medicine, mathematics, mechanics, physics, mining, games, design, and biology, among others. There exist several techniques to the efficient solving of these problems. On one hand, we found the exact methods, which aim at providing the global optimum of the given problem by exploring the complete

C. Stephanidis (Ed.): HCII 2015 Posters, Part I, CCIS 528, pp. 593–597, 2015.
DOI: 10.1007/978-3-319-21380-4_101

search space of potential solutions. A main problem of these techniques is that for several optimization problems, the search space cannot be completely explored in a reasonable amount of time. The approximate methods tackle this concern, which on the contrary, explore only promising regions of the search space in order to provide a good enough local optimum in a limited amount of time. However, they are unable to always guarantee the global optimum.

Metaheuristics are one of the most famous and widely used approximate methods for solving optimization problems [1,10]. Most of them are known for being inspired on interesting behaviors that can be found on the nature, such as the way in which ants, bees and fishes found food, or the way in which fireflies and bats move on the environment. However, solving optimization problems via metaheuristics is not always a simple trip, mainly because metaheuristics are not black boxes ready to be used to solve any optimization problem and there are different topics that must be handled before designing and implementing a metaheuristic: iterations, exploration, exploitation, move operators, and representation of solutions, among others. They must be smartly adapted to the nature of the problem and most of them must be efficiently tuned to reach good results in a reasonable amount of time. This commonly demands specific and advanced knowledge from the user. In this paper, we analyze and discuss a method in order to reduce the effort for designing and implementing metaheuristics.

2 Discussion

In general, when trying to solve optimization problems, we consider different solution techniques. The metaheuristics are one of these techniques, but succeed in correctly using them depends on the maturity of certain implementation and design knowledge. The task is not trivial, as each metaheuristic has its own behavior and characteristics, so you cannot consider it as a black box. Firstly, we need to select among single solution or population-based metaheuristics. Secondly, the representation of the solution must be established. After that, we need to identify the move operators that allow to find or to generate potential solutions. Then, the treatment of unfeasible must be handled, from which we can use three classic methods: penalization, discarding, or repairing. Finally, it will be useful to design a sampling process for the initial configuration of the metaheuristic. Details about those instructions are given in the following.

- The *selection* of the metaheuristic is the first step to determine which algorithm will attempt to solve the optimization problem. As mentioned earlier, there is a strong link between the implementation of the metaheuristic and the problem.
- The *representation of the solutions* is a key process to design and implement a metaheuristic. This process is responsible for defining the data structure to model the solution. This structure is independent from the problem domain, but is bounded by the dimension of the problem.

- The *variable domains* of the problem are the set of values that can take a solution and it is defined by the mathematical model of the optimization problem. This is strongly linked to the representation of the solution.
- The *move operators* generate a new solution, generally from an existing one. They are usually mathematical functions that define the behavior of the metaheuristic.
- The *unfeasible solutions* must be treated in different ways. When move operators generate solutions that are not in the problem domain, you must consider three classic alternatives: discard the solution, penalize, or repair it. These three alternatives are commonly part of the metaheuristic implementations.
- The *sampling process* is an important part of the process. Each metaheuristic has its own configuration so it is necessary to develop an a priori sampling and to define values for each parameter. This process is iterative and depends on the robustness of the implementation of the metaheuristic.
- The *sampling process* aims at defining proper values for each metaheuristic parameter. This process may be useful for the robustness of the metaheuristic.
- At the end, it is crucial to evaluate the performance of the implemented metaheuristic. To this end, the *trial and error process* is key in demonstrating the robustness of the metaheuristic.

This method can be seen as a knowledge discovery process that solves optimization problems using metaheuristics as solving technique. This method has been repeatedly applied by computer science students from the Pontificia Universidad Católica de Valparaíso, Chile. This method has been conducted over a period of five months, corresponding to one semester of study. Students must solve the set covering problem applying metaheuristics and compare their results with 65 reported instances. In Table 1 we illustrate the progress of three iterations of the metaheuristic implementation. In the first iteration, is hard to achieve optimal values, however several values can be close to the optimal one. At the second, and third iteration the number of optimal values improve. Finally, incorporating tuning and efficient repairing, several optimal values can be found. Indeed, several articles have been published based on the student results reached by using this method [2–9].

Table 1. A performance example of three students.

Student	Iteration #1	Iteration #2	Iteration #3
	optimal reached (%)	optimal reached (%)	optimal reached (%)
A	0	≤ 2	≤ 21
B	0	≤ 1	≤ 15
C	0	≤ 1	≤ 19

3 Conclusion

In this paper, we have illustrated a method for designing and implementing metaheuristics for solving optimization problems. This method depends on the maturity of certain basic concepts for all metaheuristic. The aim is to identify and define each of these step according to the given problem and metaheuristic. This method can be seen as a process of knowledge discovery, which has phases that must be resolved for success in the design and implementation of metaheuristic. To date, this method has been used by computer science students with very promising results. We expect to continue improving this method, detailing each of the stages.

Acknowledgments. Ricardo Soto is supported by Grant CONICYT / FONDECYT / INICIACION / 11130459, Broderick Crawford is supported by Grant CONICYT / FONDECYT / REGULAR/1140897, Fernando Paredes is supported by Grant CONICYT / FONDECYT/REGULAR/1130455 and Rodrigo Olivares & Cristian Galleguillos are supported by Postgraduate Grant Pontificia Universidad Católica de Valparaíso 2015.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
2. Crawford, B., Soto, R., Cuesta, R., Olivares-Suárez, M., Johnson, F., Olguín, E.: Two swarm intelligence algorithms for the set covering problem. In ICSOFT-EA 2014 - Proceedings of the 9th International Conference on Software Engineering and Applications, 29–31 August, 2014, Vienna, Austria, pp. 60–69 (2014)
3. Crawford, B., Soto, R., Cuesta, R., Paredes, F.: Application of the artificial bee colony algorithm for solving the set covering problem. Sci. World J. **2014**, 8 (2014)
4. Crawford, B., Soto, R., Cuesta, R., Paredes, F.: Using the bee colony optimization method to solve the weighted set covering problem. In: Stephanidis, C. (ed.) HCI 2014, Part I. CCIS, vol. 434, pp. 493–497. Springer, Heidelberg (2014)
5. Crawford, B., Soto, R., Olivares-Suárez, M., Paredes, F.: A binary firefly algorithm for the set covering problem. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) Modern Trends and Techniques in Computer Science. Advances in Intelligent Systems and Computing, vol. 285, pp. 65–73. Springer International Publishing, Switzerland (2014)
6. Crawford, B., Soto, R., Olivares-Suárez, M., Paredes, F.: Using the firefly optimization method to solve the weighted set covering problem. In: Stephanidis, C. (ed.) HCI 2014, Part I. CCIS, vol. 434, pp. 509–514. Springer, Heidelberg (2014)
7. Crawford, B., Soto, R., Olivares-Suárez, M., Paredes, F., Johnson, F.: Binary firefly algorithm for the set covering problem. In: 2014 9th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–5, June 2014
8. Crawford, B., Soto, R., Palma, W., Johnson, F., Paredes, F., Olguín, E.: A 2-level approach for the set covering problem: parameter tuning of artificial bee colony algorithm by using genetic algorithm. In: Tan, Y., Shi, Y., Coello, C.A.C. (eds.) ICSI 2014, Part I. LNCS, vol. 8794, pp. 189–196. Springer, Heidelberg (2014)

9. Cuesta, R., Crawford, B., Soto, R., Paredes, F.: An artificial bee colony algorithm for the set covering problem. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) Modern Trends and Techniques in Computer Science. Advances in Intelligent Systems and Computing, vol. 285, pp. 53–63. Springer International Publishing, Switzerland (2014)
10. Glover, F., Kochengerger, G.: A Handbook of Metaheuristics. Kluwer Academic Publishers, Boston (2003)