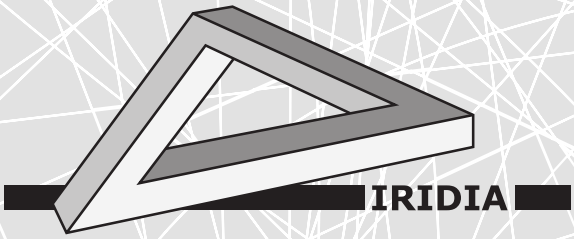

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Exploration of Metaheuristics Through Automatic Algorithm Configuration Techniques and Algorithmic Frameworks

A. Franzin and T. Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2016-005

May 2016
Last revision: October 2016

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2016-005

Revision history:

TR/IRIDIA/2016-005.001 May 2016
TR/IRIDIA/2016-005.002 October 2016

Exploration of Metaheuristics through Automatic Algorithm Configuration Techniques and Algorithmic Frameworks

Alberto Franzin
IRIDIA, Université Libre de Bruxelles (ULB)
afranzin@ulb.ac.be

Thomas Stützle
IRIDIA, Université Libre de Bruxelles (ULB)
stuetzle@ulb.ac.be

ABSTRACT

In this paper[1] we argue that flexible algorithm frameworks can be useful to capture the wide variety of algorithmic components for heuristic algorithms and serve as basic experimental frameworks. One of the utilities is that they can implement the wide variety of different algorithm components and their alternative choices for single stochastic local search methods and we are currently extending existing frameworks in that direction. We exemplify this approach considering the example of Simulated Annealing (SA). In fact, a wide variety of design choices of SA algorithms has been proposed in the literature and algorithm frameworks may (i) simply collect potentially all available choices, (ii) provide a tool for the experimental analysis of specific algorithms and component choices, and (iii) allow the generation of new algorithm variants by combining exisiting components in new ways. We show some limited computational experiments that show the benefit of tuning in this context and the way conclusions on the performance of algorithms are altered in this way.

CCS Concepts

•**Mathematics of computing** → **Simulated annealing; Randomized local search;** *Combinatorial optimization;*

Keywords

Metaheuristics; Simulated Annealing; Algorithm Design; Automatic Algorithm Configuration; QAP; Algorithm Framework

[1]The computational results differ slightly from the ones in the workshop paper for ECADA@GECCO due to an error we had when counting the steps in the sequential neighborhood exploration. This issues affects some numerical results and their interpretation; the main conclusions, however, are not affected.

GECCO'16 Companion, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4323-7/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908961.2931726

1. INTRODUCTION

Automatic algorithm configuration has shown to be a useful technique to relieve algorithm designers from tedious tasks in the tuning of different classes of algorithms and, in particular, stochastic local search (SLS) methods (also called metaheuristics). Several software packages for configuring algorithms have been proposed in the literature and among the most widely used ones we find ParamILS [10], SMAC [9], and irace [14]. These advanced automatic algorithm configuration algorithms do not only allow calibrating numerical algorithm parameters of already fully developed algorithms, but when combined with flexible algorithm frameworks, they can also be exploited for design space exploration and the generation of algorithms that have never been proposed previously in the literature. In fact, many algorithm packages for integer programming can be seen as such algorithm frameworks where by the setting of typically categorical parameters specific routines or heuristics can be switched on or off to obtain potentially new, previously unexplored algorithms. Also algorithm frameworks for heuristic algorithms can benefit in this way from the addition of automatic algorithm configuration techniques. For example, in recent research efforts configurable unified frameworks for multi-objective ACO algorithms [15], ACO algorithms for continuous optimization [13], satisfiability [11] or multi-objective evolutionary algorithms have been proposed and automatically configured algorithms were shown to be competitive and often superior to the various algorithms from which the components for these algorithm frameworks have been taken. These approaches have been further extended to allow the composition of completely new, hybrid algorithms, which can be derived from a simple recursive framework [18].

In our current work, one direction taken is to extend algorithm frameworks to large-scale, configurable frameworks that encompass a much wider set of algorithm components than currently available. A first step to do so is to generate frameworks for specific SLS methods that capture the various algorithmic components that have been proposed in the context of these methods. When doing so, one is able to instantiate the specific algorithms that have been proposed in the literature but also to instantiate completely new combinations of algorithm components even for a same SLS method. The various components implemented for a specific SLS methods can then also form the basis for a more complete framework from which new algorithms taking components from various SLS methods can be composed, for example, in a way similar to what was proposed in [18]. A side remark here is also that we think that such a direction is also

a good alternative to generating ever more (often pseudo-innovative) supposedly new methods from metaphors that often have simply re-invented previously available algorithm components, as criticised in [23].

In this work, we report on some preliminary results obtained by studying one of the most commonly used SLS methods, Simulated Annealing [12, 2], which here we describe in a component-wise fashion. We show how this representation is useful to single out the various features that compose SA, allowing the algorithm designer to intervene on specific parts of the algorithm in order to obtain the desired behaviour in an easier and more precise way and, in particular, by the usage of automatic algorithm configuration techniques. To exemplify the type of analysis and insights one may obtain, we consider here the analysis and tuning of some SA algorithms for the quadratic assignment problem (QAP) as a testbed problem for this analysis.

The remainder of the article is structured as follows. In the next section we introduce our component-based description of SA. In Section 3 we describe some SA implementations for the QAP, while in Section 4 we report some results and analysis. We conclude in Section 6 and outline some of the possible current and future lines of work.

2. COMPONENT-WISE DESCRIPTION OF SIMULATED ANNEALING

Simulated Annealing (SA) [12, 2] is one of the oldest SLS methods proposed in the literature. It draws inspiration from the annealing process in metallurgy, where an object is first quickly heated, and then slowly cooled down until it reaches a near perfect crystalline structure; In particular, the inspiration of the method came from a simulation of the process that happens in the annealing [20]. The main feature of SA is the ability of escaping locally optimal solutions by accepting worsening solutions from time to time; improving moves are usually always accepted. The amount of worsening moves accepted, and the timing in which they are accepted, is regulated by a parameter called *temperature*, which mimicks the usage of the temperature in the physical annealing process. Higher temperatures correspond to higher acceptance probabilities of worsening moves, yielding a more explorative behaviour of the search; lower temperatures correspond to tighter criteria for the acceptance of pejorative moves, making the algorithm more likely to remain in a smaller area of the search space and therefore resulting in a more exploitative behaviour. Usually, the desired behaviour is to allow the search to be more explorative in the first stages of the search, becoming then increasingly exploitative in the latter stages, once a supposedly "good" area of the search space has been found.

SA has been widely studied from a theoretical point of view and it has been widely used in many application. We refer the interested reader to [21] for a recent overview. Over the years, many implementations of SA have been proposed for a wide range of combinatorial problems, often differing in some parts from the original algorithm. These variations either concern numerical parameters for specific algorithm components used in an SA algorithm, or to alternative choices for specific algorithm components used in an SA algorithm. We can classify these variations in nine categories, of which two are problem-related and seven are related to the actual SA behaviour.

The two problem-related components are

1. the *generation of an initial solution*;
2. the *generation of solutions in the neighbourhood* of the current incumbent solution and the *choice of the neighborhood*.

Here we discuss the seven SA-specific components identified.

1. The *initial temperature*, that controls the initial acceptance probability for worsening moves; it can be a fixed value, dependent on some statistics (usually based on a preliminary random walk in the search space), or set to give a predetermined initial acceptance probability for worsening moves. Currently we have implemented seven different schemes.
2. The *neighbourhood exploration*, that determines how the search space has to be traversed. In the literature we have found two exploration schemes, the traditional random one and the scan of the neighborhood in some fixed sequential order.
3. The *acceptance criterion*, whose purpose is to decide whether a worsening solution has to be accepted or not. We have currently implemented nine possible choices.
4. The *cooling scheme*, a non-increasing function that governs the update of the temperature. Currently, we provide ten different schemes.
5. The *temperature length*, that defines the number of solutions evaluated at a certain temperature; it can be a fixed amount of moves, or an amount of moves that varies according to the progress of the search (e.g. a certain amount of accepted solutions). We implemented 12 options so far.
6. The *temperature restarting scheme*, that resets the temperature (often to its initial value) to let the algorithm be able again to accept more worsening moves; also in this case, a predefined condition (number of evaluated solutions, value of temperature), or based on the actual outcome of the search (e.g. being stuck in a local optima). Many options are possible in this case, and we implement 21 of them.
7. The *termination condition*, that decides when to terminate the search; usually this corresponds to a fixed amount of time, or the maximum number of solutions to be evaluated; Currently we have 11 different conditions.

It should also be mentioned that within this framework, we implemented also variants of SA that deviate in a larger sense, e.g. by introducing determininistic acceptance criteria, from standard SA implementations including, thus, also methods such as threshold accepting [5].

In the remainder of this work, we will denote the temperature parameter at a generic iteration i as T_i. A solution in the same instant will be referred to as s_i, and its objective function value as $f(s_i)$; $|N(s_i)|$ will be the size of the neighbourhood of s_i.

3. SIMULATED ANNEALING FOR QAP

For the purposes of this paper we consider the QAP [1] as a testbed problem; we do not delve into discussions about the components that depend on it, but we limit our analysis to the other seven general components. Here, we examine the performance of known SA schemes that have been reported in the literature and consider their respective performance before and after the tuning of their numerical parameters. All the schemes can be instantiated from the SA framework that we have developed.

Considering the QAP, over the years many SA implementations have been proposed for tackling this notoriously difficult combinatorial optimization problem. In this paper, we consider four of the proposed SA algorithms for the QAP

- Two versions proposed by Connolly [4];
- the version of Tam [24];
- the version of Bin Hussin et al. [8].

The SA versions of Connolly have become a standard SA implementation for the QAP as an implementation of it (the below mentioned Q8-7 scheme) is distributed from the QAPLIB webpage (accessible at http://anjos.mgi.polymtl.ca/qaplib/) and it has been used in a number of algorithm comparisons [7]. For the latter scheme, good scaling behavior to large QAP instances has been reported [8].

In their original formulation, the first two schemes that we evaluate feature initial and final temperatures T_0, T_f computed according to the formulas $T_0 = \delta_{min} + (\delta_{max} - \delta_{min})/10$ and $T_f = \delta_{min}$, where δ_{min} and δ_{max} are respectively the smallest and the largest gap between consecutive solutions in a random walk across the search space. The temperature is updated at every step and no temperature restart is employed. The neighbourhood is explored in a sequential order and the solutions are evaluated using the Metropolis acceptance criterion [20]; the search terminates after $50 \times |N(s)|$ moves. The first scheme (CLM) updates the temperature using the Lundy-Mees cooling scheme [17] $T_{i+1} = \frac{T_i}{a+b\times T_i}$ with $a = 1$ and $b = \frac{T_f - T_0}{50\times|N(s)|\times T_0 \times T_f}$. The second scheme (Q8-7) uses the Q8-7 cooling scheme, that is similar to the Lundy-Mees scheme but when it gets stuck in a local optimum the following move is accepted and the temperature is set to the value at which the best solution was found and then held constant for the remainder of the algorithm run.

The third scheme (Tam) sets the initial temperature at a value that gives an initial acceptance probability of 60% (based on an initial random walk). The solutions are generated randomly in the neighbourhood and evaluated with the Metropolis acceptance criterion. The temperature is updated using a geometric cooling $T_{i+1} = 0.95 \times T_i$ every $2 \times |N(s)|$ moves, and the algorithms is stopped after $50 \times |N(s)|$ moves. No temperature restart is used.

The last scheme that we consider in this analysis (Bin) chooses an initial temperature as $0.005 \times f(s_0)$, where s_0 is a randomly generated initial solution. The acceptance criterion is the Metropolis one, and the neighbourhood exploration is the sequential exploration. The temperature is updated with a geometric scheme $T_{i+1} = 0.9\times T$ every $|N(s)|$ moves[2], and is reset to its original value when it reaches a value of 1 (or lower). The algorithm is stopped after a predetermined amount of time.

4. COMPUTATIONAL RESULTS

We instantiated these four variants from a framework for automated component-wise design of metaheuristics. The framework is currently still under development, so we do not describe it in detail here; at a higher level, it contains the possible options for the various components, that get instantiated according to a grammar-based representation of the algorithm, along with their numerical values, following the framework described in [18]. This way we can reproduce the behaviour of many SAs proposed in the literature. The numerical parameters have been kept as defined by the authors in the original works for the experiments with the default parameter settings.

We tested these implementations on a set of 50 randomly generated QAP instances of size 100[3] equally divided in structured instances and random instances; each algorithm is run 30 times with different random seeds, for a maximum of 10 seconds. Results are reported in Figure 1, separated for the two classes of instances.

Clearly, Bin performs much better than the competing algorithms, taking advantage of the temperature restart and higher computational time, finding solutions less than 1% worse than the best-known solutions. The two schemes provided by Connolly also perform overall well, with average results about 2.5% worse than the optimal solution, and little variance, while the results reported for Tam are far worse, being on average 12% higher than the optimal solution. These results would suggest that Connolly's Q7-8 scheme is justified to have a reference position in the evaluation of SA implementations for the QAP.

On random instances, the algorithms have better results, in terms of both lower average and lower variance of the solution quality, when compared to the structured instances, except for Bin that obtains better results on the structured instances.

One immediate tentative improvement is to tune the numerical parameters of the four schemes and to redo the comparisons on a same computation time limit and tuned parameter settings, making the comparison more fair. The parameters we consider are

- the length of the random walk used to compute the initial temperature (from 1 to 10^5, for CLM, Q8-7 and Tam);
- a weighting coefficient for the initial temperature (from 10^{-4} to 10, for all the algorithms);
- the desired initial acceptance probability of worsening moves (for Tam);
- the coefficients a and b for the Lundy-Mees cooling scheme (from 10^{-4} to 1, for CLM and Q8-7);
- the number of consecutive non-accepted moves for stopping the temperature update in the Q8-7 cooling scheme (from 100 to 10^6, for Q8-7);

[2]In the original formulation the authors use $100 \times N$, but for uniformity with the other solutions we can translate this value in terms of neighbourhood size, given that in this work we use only instances of size 100.

[3]taken from http://iridia.ulb.ac.be/supp/IridiaSupp2011-026

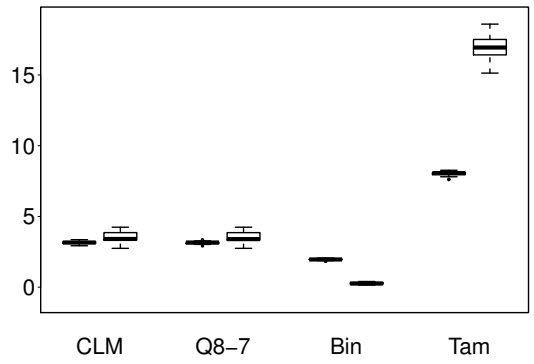

Figure 1: Relative Percentage Deviation from the best-known solutions obtained by the default versions of the four SA algorithms compared. For each algorithm there are two results reported, for random instances (plot on the left) and structured instances (on the right).

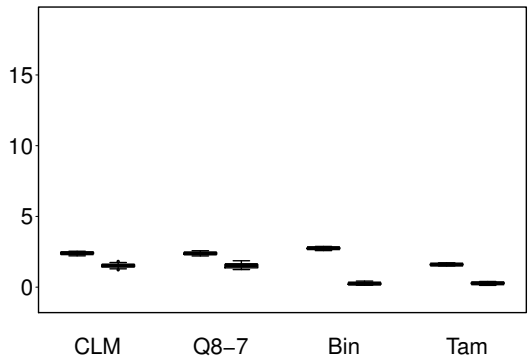

Figure 2: Relative Percentage Deviation from the best-known solutions obtained by the tuned version for the four algorithms. For each algorithm there are two results reported, for random instances (plot on the left) and structured instances (on the right).

- the coefficient α for the geometric cooling scheme (from 10^{-4} to 1, for TAM and BIN);
- the temperature length, as a coefficient of the size of the neighbourhood (from 1 to 100, for all the algorithms);
- the minimum temperature value for the temperature restart (from 1 to 10^5, for BIN);
- the number of iterations of the search, as a coefficient of the size of the neighbourhood (from 1 to 10^5, for CLM, Q8-7 and TAM).

We use the irace automatic algorithm configuration tool [14], with a budget of 2000 experiments, a set of 50 training instances of both classes (different from the ones used for the testing) and a maximum allowed runtime of each experiment of 10 seconds. The results obtained with 30 different tunings are reported in Figure 2, again separating the two instance classes.

The results for BIN are not improving much, because the original settings and allowed running time already suffice to obtain very good solutions, at least on this limited set of instances of the same size, and the power of tuning cannot be fully exploited. All the other algorithms instead score significantly better results. CLM and Q8-7 improve only slightly, as they were already reaching quite good results using their default settings; also, as expected, the variance in the results is lowered. The biggest improvement is obtained for TAM, for which the results of the tuned version are in line with the results obtained by the other algorithms.

All the tuned implementations report lower averages on the structured instances.

In Figures 3–6 we show the distributions of the parameter values given by the 30 different tunings, and how they differ from the default values. In many cases, the default values lie far away from the values of the best configurations found. In some cases the results are quite surprising: the α rate of the geometric cooling scheme, for example, is usually set to values close to 1, while the α chosen for BIN is in most of the cases around 0.5. Another perhaps surprising result is the initial acceptance probability of worsening solutions for TAM, whose average value obtained with the tuning is below 0.2, meaning that, with a proper choice of the other numerical parameters, the algorithm should not exhibit a too explorative behaviour.

5. COOLING VS. FIXED TEMPERATURE

The SA framework directly supports also component-based analysis and here we report on a simple example of such component-based analysis. The purpose of the Q8-7 cooling scheme is to discover a suitable temperature for the search, assuming that if the search is stuck in a locally optimal solution and cannot escape that neighbourhood, then the acceptance probability that was in place when the best solution was found gives a suitably good value for the temperature, and the best option is to stick to that value and not to alter it further with a cooling process.

This raises the question whether a "right" value for the temperature can be chosen from the beginning and kept unchanged until the end. Some authors [3, 22, 6] have already investigated the subject, claiming in certain cases superior results with respect to traditional cooling schemes. In this section, we report some experiments, meant to show how this kind of analysis can be easily done using a suitable framework and automatic configuration tools, rather than as a decisive experiment to confirm or refute the claim.

In order to test in a fair manner the claim that a suitable fixed value for the temperature is competitive or superior to a proper cooling behaviour, we just need to ensemble a SA selecting the following components:

- a fixed-value initial temperature scheme, to be tuned;
- no cooling, temperature length and temperature restart scheme needed;
- same acceptance criterion, neighbourhood exploration and termination condition components of the Q8-7 algorithm.

These last common components ensure that the comparison is done in the fairest possible way, without any intentional or

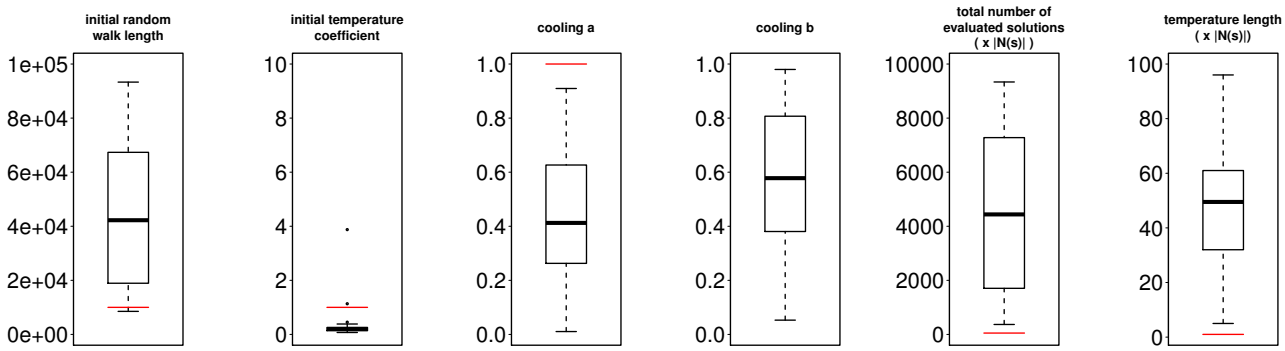

Figure 3: Distribution of the parameter values obtained after the tuning phase for the CLM algorithm, relatively to the allowed range. In red, the default value of each parameter (missing for `cooling b`, as it is dependant on other parameter values).

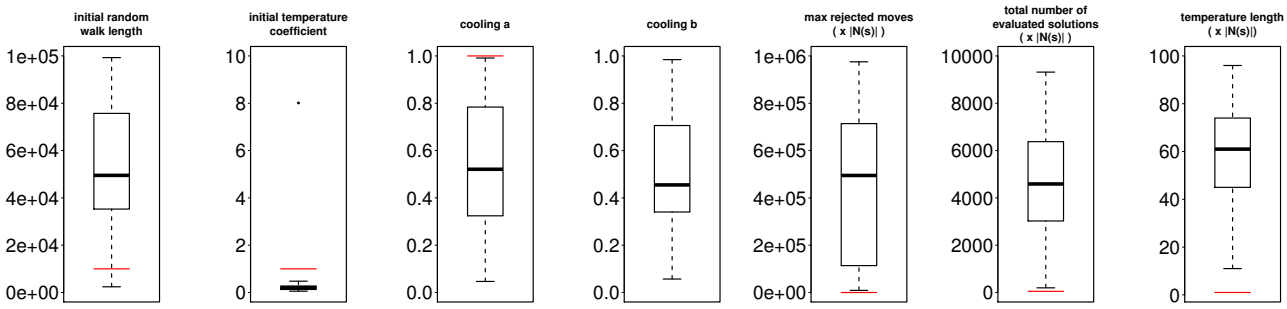

Figure 4: Distribution of the parameter values obtained after the tuning phase for the Q8-7 algorithm, relatively to the allowed range. In red, the default value of each parameter (missing for `cooling b`, as it is dependant on other parameter values).

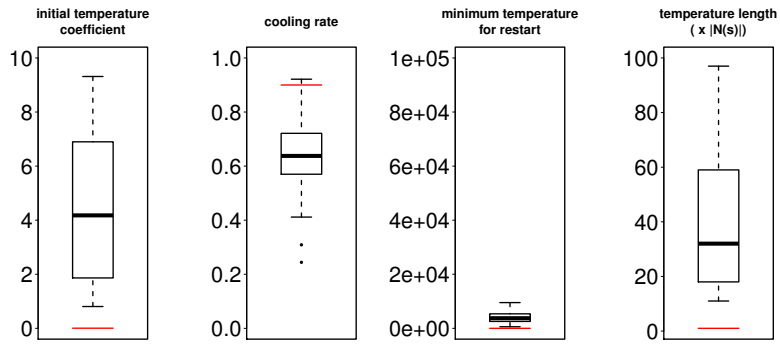

Figure 5: Distribution of the parameter values obtained after the tuning phase for the Bin algorithm, relatively to the allowed range. In red, the default value of each parameter.

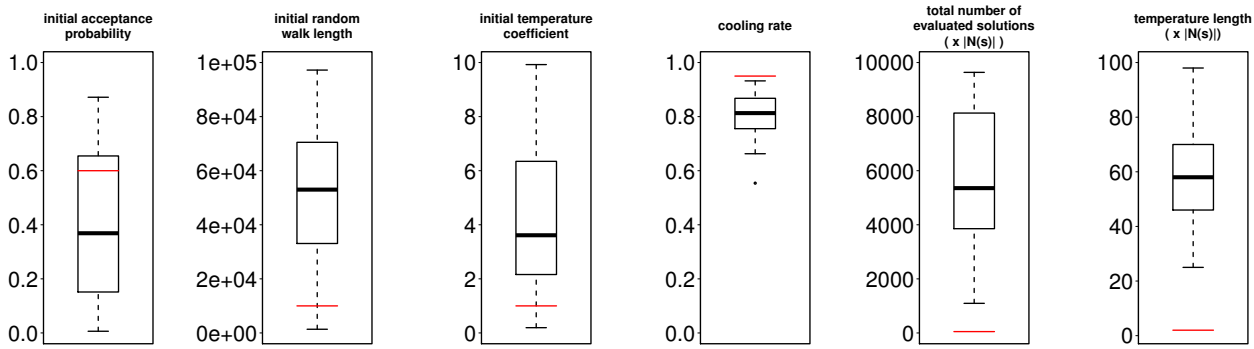

Figure 6: Distribution of the parameter values obtained after the tuning phase for the Tam algorithm, relatively to the allowed range. In red, the default value of each parameter.

unintentional bias that may arise due to the prior opinions or arguments held by the algorithm designers.

The tuning and testing of this fixed-temperature SA is performed in the same way of the previous esperiments.In Figure 7 we compare the results of this scheme with the default and the tuned implementations of Q8-7 for the different classes of instances. The results show that on random instances the fixed temperature scheme performs better than the tuned Q8-7, while on structured instances the results are slightly better than the ones of the default Q8-7, but significantly worse than the results found by its tuned version.

For an interpretation of the results with the fixed temperature scheme, it would be interesting to further analyze whether a good setting of the temperature depends on the specific instance class or, even further, on specific instances. In fact, in further experiments, which will be reported in follow-up work, we have found that a good setting of the fixed temperature depends especially on the class of instances and that good fixed temperature settings differ actually quite strongly between random and structured instances. In a sense the fixed temperature setting when tuning for the two instance classes together results in a compromise value between two very good settings for each of the classes and that further improved performance for the fixed temperature schedules (but also for the simulated annealing algorithms) can be obtained by tuning on each instance class separately.

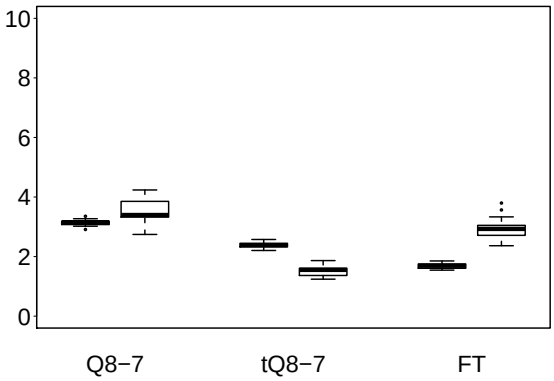

Figure 7: Comparison between non-tuned Q8-7, tuned Q8-7, tuned fixed temperature scheme. For each algorithm there are two results reported, for random instances (plot on the left) and structured instances (on the right).

6. CONCLUSIONS

We have presented a preliminary example of how a complete framework, paired with automatic algorithm configuration techniques, can be used to study existing algorithms for default and tuned parameter settings and how to provide a component-wise analysis.

This approach is also an efficient way of automatically composing algorithm; other approaches in the literature use instead evolutionary searches to generate heuristics [16, 19].

The framework is still to be extended with more options for the various components. We will further deepen the component-wise analysis of SA, in order to discover which components have the highest impact on the final results. The ultimate goals are to have a better understanding of how SA effectively works, and to be able to automatically design efficient state-of-the-art SA implementations for several problems. This work and goals can be extended in the same manner to other metaheuristics and, in theory, to any algorithm that requires its designers to make choices.

Acknowledgments

This research and its results have been made possible through funding from the COMEX project (P7/36) within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Senior Research Associate.

7. REFERENCES

[1] R. E. Burkard, E. Çela, P. M. Pardalos, and L. S. Pitsoulis. The quadratic assignment problem. In P. M. Pardalos and D.-Z. Du, editors, *Handbook of Combinatorial Optimization*, volume 2, pages 241–338. Kluwer Academic Publishers, 1998.

[2] V. Cerný. A thermodynamical approach to the traveling salesman problem. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[3] H. Cohn and M. Fielding. Simulated annealing: Searching for an optimal temperature schedule. *SIAM Journal on Optimization*, 9(3):779–802, 1999.

[4] D. T. Connolly. An improved annealing scheme for the qap. *Eur. J. Oper. Res.*, 46(1):93–100, 1990.

[5] G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175, 1990.

[6] M. Fielding. Simulated annealing with an optimal fixed temperature. *SIAM Journal on Optimization*, 11(2):289–307, 2000.

[7] L. M. Gambardella, É. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176, 1999.

[8] M. S. Hussin and T. Stützle. Tabu search vs. simulated annealing for solving large quadratic assignment instances. *Comput. Oper. Res.*, 43:286–291, 2014.

[9] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *LNCS*, pages 507–523. Springer, 2011.

[10] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *J. Artif. Intell. Res.*, 36:267–306, Oct. 2009.

[11] A. R. KhudaBukhsh, L. Xu, H. H. Hoos, and K. Leyton-Brown. SATenstein: Automatically building local search SAT solvers from components. In C. Boutilier, editor, *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 517–524. AAAI Press, Menlo Park, CA, 2009.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] T. Liao, T. Stützle, M. A. Montes de Oca, and M. Dorigo. A unified ant colony optimization algorithm for continuous optimization. *Eur. J. Oper. Res.*, 234(3):597–609, 2014.

[14] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[15] M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Trans. Evol. Comput.*, 16(6):861–875, 2012.

[16] N. Lourenço, F. Pereira, and E. Costa. Evolving evolutionary algorithms. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 51–58. ACM, 2012.

[17] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical programming*, 34(1):111–124, 1986.

[18] M.-E. Marmion, F. Mascia, M. López-Ibáñez, and T. Stützle. Automatic design of hybrid stochastic local search algorithms. In M. J. Blesa, C. Blum, P. Festa, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 7919 of *LNCS*, pages 144–158. Springer, 2013.

[19] M. A. Martin and D. R. Tauritz. Evolving black-box search algorithms employing genetic programming. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1497–1504. ACM, 2013.

[20] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[21] A. G. Nikolaev and S. H. Jacobsen. Simulated annealing. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 1–39. Springer, New York, NY, 2 edition, 2010.

[22] J. Orosz and S. Jacobson. Analysis of static simulated annealing algorithms. *Journal of Optimization theory and Applications*, 115(1):165–182, 2002.

[23] K. Sörensen. Metaheuristics - the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2013.

[24] K. Y. Tam. A simulated annealing algorithm for allocating space to manufacturing cells. *The International Journal of Production Research*, 30(1):63–87, 1992.