AI Communications 29 (2016) 223–225
DOI 10.3233/AIC-140646
IOS Press



Thesis

# MOSES: A Metaheuristic Optimization Software EcoSystem

José Antonio Parejo
*Universidad de Sevilla, Sevilla, Spain*
*E-mail: japarejo@us.es*

**Abstract.** Many problems that we face nowadays can be expressed as optimization problems. Finding the best solution for real-world instances of such problems is hard or even infeasible. Metaheuristic algorithms have been used for decades to guide the search for satisfactory solutions in hard optimization problems at an affordable cost. However, despite its many benefits, the application of metaheuristics requires overcoming numerous obstacles. First, the implementation of efficient metaheuristic programs is a complex and error-prone process. Second, since there is no analytical method to choose a suitable metaheuristic program for a given problem, experiments must be performed. Besides this, experiments are usually performed ad-hoc, with generic tools and no clear guidelines, introducing threats to validity, and making them hard to automate and reproduce. Our aim is to reduce the cost of applying metaheuristics for solving optimization problems. To that purpose, a set of tools to support the selection, configuration and evaluation of metaheuristic-based applications is presented.

Keywords: Metaheuristics, optimization, AI

## 1. Introduction

Solving optimization problems is an important task which appears in virtually all areas of human activity [3]. Metaheuristics are reusable algorithm schemes that can be tailored for each problem. Metaheuristics have proven to be a handy tool to solve hard optimization problems, providing a balance between the quality of solutions found and the execution time required by the optimization process. Some metaheuristics proposed in literature are *Evolutionary algorithms*, *Simulated annealing*, *Ant colony optimization*, *Tabu search*, etc. Solving optimization problems using metaheuristics requires performing numerous activities to be undertaken in a coordinated manner. This process, which we call the "Metaheuristic Problem Solving (MPS) life-cycle", can be structured in five major stages: *Selection*, *Tailoring*, *Implementation*, *Tuning* and *Execution*. In the *Selection* stage the specific metaheuristics to use for problem solving are chosen. In the *Tailoring* stage the algorithmic schemes of the metaheuristics are completed and tailored to the specific problem at hand, obtaining a fully specified algorithm. In the *Implementation* stage the algorithms are implemented as metaheuristic optimization programs. In the *Tuning* stage specific values for the parameters are set. The result of this stage, denoted as *MPS application*, is a tuned metaheuristic program that can be invoked for a problem instance and provides a solution (or a set of solutions). Finally, the MPS application is executed in the *Execution* stage. Figure 1 depicts this process using BPMN.

The quality of the solutions provided by a MPS application depends on the an appropriate decision making in the stages of selection, tailoring and tuning. However, current theoretical development does not provide analytical methods to make those decisions in general [2]. Therefore, we follow an empirical approach [1,2].

The effort required for running the MPS life-cycle depends strongly on tooling support. In the context of metaheuristic optimization experiments those tools are mainly statistical analysis packages and design of experiment systems. In order to reduce the implementation burden, an extensive number of software frameworks, denoted as Metaheuristic Optimization

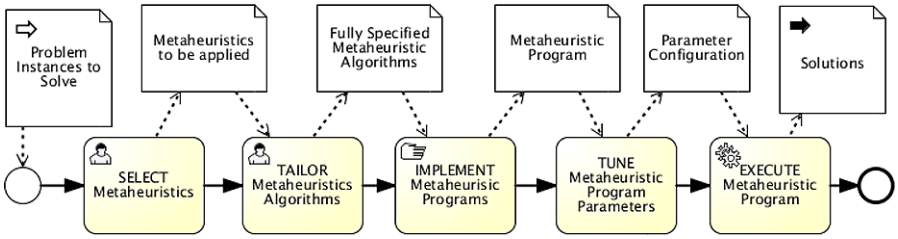


Fig. 1. Metaheuristic problem solving life-cycle. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-140646.)

Frameworks (MOFs), have been created. However, the support provided for the different metaheuristic algorithms is uneven, and its use involves overcoming a steep learning curve. Choosing the appropriate MOF is an important decision for which not much aid has been provided.

We focus on supporting the process of optimization problem solving with metaheuristics when experimentation is required and MOFs are used. Additionally, we aim at enabling the creation of automatically reproducible and easy to replicate experiments with high internal validity.

## 2. Contributions

The main contribution is a set of support tools to reduce the cost of using metaheuristics to solve optimization problems. Such contribution comprises of:

A *Comparison Framework to reduce the cost of choosing the best MOF to solve a given optimization problem*. The framework includes a comprehensive set of features that an ideal MOF should support, definitions of metrics for assessing the support of such features, and means to aggregate such assessments into general quantitative scores. Based on such comparison framework, 10 MOFs are assessed to provide a picture current state of the art. This contribution has been published in the *Soft Computing* journal [5].

Two *languages to reduce the cost of describing, automating and replicating experiments*. The Scientific Experiments Description Language (SEDL) enables the description of domain-independent experiments in a precise, unambiguous, tool-independent, and machine-processable way. SEDL documents provide all the information required to describe the design and execution of experiments including the definition of variables, hypotheses and statistical tests to be applied. SEDL also includes several extensions points for the creation of domain-specific languages. In turn, Metaheuristic Optimization Experiments Description Language (MOEDL) is an extension of SEDL for the description of metaheuristic experiments.

A set of *15 analysis operations on SEDL documents* to reduce the cost of checking the validity and replicability of the experiments. These operations automatically check the existence of validity threats. For instance, an operation that checks if the size of dataset generated by the experimental conduction is consistent with the experimental design is provided.

A *statistical analysis tool (STATService)* to reduce the cost of validating experimental conclusions by testing statistical hypotheses. STATService is designed focusing on inexperienced users with no background on statistical tests. Given an input data set, the tool automatically chooses the most suitable statistical tests providing the corresponding results. The tool has already being used by 9 laboratories in 5 countries.

A Metaheuristic Optimization Software EcoSystem (MOSES) that provides the design of a global architecture for supporting the automation of the experimentation process in the context of metaheuristic optimization. This architecture is defined in terms of service contracts, and software components that act as providers and consumers of those contracts.

A detailed description of the contributions can be found in [4].

## 3. Validation

Those contributions were validated by developing MPS-based applications for solving two relevant software engineering problems: (i) *Quality-driven web service composition*, where the goal is to find a set of candidate services that maximize the overall quality of a web service composition. This contribution was published in Expert Systems with Applications [6]; (ii) *Hard feature model generation*, where the aim is to create feature models as difficult to analyse as possible for current tools, in order to determine its performance in pessimistic scenarios. This contribution was published in Expert Systems with Applications [7].

## Acknowledgements

This work was partially supported by the EU Commission (FEDER), the Spanish and Andalusian R&D&I grants SETI (TIN2009-07366), TAPAS (TIN2012-32273), COPAS (P12-TIC-1867) and THEOS (TIC-5906).

## References

[1] T. Bartz-Beielstein and M. Preuss, Experimental research in evolutionary computation, in: *Proceedings of GECCO*, ACM, 2007, pp. 3001–3020.

[2] M. Chiarandini, L. Paquete, M. Preuss and E. Ridge, Experiments on metaheuristics: Methodological overview and open issues, Technical report, Danish Mathematical Society, 2007.

[3] C.A. Floudas and P.M. Pardalos, *Encyclopedia of Optimization*, 2nd edn, 2008.

[4] J.A. Parejo, MOSES: A Metaheuristic Optimization Software Ecosystem, PhD thesis, 2013.

[5] J.A. Parejo, S. Lozano, A. Ruiz-Cortès and P. Fernandez, Metaheuristic optimization frameworks: A survey and benchmarking, *Soft Computing* **16**(3) (2012), 527–561.

[6] J.A. Parejo, S. Segura, P. Fernandez and A. Ruiz-Cortés, QoS-aware web services composition using grasp with path relinking, *Expert Systems with Applications* **41**(9) (2014), 4211–4223.

[7] S. Segura, J.A. Parejo, R.M. Hierons, D. Benavides and A. Ruiz-Cortés, Automated generation of computationally hard feature models using evolutionary algorithms, *Expert Systems with Applications* **41**(8) (2014), 3975–3992.