A Collaborative Strategy to Reduce Initial Setup Requirements of ParamILS using EvoCa

María-Cristina Riff*
Maria-Cristina.Riff@inf.utfsm.cl

Elizabeth Montero†
Elizabeth.Montero@inf.utfsm.cl

Universidad Técnica Federico Santa María
Avenida España 1680
Valparaíso, Chile

ABSTRACT

ParamILS is a sophisticated tuning method able to provide valuable information for designers and manage conditional parameters. EvoCa is a recently proposed tuner which does not require a fine definition of the initial parameters values. In this work, we propose a collaborative framework between EvoCa and ParamILS to generate quality calibrations without requiring expertise to define a proper initial set-up. Results show that our collaborative approach is able to find good calibrations.

1. INTRODUCTION

Tuning methods have shown being effective tools for searching the space of parameter values of meta-heuristic algorithms. Several different categories of tuners can be clearly identified nowadays: Sampling methods [1, 5], model-based methods [3], screening methods [2] and meta evolutionary algorithms [4, 6].

In this work we focus on Meta evolutionary algorithms that consider the tuning problem as an optimization problem. ParamILS [4] is an iterated local search algorithm that works by searching for better parameter calibrations in the neighborhood of the current one. FocusedILS version of ParamILS uses a comparison method able to increase the number of executions for evaluating parameter calibrations when a more accurate comparison is required. EvoCa [6] is an evolutionary algorithm. Population size in EvoCa is computed considering the number of parameters and their domain sizes. The key idea is to include all the values allowed for each parameter, in an independent way, on the first population. EvoCa also uses two transformation operators: a wheel-crossover and a hill-climbing-first-improvement mutation operator.

*Supported by Fondecyt Project no. 1151456. Partially supported by the Centro Científico Tecnológico de Valparaíso (CCTVal) No. FB0821

†Supported by Postdoctoral Fondecyt Project no. 3130754

GECCO '15 July 11-15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07.

DOI: http://dx.doi.org/10.1145/2739482.2764696

Most research related to tuning methods has been focused on the way the parameter calibrations are generated, searched or analyzed. Experimental setups are presented in order to obtain the best of each tuner, but little attention is paid in how those tuning scenarios should be defined without expert knowledge about the target algorithm. With this work we address the ongoing work proposed by ParamILS'authors "To enhance ParamILS with dedicated methods for dealing with continuous parameters that do not require discretization by the user" [4]. The contribution of this paper is a collaborative strategy that uses EvoCa to define the initial set-up for ParamILS.

2. TUNING PROBLEM

Formally, the tuning problem can be defined as follows:

DEFINITION 2.1. *Given a metaheuristic code M, an instance of the tuning problems consists in a 4-tuple $P = (M, \Theta, \Pi, \kappa_{max})$, where Θ is the configurations space for M. Π is the set of input problem instances, $g(\theta, \Pi)$ is a function that computes the expected gain (e.g., the quality of the solutions) of running M using instance $\pi \in \Pi$ when using configuration θ. κ_{max} is a time out after which all instances of M will be terminated if they are still running.*
Any configuration $\theta \in \Theta$ is a candidate configuration of P. The gain of a candidate configuration θ is given by:

$$G_P(\theta) = mean_{\pi \in \Pi}(g(\theta, \pi)) \quad (1)$$

ParamILS uses iterative local search for focusing its search in some regions of the configurations space. Moreover, it is able to manage conditional parameters discarding irrelevant search and it has been shown to be very effective on tuning algorithms having many parameters as well as for configurating algorithms.

3. COLLABORATION

In order to use ParamILS the designer must discretize target algorithm's parameter configuration space Θ defining a subset $\Theta_d \leq \Theta$, thus ParamILS focus its search on the set Θ_d. We have observed that the better is defined Θ_d on the configurations search space, the more efficiently ParamILS can determine the good values for the parameters.

In general, the Θ space can not be completely visited. The definition of the subset Θ_d seems to be crucial for the performance of ParamILS. Because EvoCa is a tuner that does not require an initial discretization of the configurations and it has shown be effective to find parameter values for metaheuristics, we propose a two-level algorithm where EvoCa is

used in the first level to define the set Θ_d. The evaluation of the performance of the algorithm using different parameters values is used by EvoCa until κ_{max} condition. Given the set Θ_d of configurations obtained by EvoCA our collective strategy will give the control to ParamILS to continue the calibration until κ_{max_2} condition. The pseudocode of our strategy is shown in algorithm 1.

Algorithm 1: EvoCa+ParamILS

```
1  GenerateInitialPopulation (P);
2  while (κ_max condition is not met) do
3  |  child ← Wheel-cross(P);
4  |  Evaluate (child,R);
5  |  ReplaceWorst(P,child);
6  |  mutated_child ← Mutate(child);
7  |  Evaluate (mutated_child, R);
8  |  if mutated_child is better than child then
9  |  |  ReplaceSecondWorst(P,mutated_child);
10 |  end
11 end
   /*Stop using EvoCa --- Start using ParamILS   */
12 definition_file ←WriteParamILSSetUpFile(P);
13 ParamILS(definition_file);
```

3.1 Experimental Setup

For our experiments we use the ACOTSP implementation of Ant Colony Optimization algorithms for the Traveling Salesman Problem. ACOTSP has 11 parameters: three categorical, four integer and four continuous. Categorical parameter "algorithm" determines the ACO used to solve the problem. Continuous parameters α, β and ρ are typical parameters required by all the ACO algorithms. Most of the remaining parameters are conditionals on the ACO algorithm used. The quality of each ACOTSP execution is measured as the relative distance to the optimum.

In our scenarios, we consider as training test ten random Euclidean TSP instances of each 1000, 1500, 2000, 2500 and 3000 cities. The test set is composed of 250 TSP instances (50 instances of each of the previous sizes). A cutoff time of five seconds per run and a budget of 5000 ACOTSP executions were fixed.

We analyzed four types of division of the total budget: 25% of total budget assigned to EvoCa and the remaining 75% assigned to ParamILS (E+PILS(25%)), 50% to EvoCa and the remaining 50% to ParamILS, 75% to EvoCa and the remaining 25% to ParamILS and a special case considering twice the budget: 100% to EvoCa and 100% to ParamILS.

3.2 Discussion

Graph in figure 1 shows the results obtained. Here it is possible to observe that the four collaborative schemes studied show a good quality performance when compared to all the set-ups of parameter values considered. Results in table 1 show the Wilcoxon tests comparing these results. From the tests we can observe that scenario E+PILS(25%) shows to be the best performing approach.

4. CONCLUSIONS

In this work we have focused our attention on the use of two well-known tuners: ParamILS and EvoCa. We have noted that the good choice of the initial points was shown to be crucial to obtain a good performance. We have proposed a collaboration scheme using EvoCa and ParamILS. Our collaboration is useful when the user does not have good information about where are placed the best configurations in the search space. Results show that using EvoCa in the first step of the collaboration reduces the configuration's search space and allows ParamILS to focus on the most promising regions. For a future work we study to incorporate mechanisms in EvoCa to manage conditional parameters.

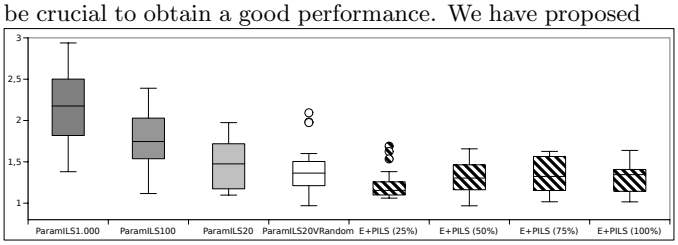

Figure 1: Performance Comparison

Comparison			p-value
ParamILS1.000	vs	**E+PILS (25%)**	.000
ParamILS1.000	vs	**E+PILS (50%)**	.000
ParamILS1.000	vs	**E+PILS (75%)**	.000
ParamILS1.000	vs	**E+PILS (100%)**	.000
ParamiILS20	vs	**E+PILS (25%)**	.011
ParamiILS20	vs	E+PILS (50%)	**.097**
ParamiILS20	vs	E+PILS (75%)	**.154**
ParamiILS20	vs	E+PILS (100%)	**.123**
ParamILS20VRandom	vs	**E+PILS (25%)**	.007
ParamILS20VRandom	vs	E+PILS (50%)	**.330**
ParamILS20VRandom	vs	E+PILS (75%)	**.522**
ParamILS20VRandom	vs	E+PILS (100%)	**.277**
E+PILS (25%)	vs	E+PILS (50%)	.021
E+PILS (25%)	vs	E+PILS (75%)	**.083**
E+PILS (25%)	vs	E+PILS (100%)	**.070**
E+PILS (50%)	vs	E+PILS (75%)	**.701**
E+PILS (50%)	vs	E+PILS (100%)	**.701**
E+PILS (75%)	vs	E+PILS (100%)	**.784**

Table 1: Statistical performance Comparison

5. REFERENCES

[1] B. Adenso-Díaz and M. Laguna. Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operations Research*, 54(1):99–114, 2006.

[2] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *4th international conference on Hybrid metaheuristics*, pages 108–122, 2007.

[3] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *IEEE Congress on Evolutionary Computation*, pages 773–780. IEEE, 2005.

[4] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[5] R. Myers and E. R. Hancock. Empirical modelling of genetic algorithms. *Evolutionary Computation*, 9(4):461–493, 2001.

[6] M.-C. Riff and E. Montero. A new algorithm for reducing metaheuristic design effort. In *IEEE Congress on Evolutionary Computation*, pages 3283–3290, 2013.