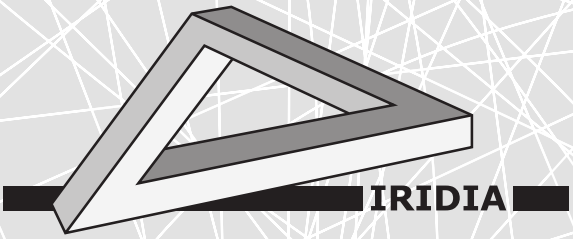


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Off-line *vs.* On-line tuning: A study on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the TSP

Paola Pellegrini, Thomas Stützle, and
Mauro Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2010-009

April 2010

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/2010-009

Off-line *vs.* On-line tuning: A study on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the TSP

Paola Pellegrini
Dipartimento di Matematica Applicata
Università Ca' Foscari Venezia, Venezia, Italia

Thomas Stützle and Mauro Birattari
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
Contact: paolap@unive.it, stuetzle@ulb.ac.be, mbiro@ulb.ac.be

April 2010

Abstract

Virtually all stochastic local search algorithms require the determination of an appropriate setting of their parameters to reach high performance. The parameter tuning approaches that have been proposed in the literature can be classified into two families: *on-line* and *off-line* tuning. In this paper, we compare the results achieved with these two approaches. In particular, we report the results of an experimental study based on a prominent ant colony optimization algorithm, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, for the traveling salesman problem. In particular, we observe the performance of on-line parameter tuning for two different parameter adaptation schemes and for different numbers of parameters to be tuned. Our experimental results indicate that under the experimental conditions chosen here, on-line parameter adaptation can improve over default parameter settings if these are used out of the context for which they were developed. However, the results also clearly indicate that off-line tuned parameter settings are preferable.

1 Introduction

The performance of virtually all stochastic local search (SLS) algorithms [1], depends on the appropriate instantiation of numerical and categorical parameters [2]. Methods that find good parameter settings in an algorithmical, automatic way have recently received a strongly increasing attention by the research community [3, 4, 2, 5, 6, 7]. One main motivation for this tendency is to alleviate algorithm designers from the tedious and error-prone task of hands-on parameter adaptation. The available approaches for this task can be classified at an abstract level into either *off-line* or *on-line* approaches.

The off-line approaches exploit the knowledge gained in an *a priori* tuning phase, where parameter values are optimized based on a training set of instances. The algorithm is then deployed in a production phase using the selected parameter setting. Off-line approaches are typically black-box tools: the process of parameter selection is typically applied without adjustment to the peculiar behavior of the SLS algorithm at hand. Examples of off-line approaches are F-race [4] and its iterated version [8], CALIBRA [9] and ParamILS [6]. The main cost associated to off-line algorithm configuration is the expensive use of resources in the *a priori* experimental phase.

This cost is avoided in on-line tuning approaches, which adapt the parameter values while solving an instance. An advantage of on-line tuning approaches is that they may adjust the parameter values to the characteristics of the particular instance that is being tackled. Hence, intuitively, on-line tuning approaches should benefit relative to off-line tuning when the instance class being tackled is more heterogeneous. In order to adjust the value of the parameters, on-line approaches often use either some search-based mechanism or a mechanism that is based on feedback from the search process. A particularly successful class of on-line algorithms are reactive search approaches as exemplified by reactive tabu search [3]. These approaches adapt typically very few (often only one) key parameter of an algorithm and require substantial insight into algorithm behavior for their development. On-line parameter adaptation has also received a strong interest mainly in the evolutionary computation community [7], where often rather general-purpose parameter adaptation schemes are studied.

In this paper, we compare the performance of off-line and on-line parameter tuning schemes on an ant colony optimization (ACO) algorithm. In particular, we study the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [10] to the traveling salesman problem (TSP). Our experimental setup is based on the initial conjectures that (i) for homogeneous sets of instances the off-line tuning should result in excellent performance; (ii) the more parameters are adapted the worse should get the performance of on-line tuned algorithms; and (iii) that for more heterogeneous instance sets on-line tuning should have an advantage over off-line.

As off-line tuning method we use F-race [11] on the set of candidate configurations we considered. The on-line tuning approaches are given the same set of candidate configurations and we test two on-line [12, 13] approaches, which have shown to be promising for ACO algorithms. Each of the on-line approaches is tested for various numbers of parameters that are to be adapted online. Our results indicate that even if we knew a priori for all the possible subsets of parameters of equal cardinality the subset that results in the best performance, off-line tuning would remain the method of choice. In particular, in our example we can show that, when using off-line tuned parameters as initial values in on-line tuning, the performance of the latter increasingly worsens as the number of parameters to be adjusted increases.

2 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

$\mathcal{MM}$AS [10] is one of the most prominent ACO algorithms. It extends ant system by a more aggressive pheromone update, the usage of upper and lower bounds on the range of possible pheromone trails, and few other details. For the experiments, we use the $\mathcal{MM}$AS implementation provided by the ACOTSP software [14]. In the experiments, we use as a local search the 2-opt algorithm. We refer the reader to the ACOTSP code and the original paper [10] for any detail on the characteristics of the algorithm. We shortly describe here the six parameters that we consider for tuning. The parameters include α and β, which weight the influence of the pheromone trail strength and the heuristic information, respectively, on the probability of choosing a specific edge in the solution construction; m, the number of ants in the colony; and ρ, which represents the pheromone evaporation rate. Here, ants use the pseudo-random proportional action-choice rule of Ant Colony System [15], where with a probability q_0 an ant chooses deterministically, when being at a city i, the next city j as the one for which the product $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$ is maximal, where τ_{ij} and η_{ij} are the pheromone trail strength and the heuristic information, respectively. With a probability $1 - q_0$ the next city is chosen probabilistically as usual in $\mathcal{MM}$AS. A further parameter nn indicates how many cities are considered at most as candidates for the next city to be chosen.

3 Approaches for off-line and on-line tuning

For observing the impact of different tuning procedures on the performance of the algorithm, we consider one off-line and two on-line approaches.

For off-line tuning we apply F-Race [2, 4]. F-Race takes as input a set of algorithm configurations and a sequence of instances. During the execution of F-race, at each step, all surviving candidates are run on one additional instance. After each step, configurations are discarded as they appear to be suboptimal on the basis of the available information. For more details we refer to [4].

The first on-line approach follows the self-adaptive approach [16], that is, the determination of the parameter values is integrated into the algorithms search process. This is done by associating pheromone trails to each possible value of a parameter and to use the ants' construction mechanism to choose which parameter value to adopt. Various authors have proposed variants of such a self-adaptive approach [12, 17, 18, 19]. The available approaches differ mainly in two aspects: parameters can be associated either to each single ant or to the colony as a whole; and, parameters can be considered either independent from one another or as interdependent. Our implementation treats parameters as interdependent and defines their selection at the colony-level. This is done because using different parameter settings for each ant incurs a strong run-time penalty because the speed-up techniques used in the ACOTSP code (essentially pre-computations of values required in the solution construction) would not be effective anymore when using different parameters for each ant. All six parameters described in Section 2 can be adapted in this framework: $q_0, \beta, \rho, m, \alpha, nn$.

The second on-line tuning mechanism uses a search-based procedure for selecting the best values of parameters in the run of the algorithm [13]. The ant colony is split in groups of equal size; a parameter setting in the neighborhood of the incumbent one is assigned to each of them. The neighborhood of the current configuration is defined by all possible combinations that are obtained by increasing or decreasing the value of one parameter and keeping fixed all others. The configuration that corresponds to the best solution generated is used as center of the neighborhood for the next iteration. With this mechanism, only the four parameters involved in solution construction can be adapted, that is, α, β, nn, q_0.

4 Experimental setup

We present an experimental analysis aiming at comparing the performance of off-line and on-line tuning in different experimental conditions. We consider six versions of the $\mathcal{MM}$AS algorithm for the TSP, depending on the tuning procedure used:

- *literature* (L): parameter values are set as suggested in the literature [20]. These settings are highlighted in Table 1. This set of experiments is run as a baseline comparison;
- *off-line* (OFF): F-Race determines the values of the six parameters, which are then maintained fixed throughout all runs;
- *self-adaptive on-line* (SA): the self-adaptive mechanism is used, starting from the parameter setting suggested in the literature [20];
- *search-based on-line* (SB): the search-based adaptation mechanism is used, starting from the parameter setting suggested in the literature [20];
- *off-line + self-adaptive on-line* (OFF+SA): the self-adaptive mechanism is used, starting from the parameter setting returned by F-Race;
- *off-line + search-based on-line* (OFF+SB): the search-based adaptation mechanism is used, starting from the parameter setting returned by F-Race.

When an on-line approach is used, we solve each instance with all the possible subsets of the set of adaptable parameters. In this way, we study how results change if the number of parameters adapted increases. Moreover, for each possible cardinality of the subset of parameters, we register the performance of the algorithm algorithm for all the possible combination of parameters. In the rest of the paper the name of all versions that include on-line tuning are followed by a number

Table 1: Values that can be chosen for each parameter. The values reported in bold type are the ones suggested in the literature [20]. They are the default values used in variant L.

parameter	values
α	0.5, **1**, 1.5, 2, 3
β	1, **2**, 3, 5, 10
ρ	0.1, **0.2**, 0.3, 0.5, 0.7
q_0	**0.0**, 0.25, 0.5, 0.75, 0.9
m	5, 10, **25**, 50, 100
nn	10, **20**, 40, 60, 80

Table 2: Sets of instances considered. $U(a, b)$ indicates that for each instance of a set a number was randomly drawn between a and b. F-race selection indicates the parameter settings selected by F-Race for each set of instances for a computation time limit of 10 CPU seconds.

set	number of nodes	spatial distribution	F-Race selection					
			α	β	ρ	q_0	m	nn
1	2000	uniform	1	5	0.75	0.5	25	20
2	2000	clustered	2	1	0.25	0.75	25	40
3	2000	uniform and clustered	1	1	0.25	0.9	25	20
4	$U(1000, 2000)$	uniform	1	5	0.75	0.25	50	20
5	$U(1000, 2000)$	clustered	2	2	0.25	0.75	50	40
6	$U(1000, 2000)$	uniform and clustered	1	1	0.25	0.9	50	20

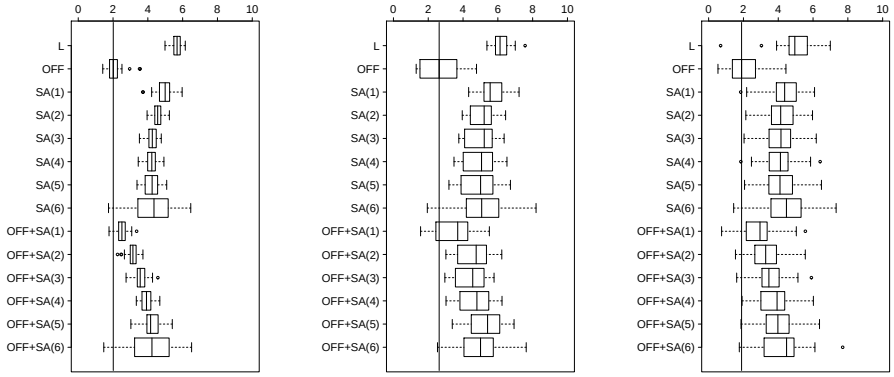
between parenthesis indicating how many parameters are adapted. As said before, the publicly available ACOTSP software [14] was used and compiled with `gcc`, version 3.4.

For a fair comparison between off-line and on-line tuning, the same set of parameter values are available to the two approaches, that is, at each step the approaches can choose among a common set of parameter values. The possible values (used in this order in the second online adaptation scheme) are reported in Table 1.

We consider six sets of instances, all generated using Portgen, the instance generator adopted in the 8th DIMACS Challenge on the TSP [21]. They differ in the number of cities included and in their spatial distribution, for details we refer to Table 2, where also the parameter values chosen by F-Race are indicated. We created these sets for having various levels of heterogeneity. The instance sets range from homogeneous sets where all instances are of a same size and a same spatial distribution of the nodes (either uniformly at random or clustered) to increasingly heterogeneous ones where the instances differ either in their size or also in the spatial distribution of the nodes; hence, the most heterogeneous set is set 6.

For each set of instances, a separate run of F-race is performed using 1000 training instances. All combinations of the values reported in Table 1 are considered as candidates settings. Hence, a total of 15,625 configurations is tested, on a maximum total number of runs equal to 156,250.

We executed experiments with two different termination criteria, after 10 and 60 CPU seconds as measured on Xeon E5410 quad core 2.33GHz processors with 2x 6 MB L2-Cache and 8 GB RAM, running under Debian GNU/Linux. A computation time of 10 CPU seconds allows ants to generate about 2 500-3 000 solutions. The results presented in Section 5 depict the percentage error with respect to the optimal solution for 44 new test instances of each set.



(a) Set 1 (b) Set 3 (c) Set 6

Figure 1: **Results simulating no *a priori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

5 Experimental results

The results for all sets of instances described in Section 4 are reported in the Appendix. Due to the limited space available, we focus here on results for sets 1, 3, and 6: 2000 uniformly distributed nodes, 2000 nodes either clustered or uniformly distributed, and a random number of nodes between 1000 and 2000 either clustered or uniformly distributed. Moreover, the results of the search-based adaptation scheme are just shortly discussed; they are not graphically presented. We compare the two tuning approaches simulating different levels of knowledge on which are the most relevant parameters to be tuned on-line.[1]

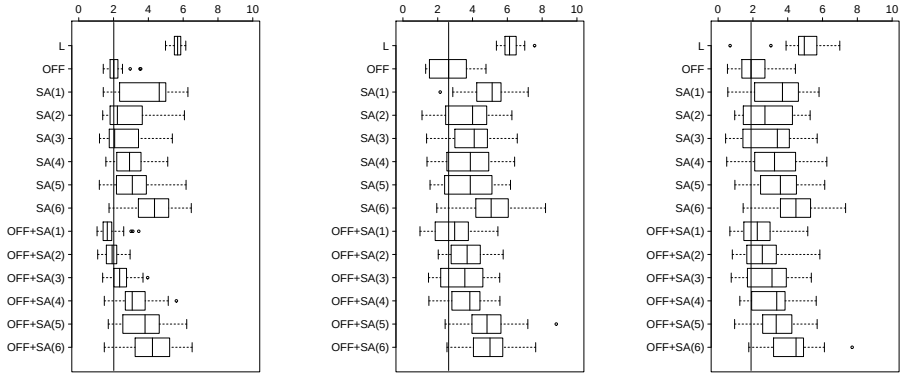
5.0.1 Experiment 1: no *a priori* knowledge on parameter importance for on-line tuning.

If no a priori knowledge on parameter importance for on-line tuning is available, we use the average computed across all possible combinations for each number of parameters tuned as an indication of the expected performance level. These aggregate results are presented in Figure 1: The boxplots summarize the average results in terms of percentage error.

The performance of OFF is the best for all the sets of instances considered. The difference with respect to the *literature* version and to all *self-adaptive on-line* ones is always statistically significant at the 95% confidence level, according to the Wilcoxon rank-sum test. The only exception is represented by OFF+SA(1) on set 2. Interestingly, the literature version (L) appears to be the worst for all sets. The reason is probably that the default literature settings have been developed for situations where the computation time available is rather large.

Depending on whether L or OFF settings are used as initial parameter values, different behavior of the on-line parameter adaptation schemes an be observed. In the first case (results labeled SA in the plots), the on-line parameter adaptation schemes help to improve the reached solutions quality, the best being to adapt three or four parameters. Clearly, on-line has some potential to improve upon fixed initial parameter values if these are not chosen appropriately. The result is very different in the second case, when starting from OFF parameter settings (results labeled OFF+SA in the plots). In fact, in this case on-line tuning clearly worsens the final solution quality in a quite regular fashion. Interestingly, the more parameters are adapted, the worse is the final

[1] A parallel analysis has been performed considering as stopping criterion for each variant the construction of as many solutions as the *off-line* version did in 10 seconds. The results achieved are not significantly different from the ones reported here.



(a) Set 1 (b) Set 3 (c) Set 6

Figure 2: **Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

average solution quality reached. Remarkably, this conclusion remains the same for different levels of the heterogeneity of the instance sets.

In the parallel analysis based on the search-based adaptation, we can draw very similar conclusions. OFF results always statistically better than both SB and OFF+SB. The only difference consists in the observation that in some cases L is better than SB and even of OFF+SB. In general, the performance of the search-based adaptation approach appears worse than the one of the self-adaptive.

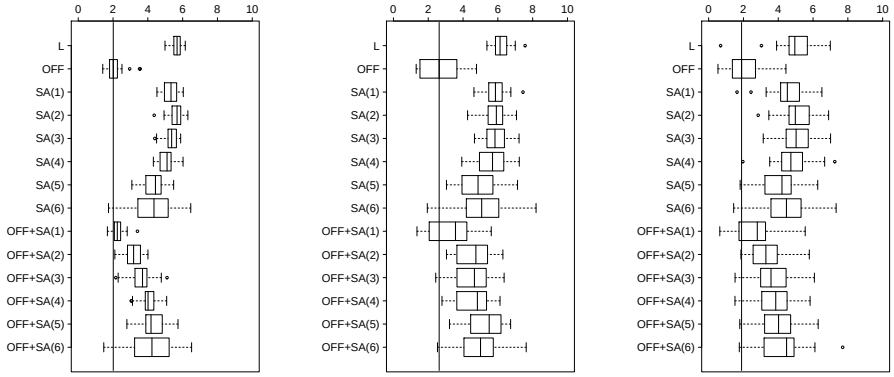
5.0.2 Experiment 2: perfect *a posteriori* knowledge on parameter importance for on-line tuning.

Here we simulate the case in which the algorithm designer knows exactly, which are the most important parameters to be tuned. This is done by considering the *a posteriori* best configuration for each cardinality of the subsets of parameters to be tuned. In other words, for each possible subset of one, two, ... , six parameters that are adapted on-line, we select the subset that results in the lowest average cost. Such a choice introduces a bias in favor of the on-line tuned versions, but since, as shown below, the off-line version remains preferable, this does not invalidate the main conclusions of the analysis. The results of this best-case analysis are reported in Figure 2.

Interestingly, the off-line tuned version performs significantly better than most of the other versions. Surprisingly, on-line tuning is competitive with off-line tuning in the case when the heterogeneity of the instance set is minimal: in sets 1 and 2 (not shown here), OFF+SA(1) is significantly better than OFF. Further analysis has shown that except for (i) SA(3) and OFF+SA(2) for set 1, (ii) OFF+SA(2) and OFF+SA(3) for set 2, and (iii) OFF+SA(1) for sets 4 and 5, all other subsets of on-line tuned parameters are significantly worse than OFF. Hence, even for the most heterogeneous class of instances, set 6, OFF is performing better than the on-line tuned version. In all the cases, L is significantly worse than all the other versions.

The general trend followed by the quality of the final results as a function of the number of parameters adapted is equivalent to the one observed in Figure 1 and that it confirms also that a good starting point for the on-line tuning, as given by OFF, is preferable over a poor performing starting point, as given by L in this case.

Very similar conclusions can be drawn observing the results obtained using the search-based adaptation mechanism: In particular, OFF is always significantly better than all other versions. Instead, the comparison between L and the versions including on-line tuning is slightly more favorable to the former (see Appendix).



(a) Set 1 (b) Set 3 (c) Set 6

Figure 3: **Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

5.0.3 Experiment 3: realistic *a priori* knowledge on parameter importance for on-line tuning.

For understanding to which extent the best *a posteriori* configurations are those that one would actually test if she wished to adapt a given number of parameters, we asked six researchers and practitioners in the field of ACO to indicate their potential selection of the subset of parameters to be tuned on-line. The aggregated results are reported in Figure 3. We represent the average percentage error over the combinations of parameters suggested.

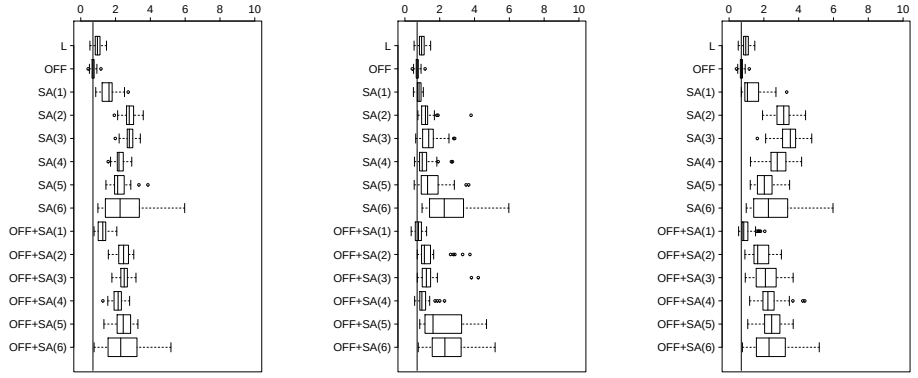
Obviously, OFF is the best performing version, given that is was already best in the previous two experiments. For what SA is concerned, the results are close to the ones observed when considering the average over all combinations. In particular, the variance of the distribution of the percentage error is quite low, and the quality of the solution is comparable to the one achieved by the *literature* version. When we consider OFF+SA, the results of the survey bring to solutions that are in between the average and the *a posteriori* best configuration. Let us remark that the difference between these two representations of the results is in this case quite moderate. Thus, if a well performing initial parameter setting is used, the intuition on the set of parameters that is convenient to be adapted can be expected to bring close to the best possible results. The same observations also hold for the search-based adaptation mechanism (see Appendix).

5.0.4 Experiment 4: long runs.

In this experiment, we examine the impact of the termination condition on the results. In particular, we executed the same set of experiments on the instances of set 1 for a maximum CPU time of 60 seconds (instead of the previously used 10 seconds). The rationale of these experiments is to give the on-line tuning mechanism a longer time to adjust parameters. Figure 4 reports the results achieved using the three cases of no *a priori* information, perfect *a posteriori* information, and realistic *a priori* information, which have been examined in the previous three experiments.

The conclusions that can be drawn are very much in line with those for the shorter computational time: Off-line tuning allows $\mathcal{MM}$AS to achieve the best performance with respect to all the other versions. The differences are statistically significant (checked using the Wilcoxon rank-sum test). A major improvement is experienced by the *literature* version: as we expected, longer runs allow the parameter setting suggested in [20] to achieve good results, confirming the knowledge that these parameter settings have been defined for longer computation times.

The relative performance of the on-line tuned versions with respect to OFF and L slightly



(a) no *a priori* knowledge (b) perfect *a posteriori* knowledge (c) realistic *a priori* knowledge

Figure 4: **Results in long runs.** Runs of 60 seconds. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

improves. When considering the effect of on-line tuning averaged across all subsets of parameters that are adapted, the effect of different cardinalities of these subsets reflects quite closely the above observations: we cannot identify a clear trend for SA, while for OFF+SA we note that the results get worse as the number of parameters that are adapted on-line increases. If we suppose perfect *a posteriori* information, the behavior of SA and OFF+SA is much improved (Figure 4(b)). The results are always significantly worse than OFF, while in some cases they become comparable to L: we have not observed any statistically significant difference between L and SA(4) and OFF+SA(4); the difference is significant in favor of SA(1) and OFF+SA(1) and in favor of L in all other cases.

Observing the results of the survey, representing realistic *a priori* knowledge, (Figure 4(c)), we can notice that the performance of the selected configurations in some cases are even worse than the overall average (Figure 4(a)). This happens for SA(2), SA(3), SA(4), OFF+SA(4), and OFF+SA(5). This observation strengthen the claim that tuning on-line just one or two parameters, instead of a high number of them, is the most convenient choice: Not only we can expect the approach to guarantee quite good results (in some cases not worse than off-line tuning), but also we can assume that our intuition allows us to choose the parameters to adapt so that the potentials of the tuning are actually exploited.

The conclusions on the comparison between off-line *vs.* on-line tuning are the same when we consider the search-based adaptation scheme (see Appendix). What is different is the robustness of SB and OFF+SB with respect to the combination of parameters adapted. In fact, the results of SB when the parameter adapted is either β or nn are consistently rather poor, while this is not the case for α and m. This implies that, even if the search-based approach adapting the *a posteriori* best parameter outperforms the versions with more parameters tuned, the situation is reversed if the wrong single parameter is picked. In fact, four of the six researchers we interviewed identified β as their selection in case one parameter was to be adapted.

5.0.5 Summary of results.

From the results just described we can deduct the following main conclusions:

- off-line tuning performs better than on-line tuning under all the experimental conditions tested;
- on-line tuning generally achieves better results when few parameters (one or two) are considered;

- on-line tuning is more robust if the initial parameter setting is a well-behaving configuration instead of the one suggested in the literature. This is true with respect both to the specific set of parameters adapted, and to the adaptation mechanism adopted.

The heterogeneity of the class of instances tackled does not appear to have a strong impact on the relative performance of the different versions.

6 Conclusions

In this paper we have compared the results achieved by $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System when its parameters are tuned off-line and when they are tuned on-line.

The principal indication that we can learn from the analysis presented is that, whenever possible, off-line tuning should be used. If we do not have at our disposal enough resources to devote to this task, on-line tuning may be a convenient choice if we adapt only few parameters. Nonetheless, if a rather long computational time is available for each instance, we should consider the setting considered in the literature as a good alternative.

These conclusions need to be tested on other combinatorial optimization problems. The merits of on-line tuning, for example, may emerge if the instances to be tacked are extremely different from each other, as it is the case for some scheduling problems. Further research will be performed in this direction.

In the cases in which on-line tuning may be advantageous, the results reported suggest that the implementation of a hybrid between the off-line and on-line approach may be very promising: parameters may be first tuned off-line, and then one or two of them may be adapted while solving each instance. In this way, high quality solutions may be found. The robustness of such a hybrid appears quite strong, both with respect to the specific adaptation mechanism selected, and with respect to the parameter to adapt. In particular, thanks to a social experiments, we could observe that the intuition on the most important parameters may allow to fully exploit the hybrid tuning, while it may be misleading if we use an on-line approach.

Acknowledgments

Mauro Birattari and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are Research Associates. Moreover, the authors would like to thank the colleagues that participated in the survey described in the paper.

References

[1] Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA 94104, USA (2005)

[2] Birattari, M.: Tuning Metaheuristics: A machine learning perspective. Springer, Berlin, Germany (2009)

[3] Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Volume 45 of Operations Research/Computer Science Interfaces. Springer Verlag, Berlin, Germany (2008)

[4] Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In Langdon, W., et al., eds.: GECCO 2002, San Francisco, CA, Morgan Kaufmann Publishers (2002) 11–18

[5] Coy, S., Golden, B., Runger, G., Wasil, E.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics **7**(1) (2001) 77–97

[6] Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research **36** (2009) 267–306

[7] Lobo, F., Lima, C.F., Michalewicz, Z.: Parameter Setting in Evolutionary Algorithms. Springer, Berlin, Germany (2007)

[8] Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In Bartz-Beielstein, T., et al., eds.: HM 2007. Volume 4771 of Lecture Notes in Computer Science., Springer Verlag, Berlin, Germany (2007) 108–122

[9] Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research **54**(1) (2006) 99–114

[10] Stützle, T., Hoos, H.H.: MAX–MIN ant system. Future Generation Computer Systems **16**(8) (2000) 889–914

[11] Birattari, M.: Race. R package (2003) http://cran.r-project.org.

[12] Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Transactions on Evolutionary Computation **11**(5) (2007) 651–665

[13] Anghinolfi, D., Boccalatte, A., Paolucci, M., Vecchiola, C.: Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In Li, X., et al., eds.: SEAL. Volume 5361 of Lecture Notes in Computer Science., Springer Verlag, Berlin, Germany (2008) 411–420

[14] Stützle, T.: ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem. `http://www.aco-metaheuristic.org/aco-code` (2002)

[15] Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 53–66

[16] Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter control in evolutionary algorithms. [7] 19–46

[17] Randall, M.: Near Parameter Free Ant Colony Optimisation. In Dorigo, M., et al., eds.: ANTS 2004. Volume 3172 of Lecture Notes in Computer Science., Springer Verlag, Berlin, Germany (2004) 374–381

[18] Förster, M., Bickel, B., Hardung, B., Kókai, G.: Self-adaptive ant colony optimisation applied to function allocation in vehicle networks. In: GECCO'07, New York, NY, ACM Press (2007) 1991–1998

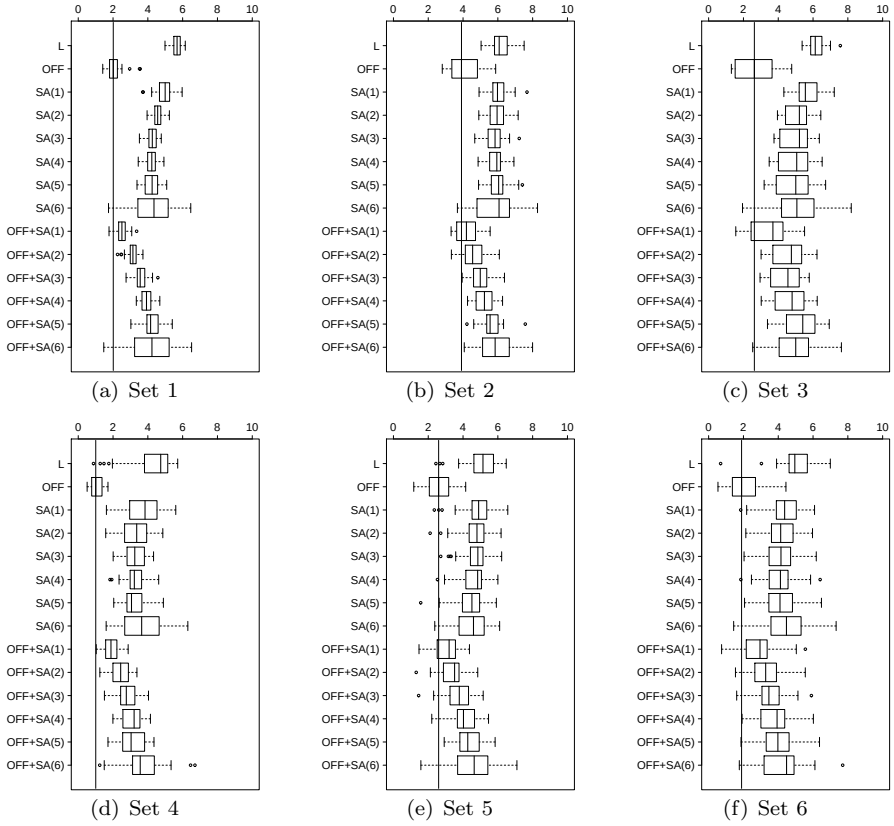
[19] Khichane, M., Albert, P., Solnon, C.: A reactive framework for ant colony optimization. In Stützle, T., ed.: Learning and Intelligent OptimizatioN (LION). Volume 5851 of Lecture Notes in Computer Science., Springer Verlag, Berlin, Germany (2009) 119–133

[20] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)

[21] Johnson, D., McGeoch, L., Rego, C., Glover, F.: 8th dimacs implementation challenge. `http://www.research.att.com/$\sim$dsj/chtsp/` (2001)

Appendix: Further Experimental Results

In the following we report the results of the whole experimental analysis described in Section 4. The graphs are organized as in Section 5.



(a) Set 1 (b) Set 2 (c) Set 3

(d) Set 4 (e) Set 5 (f) Set 6

Figure 5: **Results simulating no *a priori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

Experiments 1: no *a priori* knowledge on parameter importance for on-line tuning.

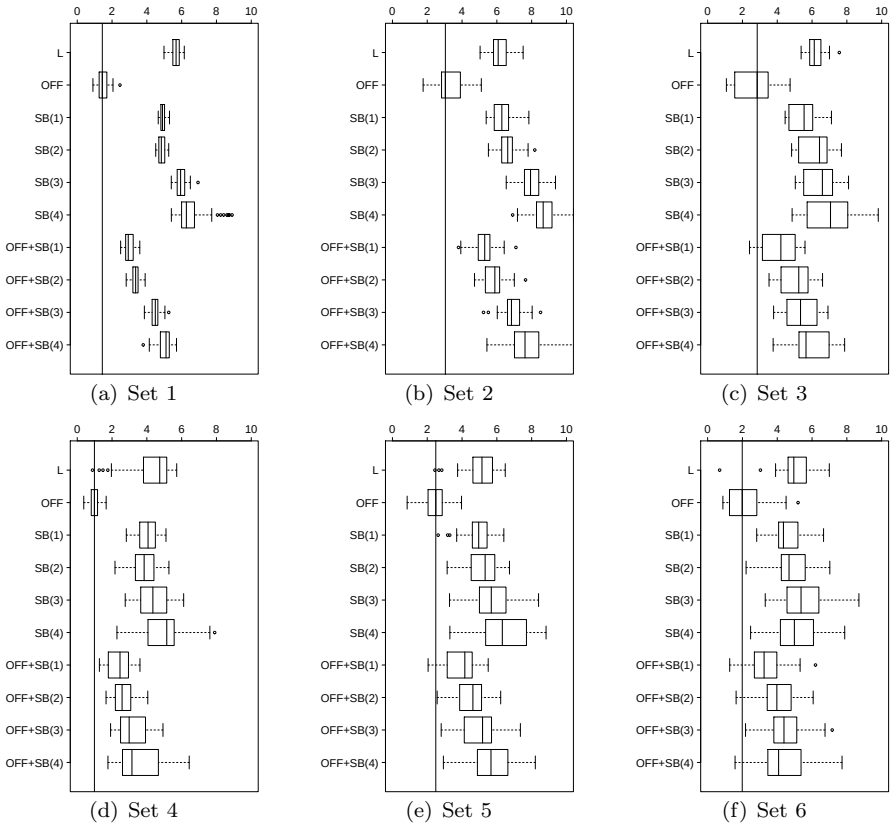
If no a priori knowledge on parameter importance for on-line tuning is available, we can expect the results to be similar to the average computed across all possible combinations for each number of parameters tuned. These aggregate results are presented in Figures 5 and 6: The boxplots summarize the average results in terms of percentage error.

Experiments 2: perfect *a posteriori* knowledge on parameter importance for on-line tuning.

The case in which the algorithm designer knows exactly which are the most important parameters to be tuned is simulated considering the *a posteriori* best configuration for each number of parameters adapted. Such a choice introduces a bias in favor of the on-line tuned versions. These results are presented in Figures 7 and 8: The boxplots summarize the results in terms of percentage error.

Experiments 3: realistic *a priori* knowledge on parameter importance for on-line tuning.

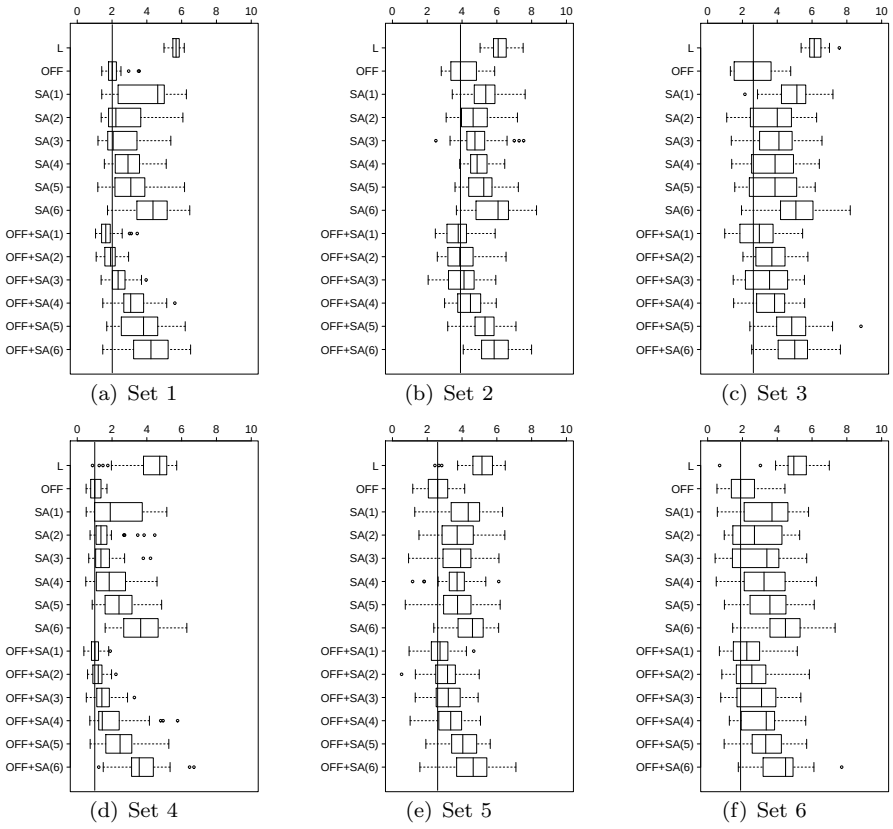
For understanding to which extent the best *a posteriori* configurations are those that one would actually test if he wished to adapt a given number of parameters, we asked six researchers and practitioners in the field of ACO to indicate their potential selection. The aggregated results are



(a) Set 1 (b) Set 2 (c) Set 3

(d) Set 4 (e) Set 5 (f) Set 6

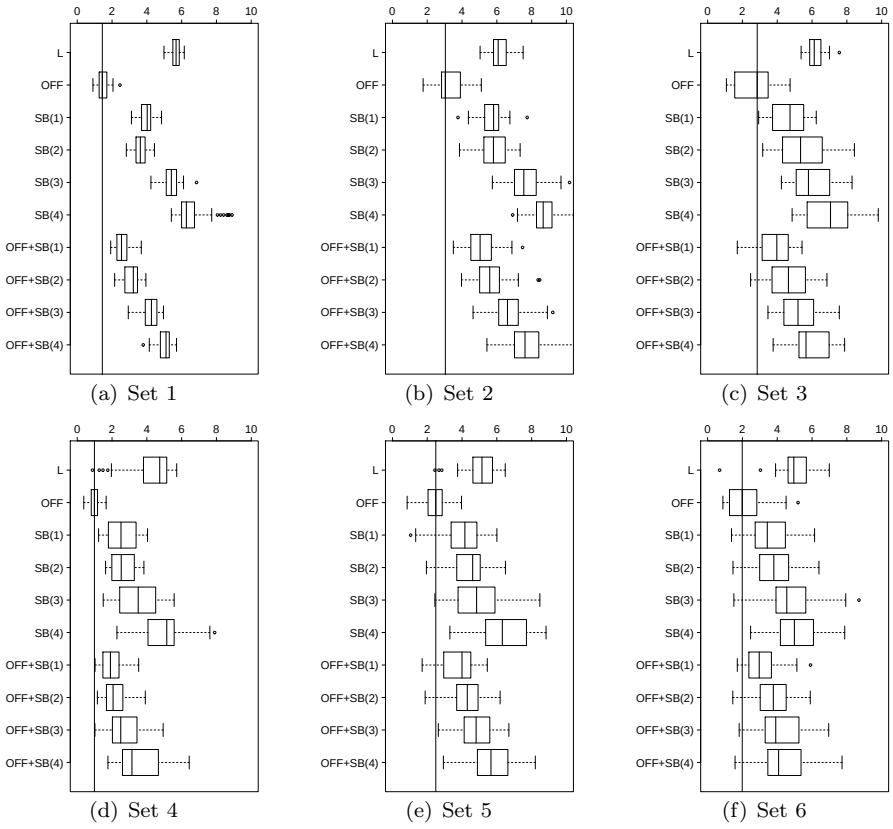
Figure 6: **Results simulating no *a priori* knowledge on parameter importance for online tuning.** Runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.



(a) Set 1 (b) Set 2 (c) Set 3

(d) Set 4 (e) Set 5 (f) Set 6

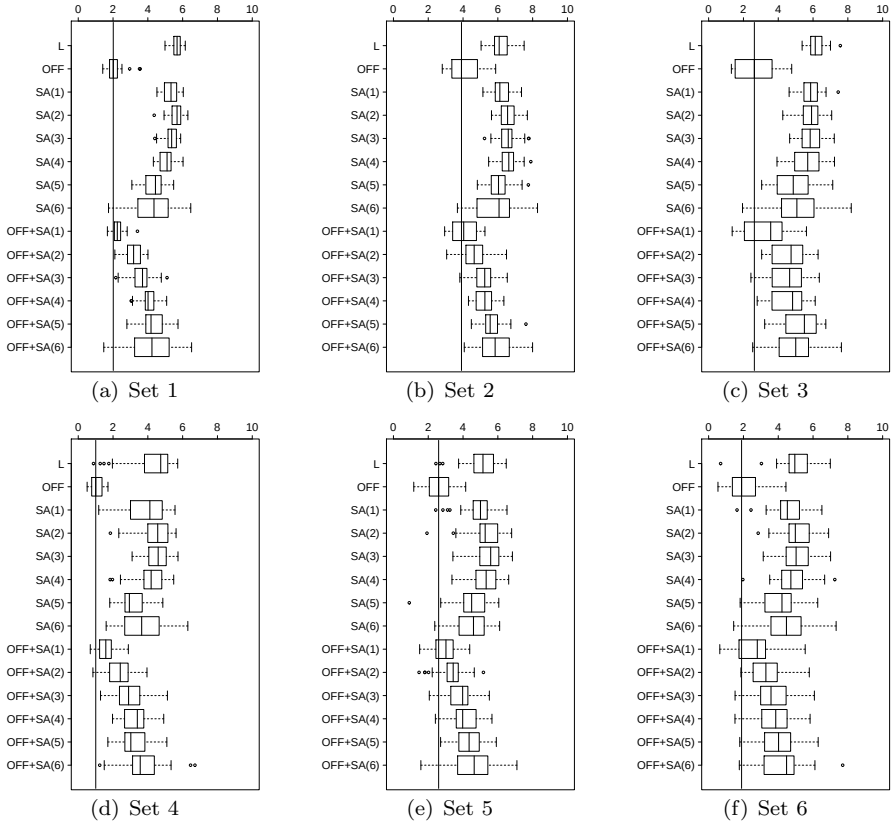
Figure 7: **Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.



(a) Set 1 (b) Set 2 (c) Set 3

(d) Set 4 (e) Set 5 (f) Set 6

Figure 8: **Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.



(a) Set 1 (b) Set 2 (c) Set 3

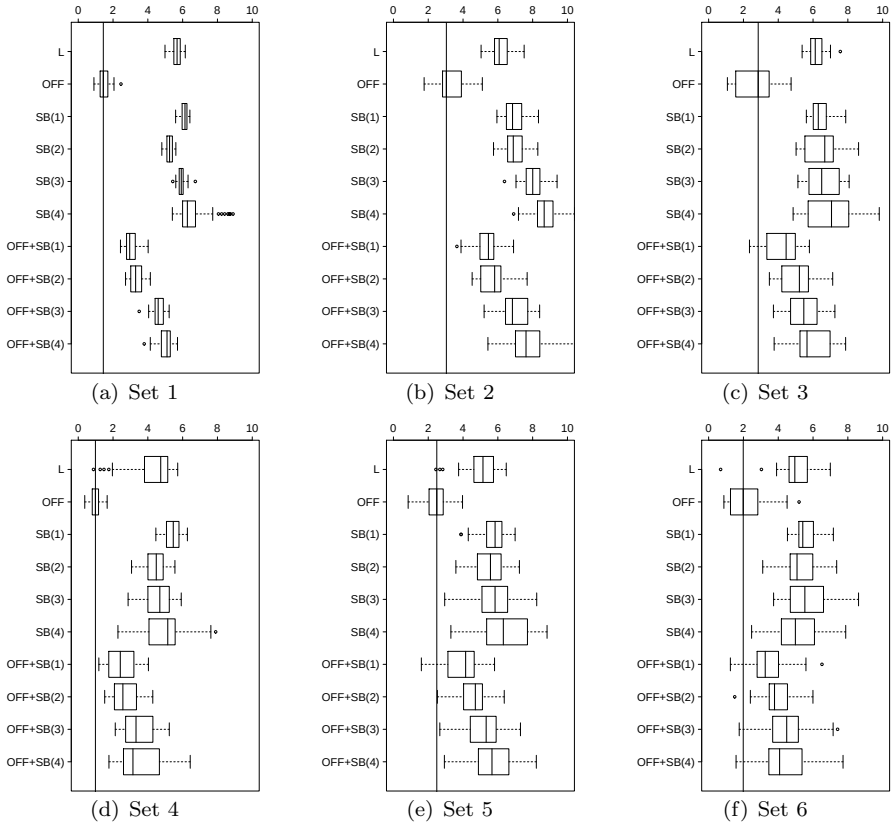
(d) Set 4 (e) Set 5 (f) Set 6

Figure 9: **Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

reported in Figures 9 and 10. We represent the average percentage error over the combinations of parameters suggested.

Experiments 4: long runs.

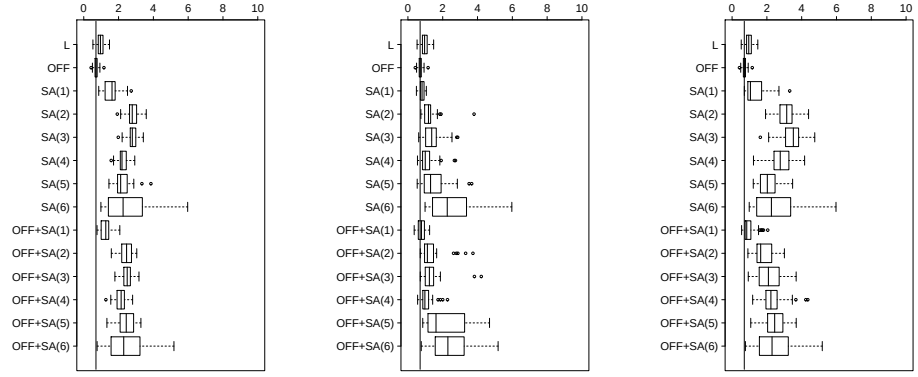
For understanding the impact on the results of the computational time used as stopping criterion, we execute an equivalent set of experiments running the six versions of $\mathcal{MM}$AS on instances of set 1 for 60, instead of 10, seconds. Figures 11 and 12 report the results achieved. The three cases of no *a priori* information, perfect *a posteriori* information, and realistic *a priori* information on the most important parameters to tune are depicted for the self-adaptive mechanism.



(a) Set 1 (b) Set 2 (c) Set 3

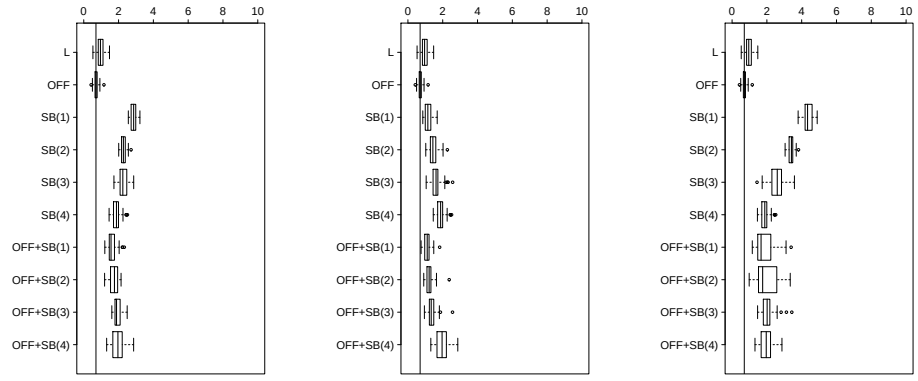
(d) Set 4 (e) Set 5 (f) Set 6

Figure 10: **Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning.** Runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.



(a) no *a priori* knowledge (b) perfect *a posteriori* knowledge (c) realistic *a priori* knowledge

Figure 11: **Results in long runs.** Runs of 60 seconds. Self-adaptation mechanism. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.



(a) no *a priori* knowledge (b) perfect *a posteriori* knowledge (c) realistic *a priori* knowledge

Figure 12: **Results in long runs.** Runs of 60 seconds. Search-based mechanism. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.