

Companion to Off-line *vs.* On-line tuning: *MAX-MIN* Ant System for the TSP

Paola Pellegrini¹, Thomas Stützle², and Mauro Birattari^{2*}

¹ Dipartimento di Matematica Applicata,
Università Ca' Foscari Venezia,
Venezia, Italia

² IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

In the paper [1] we compare the results achieved with off-line and on-line tuning approaches. In particular, we report the results of an experimental study based on *MAX-MIN* Ant System (*MMAS*) for the traveling salesman problem (TSP). In this framework, we analyze the impact on the performance of on-line tuning of the number of parameters adapted. Moreover, thanks to a social experiment carried out, we observe the effect of the expertise of the implementer on the behavior of the algorithm with on-line tuned parameters.

Here, the results of the whole experimental analysis are reported.

1 Experimental setup

We present an experimental analysis aiming at comparing the performance of off-line and on-line tuning in different experimental conditions. We apply the *MMAS* algorithm to the TSP. We consider twelve versions of the algorithm, according to the tuning procedure used:

1. *literature* (L): parameter values are set as suggested in the literature [2]. They are highlighted in Table 1. This set of experiments is run as a baseline for following comparisons;
2. *off-line* (OFF): the F-Race procedure [3, 4] is applied for fixing the values of the six parameters discussed above, which are then maintained fixed throughout all runs;
3. *self-adaptive on-line* (SAa): the self-adaptive adaptation mechanism [5] is used with ant-level parameters, starting from the parameter setting suggested in the literature [2]: each ant selects its own setting;
4. *self-adaptive on-line* (SA): the self-adaptive adaptation mechanism [5] is used, starting from the parameter setting suggested in the literature [2]: parameter setting is fixed at the beginning of each iteration, and after the colony has completed its activity pheromone is deposited on the edges connecting the parameter values used (or the ones with which the best-so-far solution was generated);

* 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.

5. *self-adaptive on-line* (SAc-b): the self-adaptive adaptation mechanism [5] with colony-level parameters with multiple-colony comparison, best solution, is used, starting from the parameter setting suggested in the literature [2]: parameter setting is fixed at the beginning of each iteration, and after the colony has completed its activity pheromone is deposited on the edges connecting the parameter values used in the iteration, among the last 25, in which the best solution was found;
6. *self-adaptive on-line* (SAc-m): the self-adaptive adaptation mechanism [5] with colony-level parameters with multiple-colony comparison, best solution, is used, starting from the parameter setting suggested in the literature [2]: parameter setting is fixed at the beginning of each iteration, and after the colony has completed its activity pheromone is deposited on the edges connecting the parameter values used in the iteration, among the last 25, in which the average solution cost was the lowest;
7. *search-based on-line* (SB): the search-based adaptation mechanism [6] is used, starting from the parameter setting suggested in the literature [2];
8. *off-line + self-adaptive on-line* (OFF+SAa): the self-adaptive adaptation mechanism [5] is used with ant-level parameters, starting from the parameter setting returned by F-Race [3, 4]: each ant selects its own setting;
9. *off-line + self-adaptive on-line* (OFF+SA): the self-adaptive adaptation mechanism [5] is used, starting from the parameter setting returned by F-Race [3, 4]: parameter setting is fixed at the beginning of each iteration, and after the colony has completed its activity pheromone is deposited on the edges connecting the parameter values used (or the ones with which the best-so-far solution was generated);
10. *off-line + self-adaptive on-line* (OFF+SAc-b): the self-adaptive adaptation mechanism [5] with colony-level parameters with multiple-colony comparison, best solution, is used, starting from the parameter setting returned by F-Race [3, 4]: parameter setting is fixed at the beginning of each iteration, and after the colony has completed its activity pheromone is deposited on the edges connecting the parameter values used in the iteration, among the last 25, in which the best solution was found;
11. *off-line + self-adaptive on-line* (OFF+SAc-m): the self-adaptive adaptation mechanism [5] with colony-level parameters with multiple-colony comparison, best solution, is used, , starting from the parameter setting returned by F-Race [3, 4]: parameter setting is fixed at the beginning of each iteration, and after the colony has completed its activity pheromone is deposited on the edges connecting the parameter values used in the iteration, among the last 25, in which the average solution cost was the lowest;
12. *off-line + search-based on-line* (OFF+SB): the search-based adaptation mechanism [6] is used, starting from the parameter setting returned by F-Race [3, 4].

When an on-line approach is used, we solve each instance adapting alternatively one, two, ... , six parameters. In this way, we study how results change if the number of parameters adapted increases. Moreover, for each number of parameters adapted, we register the performance of the algorithm for all the possible

Table 1. Values that can be chosen for each parameter. The values reported in bold type are the ones suggested in the literature [2]. They are the values used in L setting.

parameter	values	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 6	q_0	0.0 , 0.25, 0.5, 0.75, 0.9
m	5, 10, 25 , 50, 100	n	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 a computation time limit of 10 CPU seconds.

set	number of nodes	spatial distribution	F-Race selection					
			α	β	ρ	q_0	m	n
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

combinations of parameters. In the rest of the paper, the name of all versions that include on-line tuning are followed by a number between parenthesis indicating how many parameters are adapted. The adaptation schemes are included into the ACOTSP software [7].

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 self-adaptation scheme) are shown 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 [8]. 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; the most heterogeneous set is set 6.

For each set of instances, a separate run of F-race is performed using 1000 training instances. The instances used for the tuning and the experimental phase are randomly selected, and the two sets are disjoint. All combinations of the values reported in Table 1 are considered as candidate settings. Hence, a total of 15,625 configurations is tested, on a maximum total number of runs equal to 156,250. The computation time available for each run is equal to the one considered in the experiments.

We executed experiments with two different termination criteria, 10 and 60 CPU seconds as measured on Xeon E5410 quad core 2.33GHz processors with 2x6 MB L2-Cache and 8 GB RAM, running under the Linux Rocks Cluster

Distribution. The code is compiled with `gcc`, version 3.4. In 10 CPU seconds, the ACOTSP code generates about 2 500-3 000 solutions for instances of set 1. The results presented in Section 2 depict the percentage error with respect to the optimal solution for 44 new test instances of each set. We performed one run on each instance for each parameter configuration [?,9].

We analyzed the performance of each on-line version also considering as stopping criterion the construction of as many solutions as the *off-line* version. The results achieved are not qualitatively different from the ones obtained considering time as stopping criterion. In the following we show that on-line tuning is not a convenient choice in the setting considered. This result is not due to the time overhead implied by parameter adaptation, but due to the nature of the adaptation itself.

2 Experimental results

The aim of this experimental analysis is making a comparison between off-line and on-line tuning in different contexts.

We compare the three tuning approaches simulating different levels of knowledge on the importance on parameter importance for on-line tuning

1. No *a priori* knowledge on parameter importance for on-line tuning. In this case, we can expect the results to be similar to the average computed across all possible combinations for each number of parameters tuned. In the representation of these aggregate results, the boxplots summarize the average results in terms of percentage error.
2. Perfect *a posteriori* knowledge on parameter importance for on-line tuning. In this 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. In the Figures presenting these results, the boxplots summarize the results in terms of percentage error.
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 reported in Figures. We represent the average percentage error over the combinations of parameters suggested.

Moreover, 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 *MMAS* on instances of set 1 for 60, instead of 10, seconds.

We present the results for each on-line tuning approach separately. Moreover we use two stopping criteria alternatively: first a fix computational time

is allowed, either 10 or 60 seconds; second the number of objective function evaluations is taken into account, and the limit is set equal to the number of evaluations performed by the *off-line* version on each instance

2.1 Self-adaptation: computation time as stopping criterion

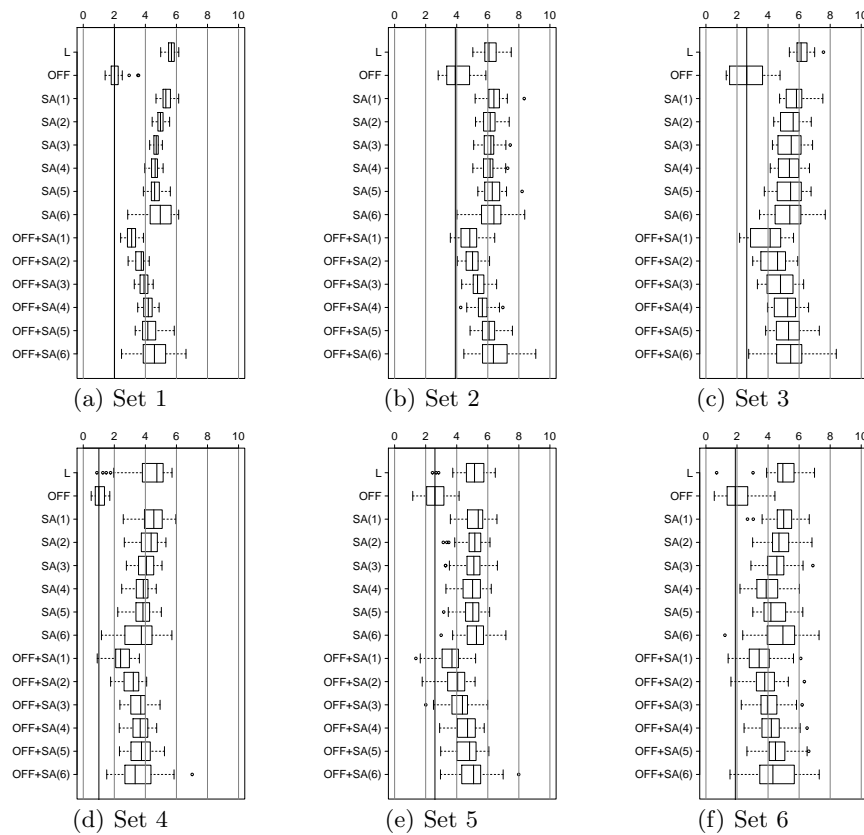


Fig. 1. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median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

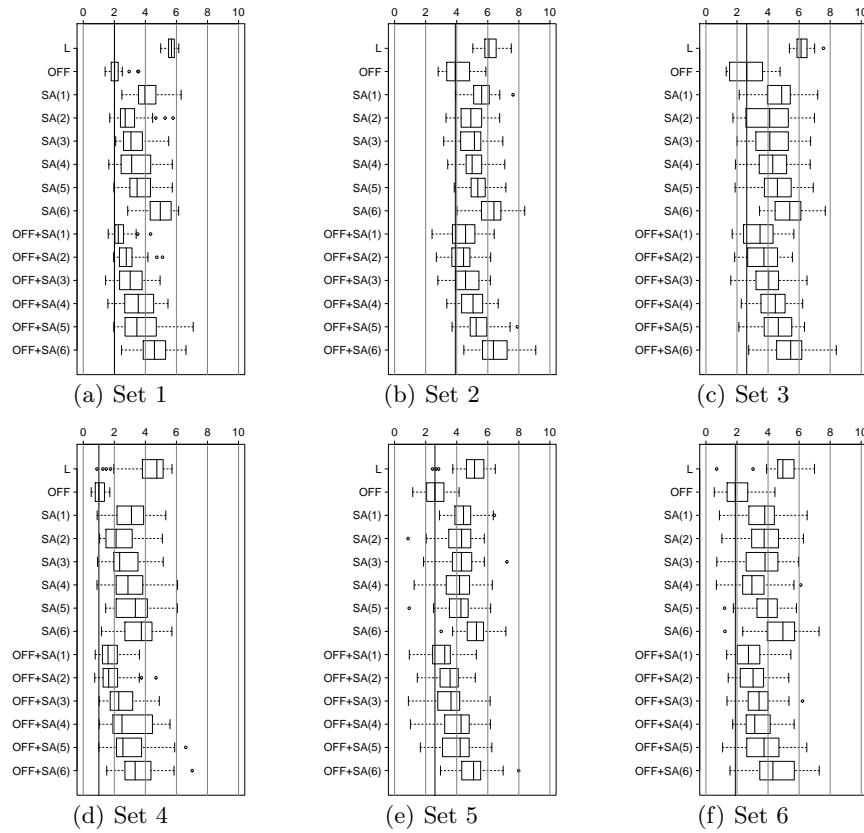


Fig. 2. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the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

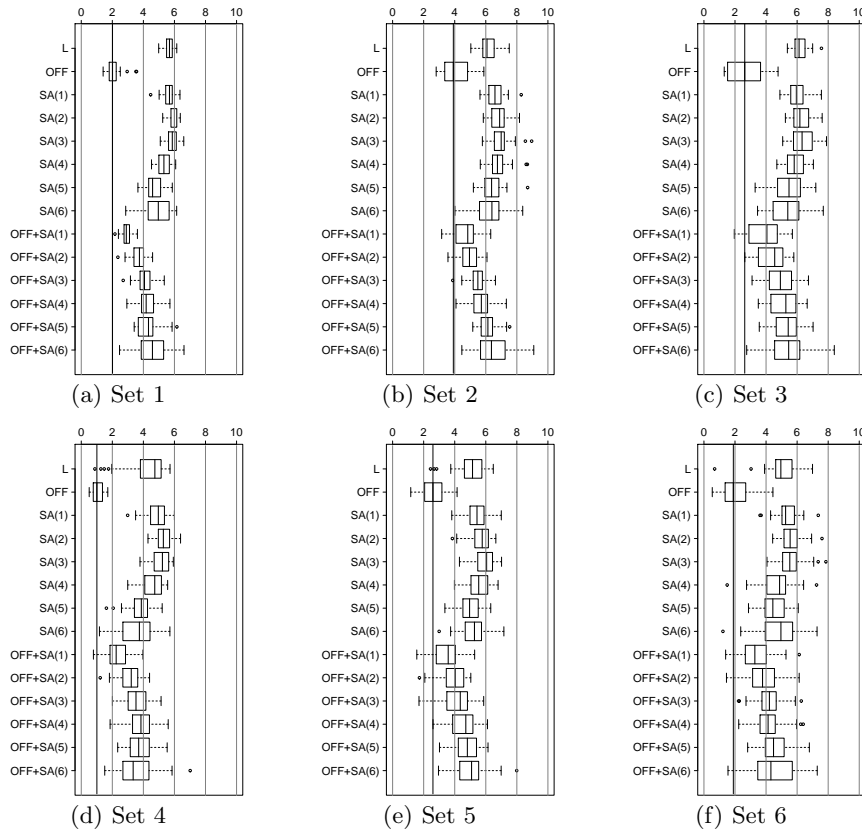


Fig. 3. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the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

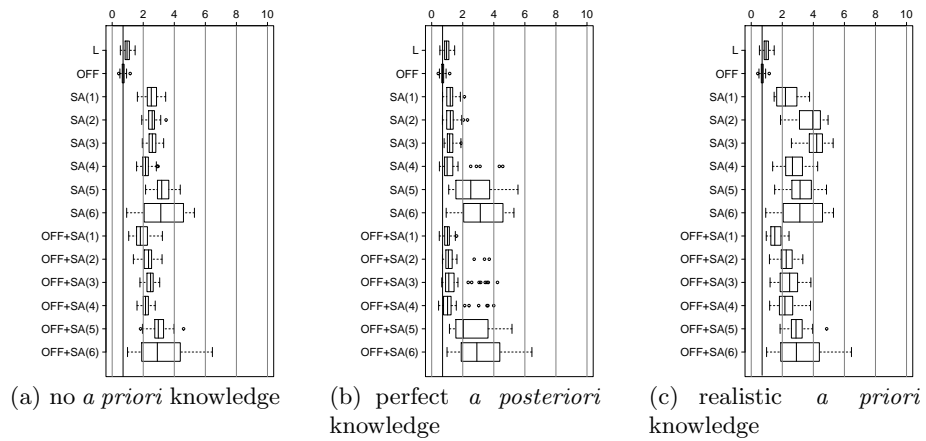


Fig. 4. Results in long runs. Runs of 60 seconds. Self-adaptation mechanism. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.2 Self-adaptation with multiple-colony comparison, best solution: computation time as stopping criterion

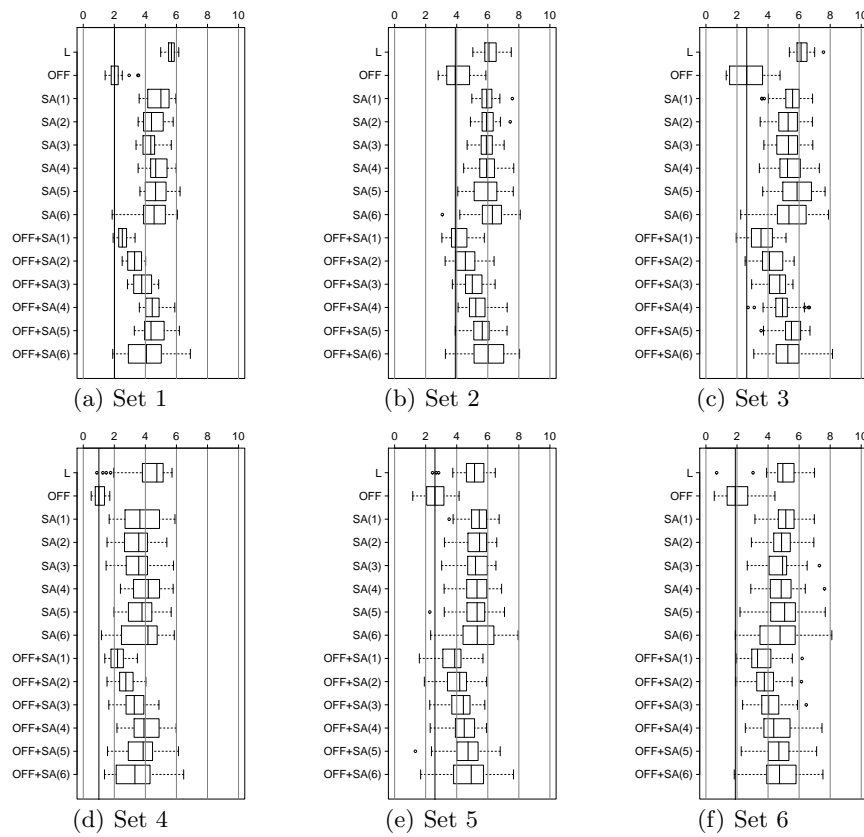


Fig. 5. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one.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

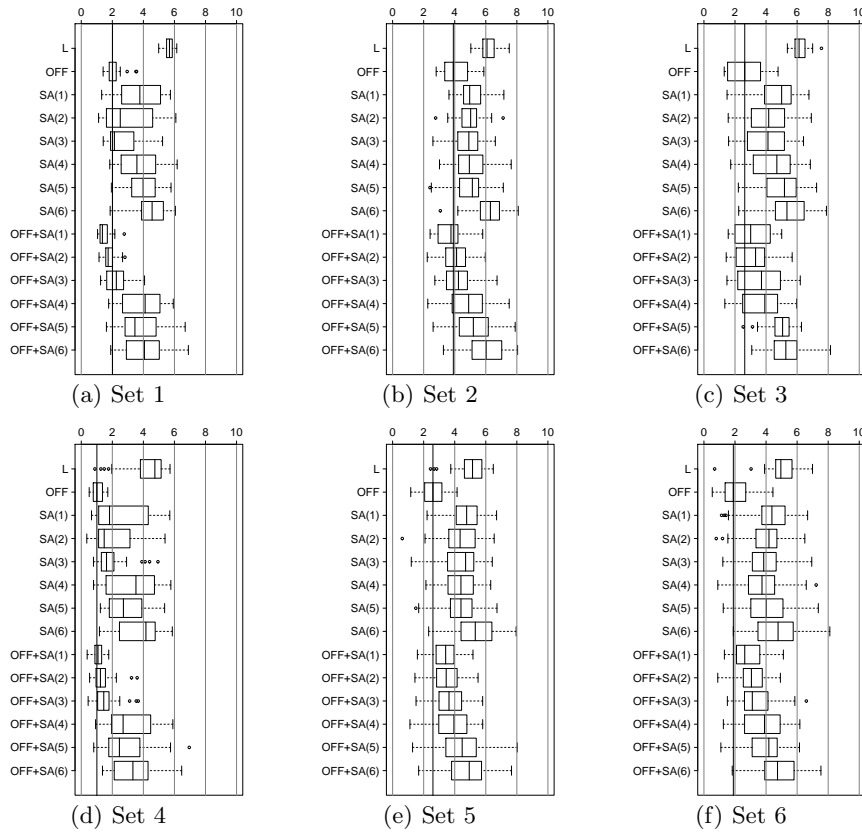


Fig. 6. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

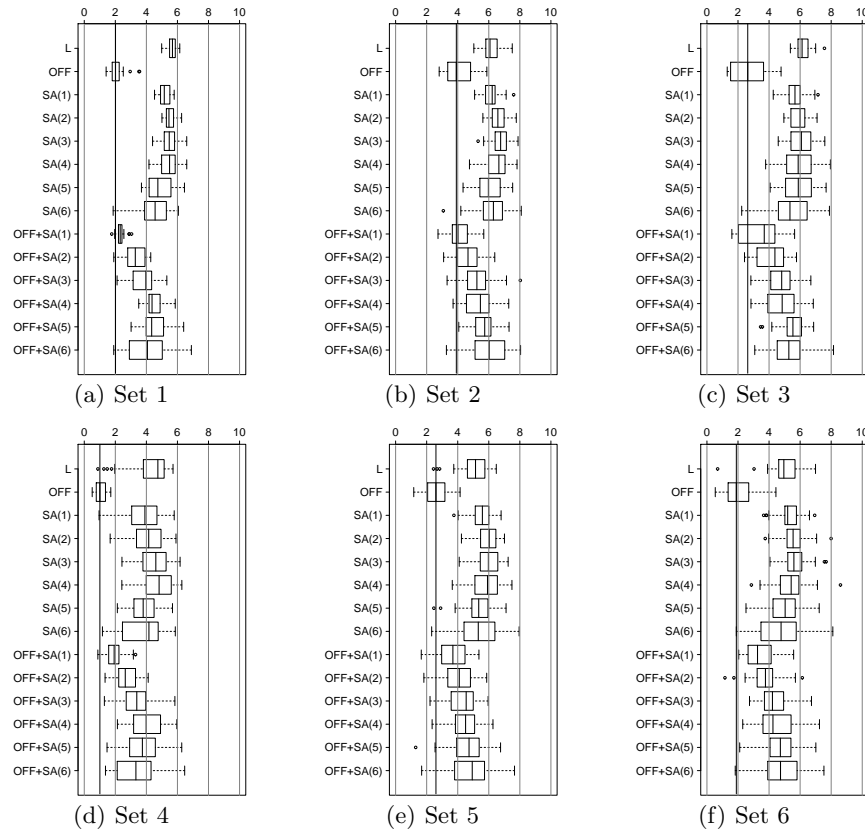


Fig. 7. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

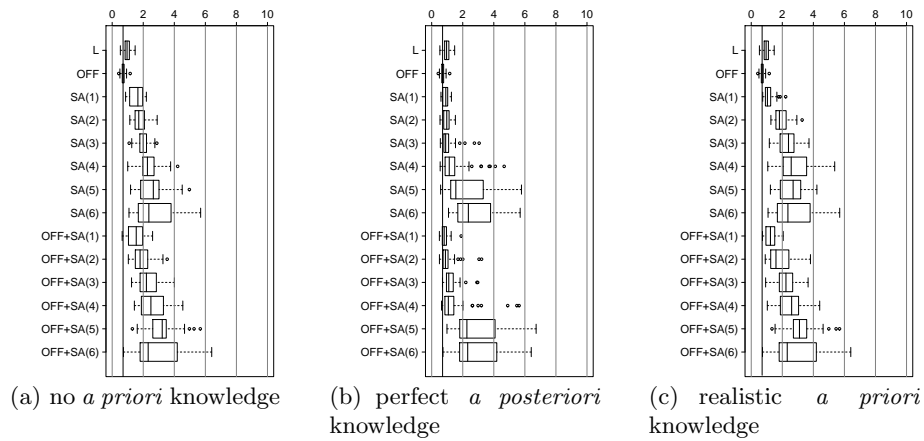


Fig. 8. Results in long runs. Runs of 60 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.3 Self-adaptation with multiple-colony comparison, average solution: computation time as stopping criterion

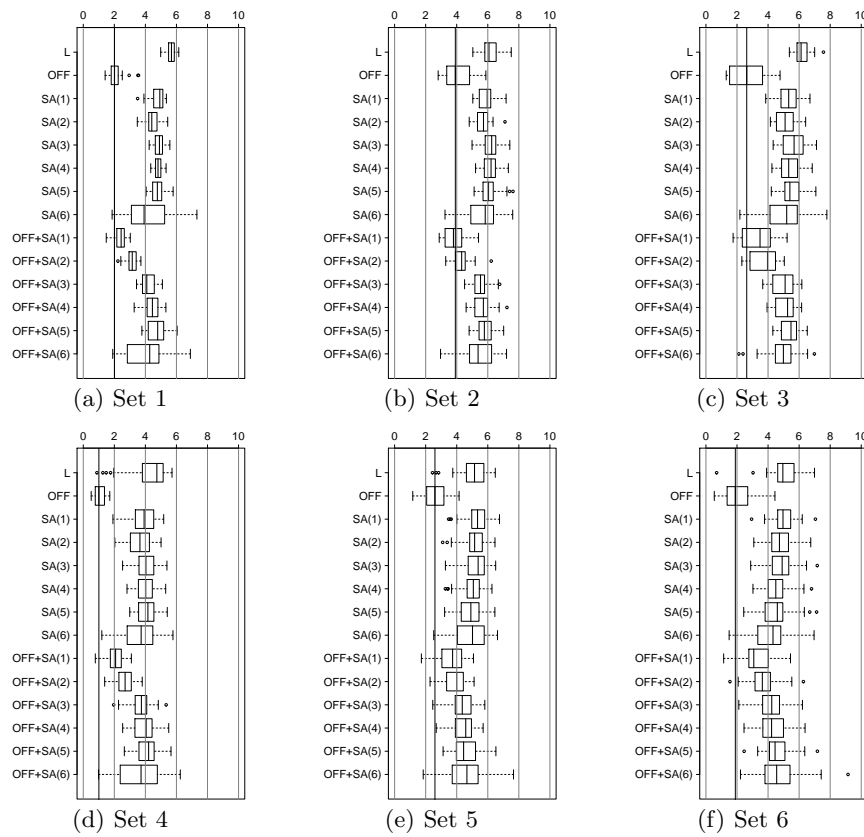


Fig. 9. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

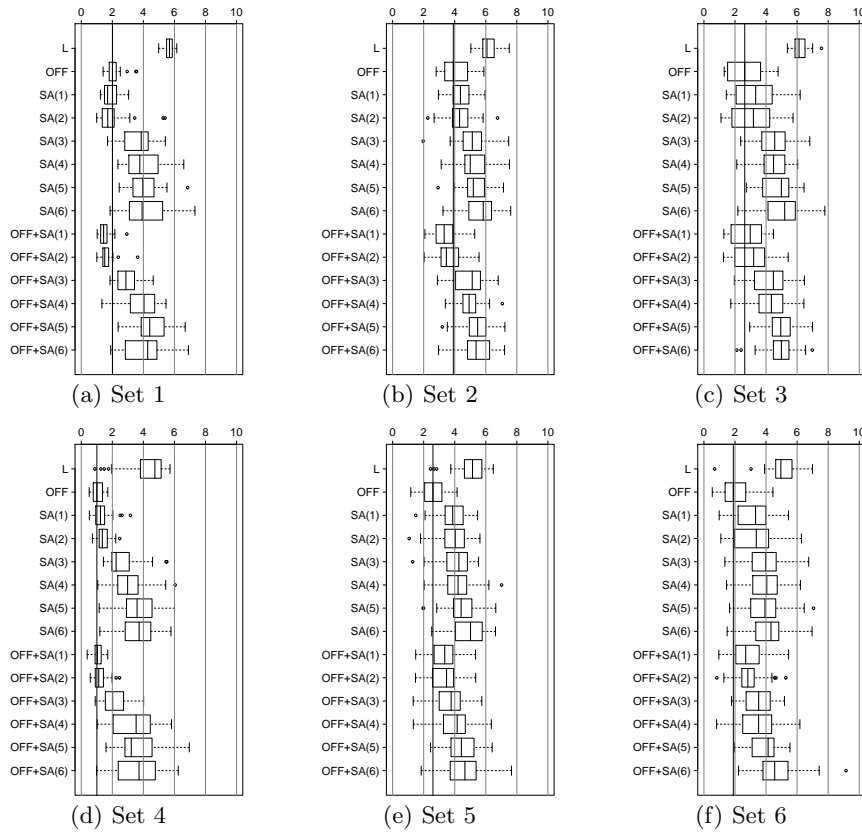


Fig. 10. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

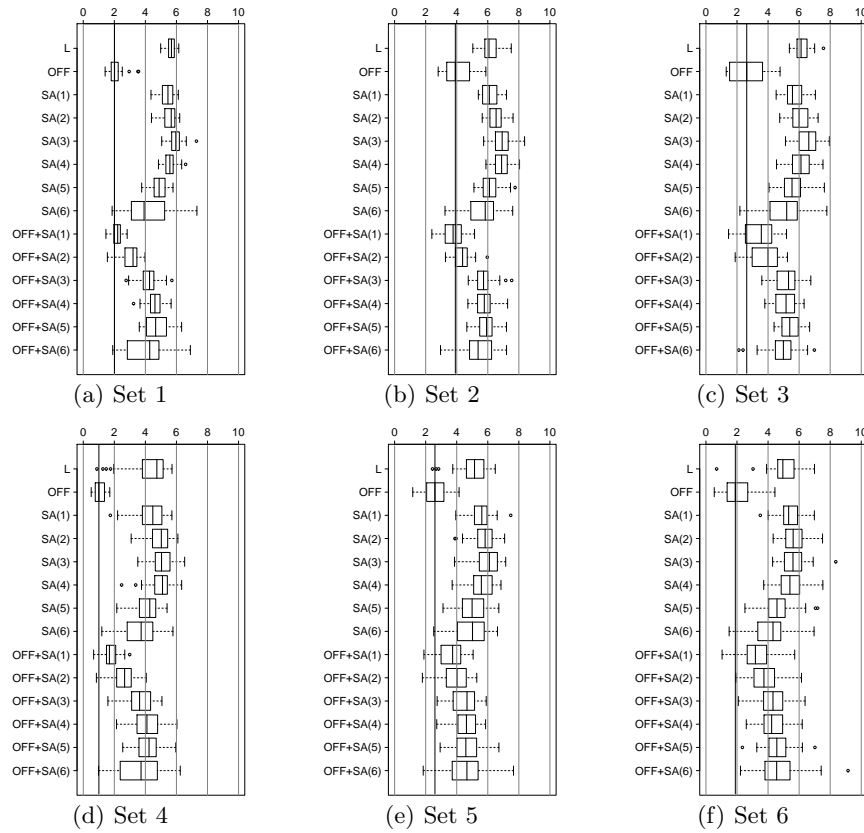


Fig. 11. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

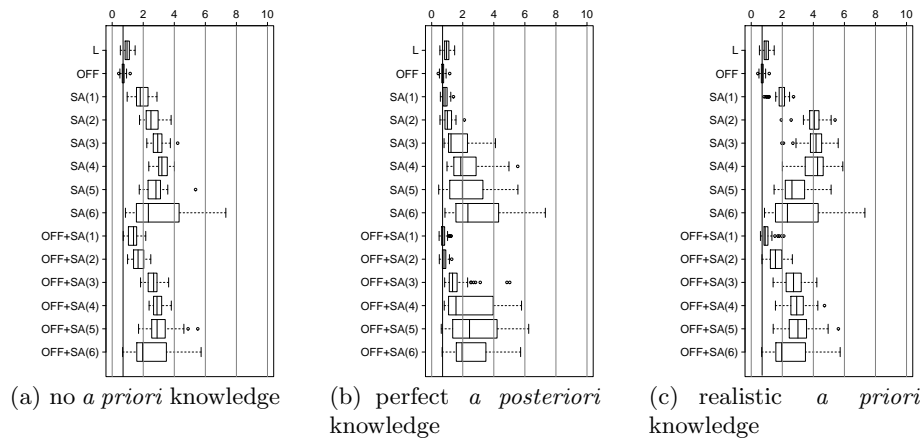


Fig. 12. Results in long runs. Runs of 60 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.4 Self-adaptation with ant-level parameters: computation time as stopping criterion

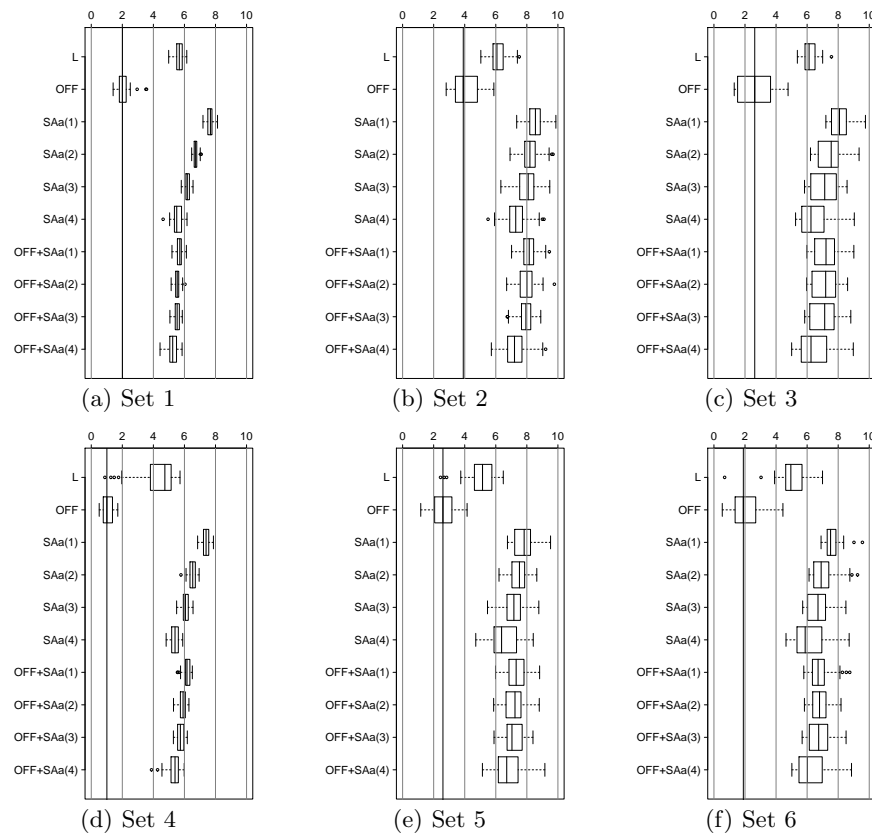


Fig. 13. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with ant-level parameters. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

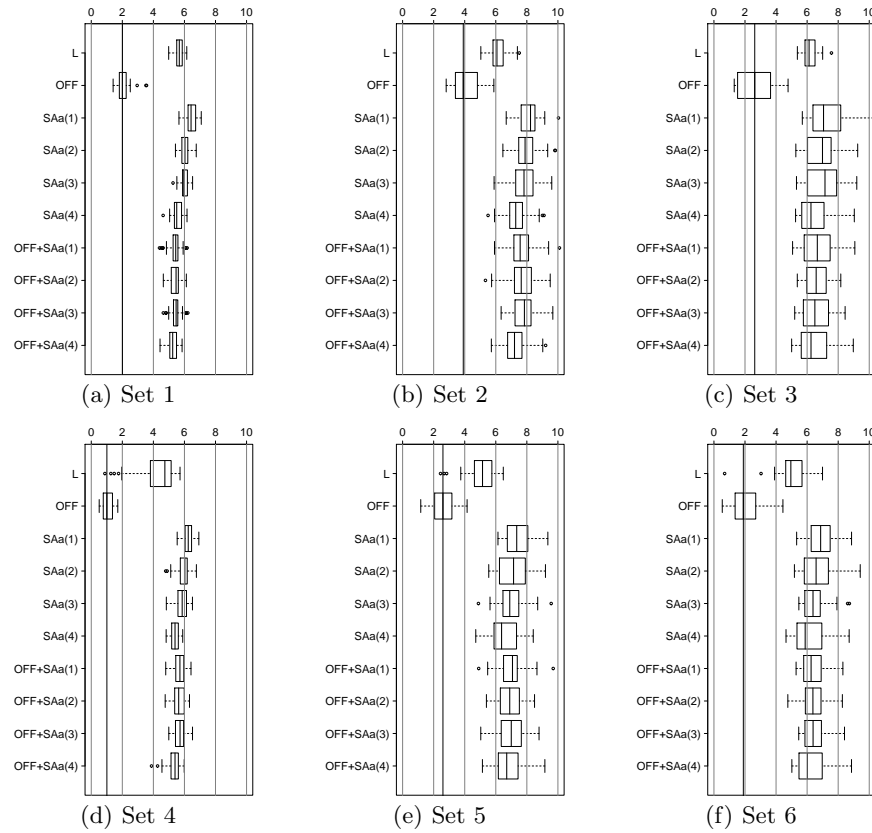


Fig. 14. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with ant-level parameters. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

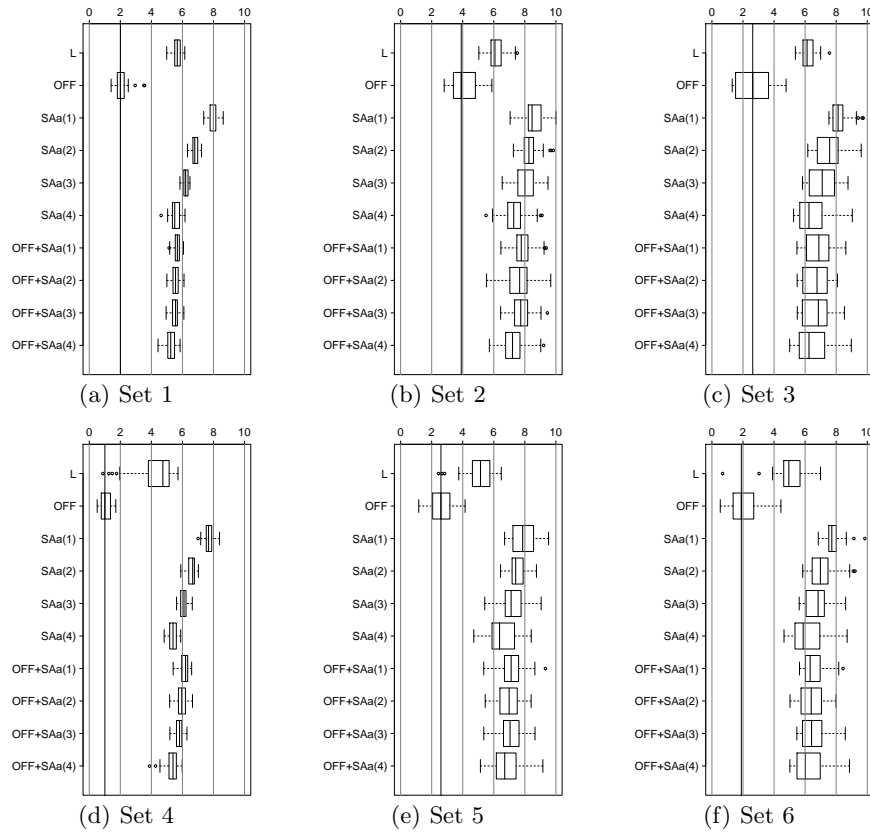


Fig. 15. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Self-adaptation mechanism with ant-level parameters. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

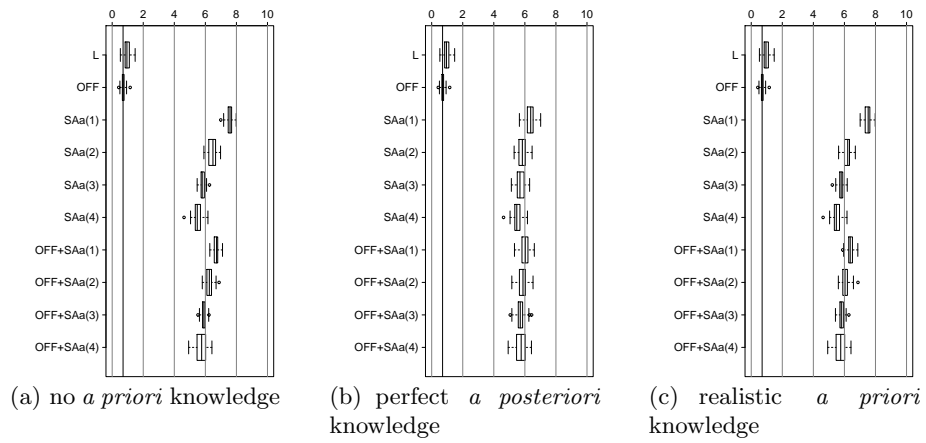


Fig. 16. Results in long runs. Runs of 60 seconds. Self-adaptation mechanism with ant-level parameters. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.5 Search-based adaptation: computation time as stopping criterion

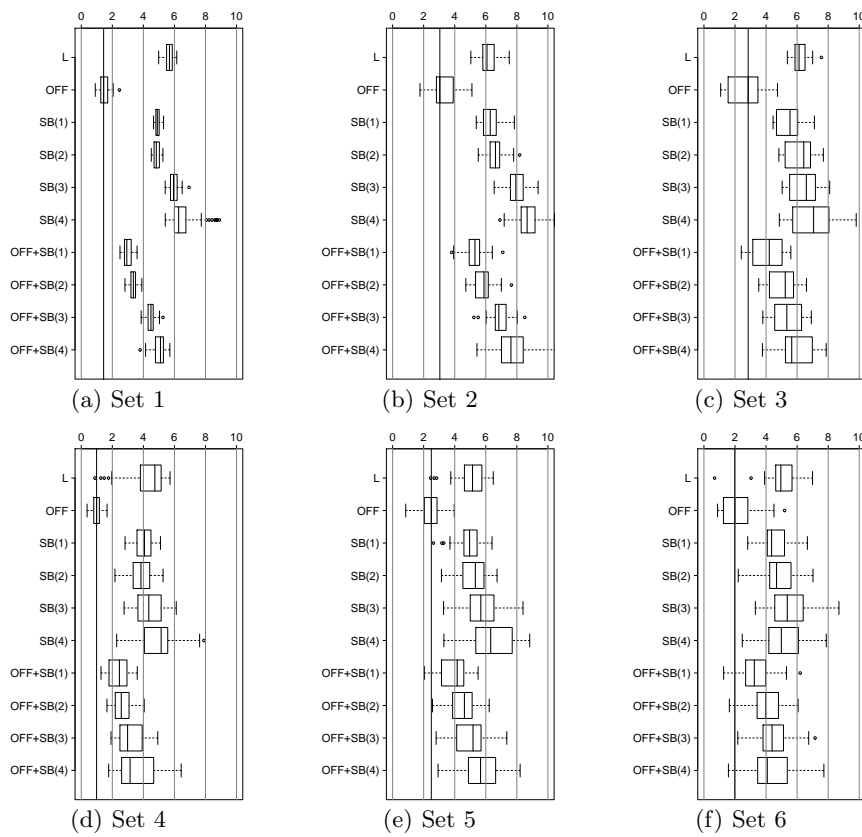


Fig. 17. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

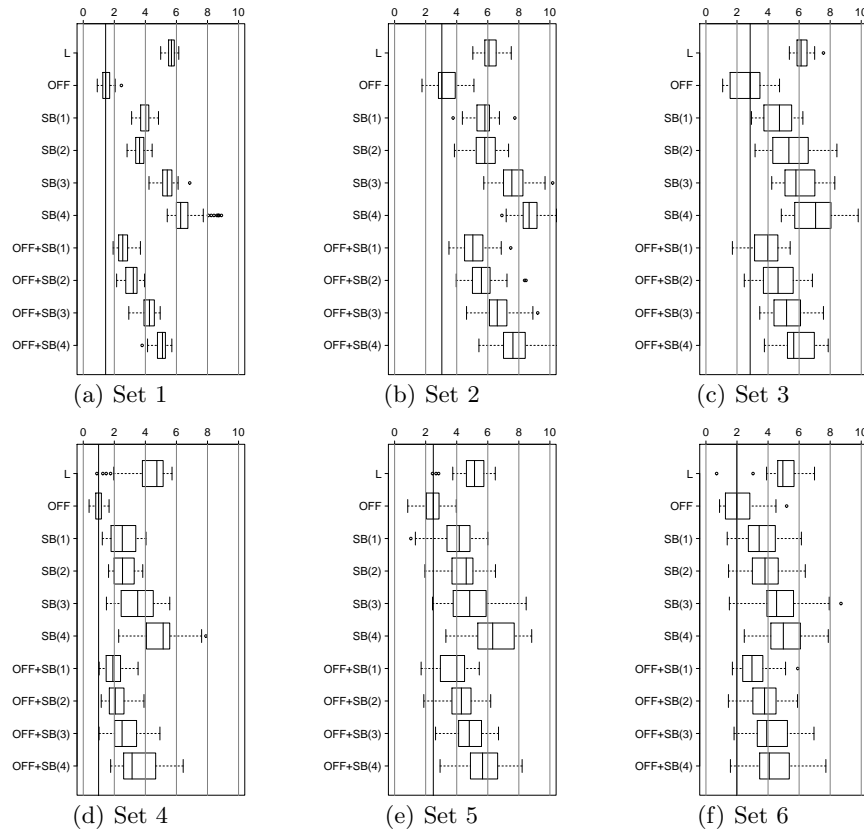


Fig. 18. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

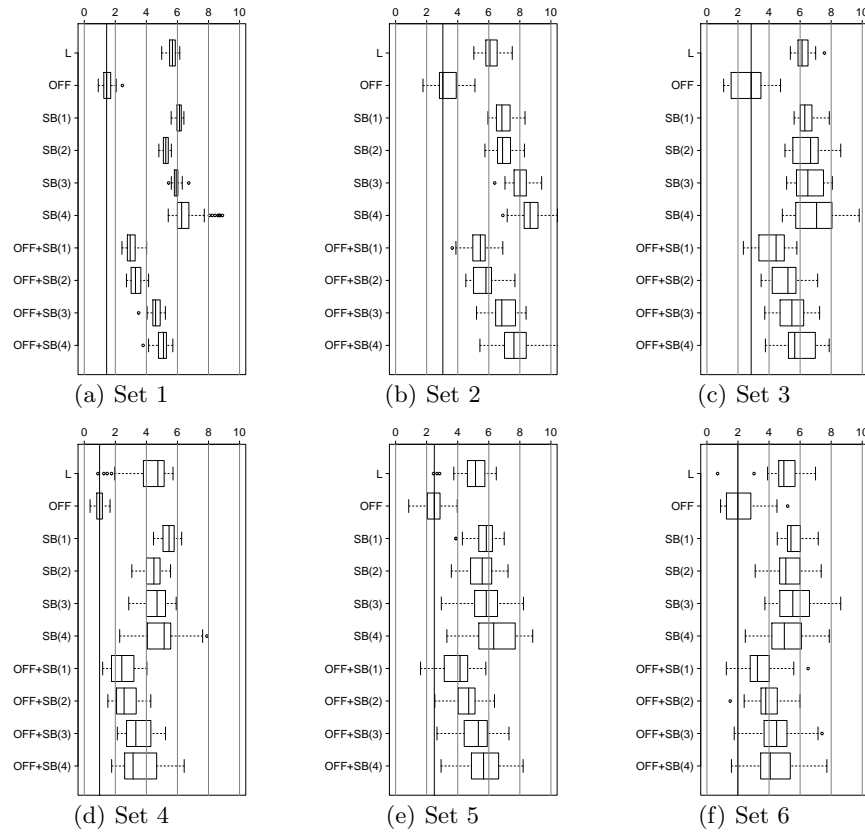


Fig. 19. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

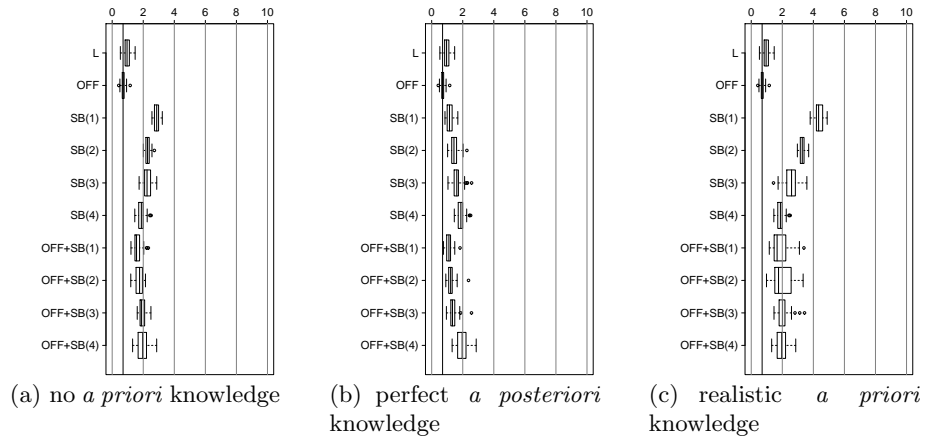


Fig. 20. Results in long runs. Runs of 60 seconds. Search-based mechanism. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.6 Self-adaptation: number of objective function evaluations as stopping criterion

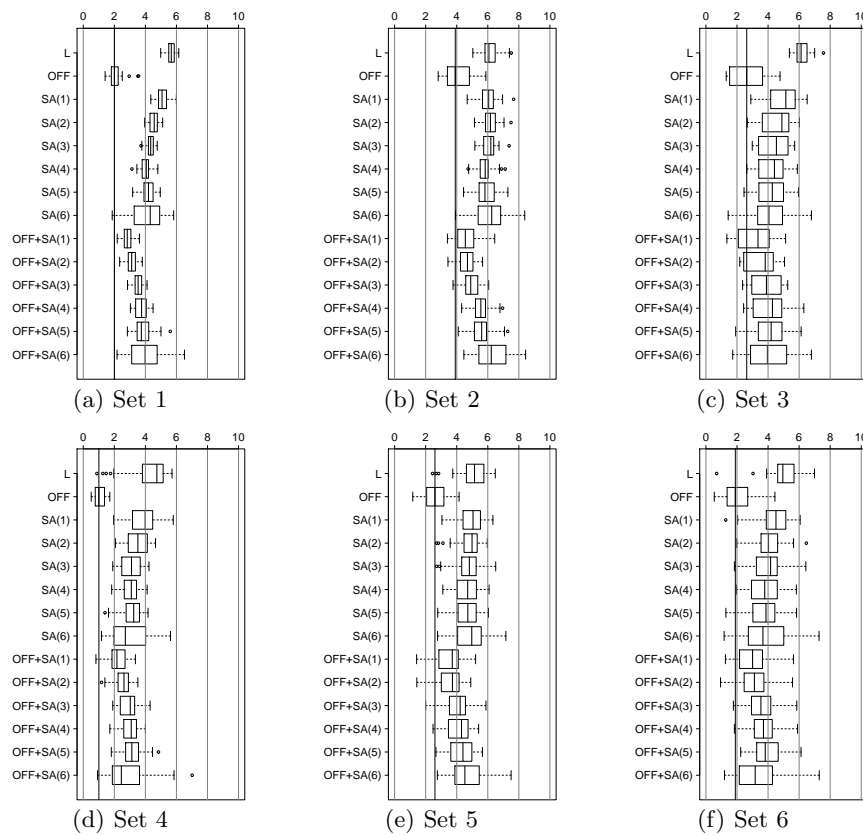


Fig. 21. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage error. 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.

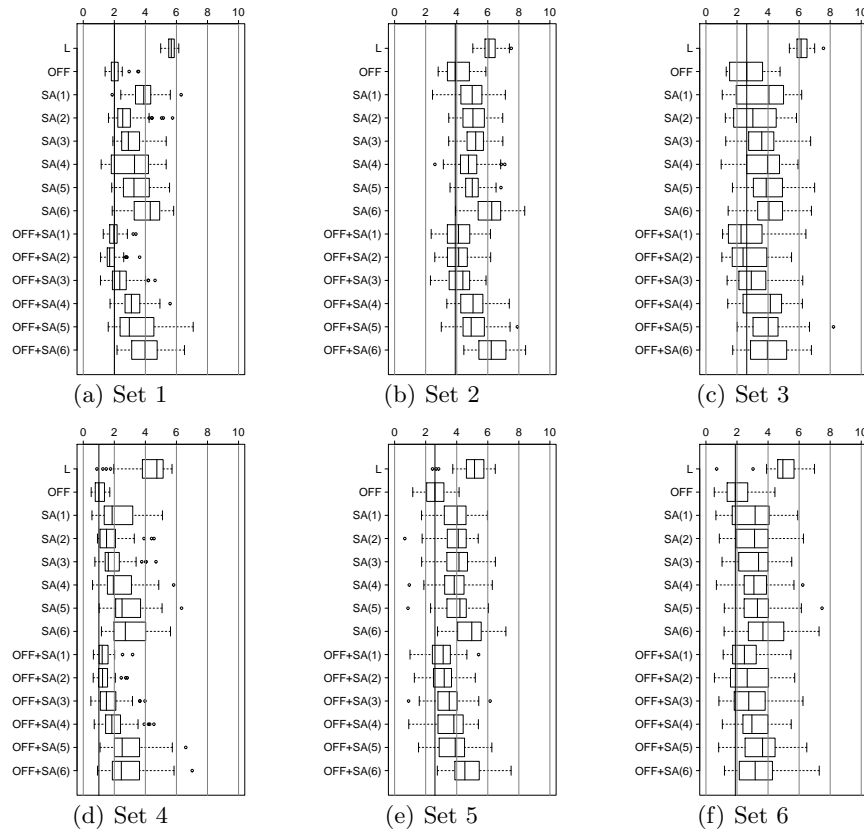


Fig. 22. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage 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.

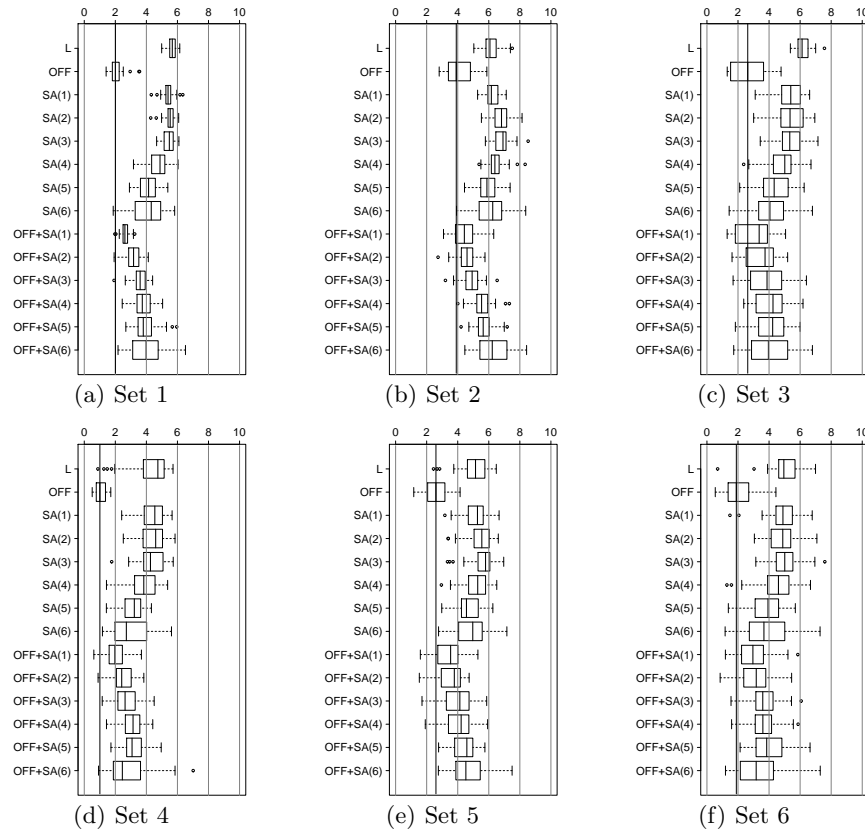


Fig. 23. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism. The horizontal axis represents the percentage 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.

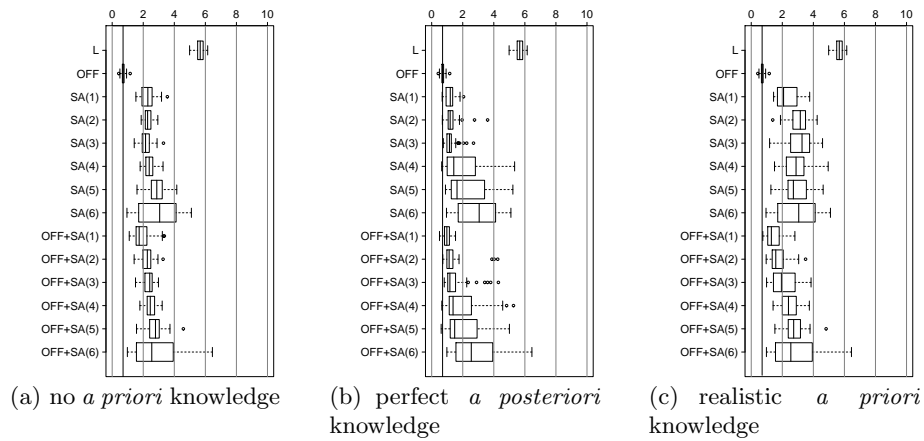


Fig. 24. Results in long runs. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 60 seconds. Self-adaptation mechanism. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.7 Self-adaptation with multiple-colony comparison, best solution: number of objective function evaluations as stopping criterion

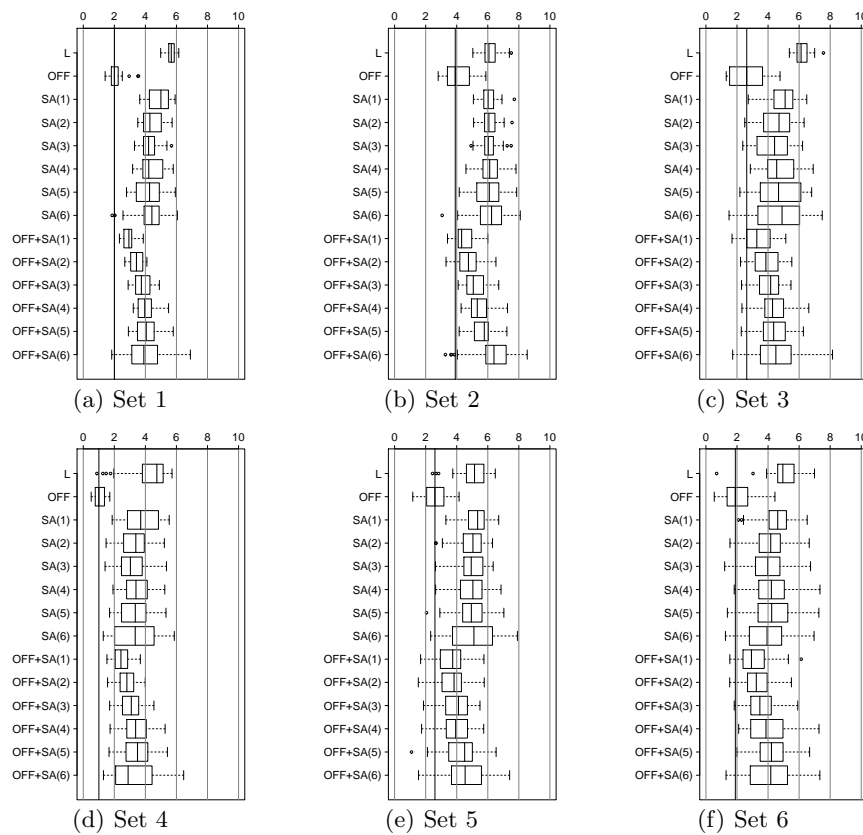


Fig. 25. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. The 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.

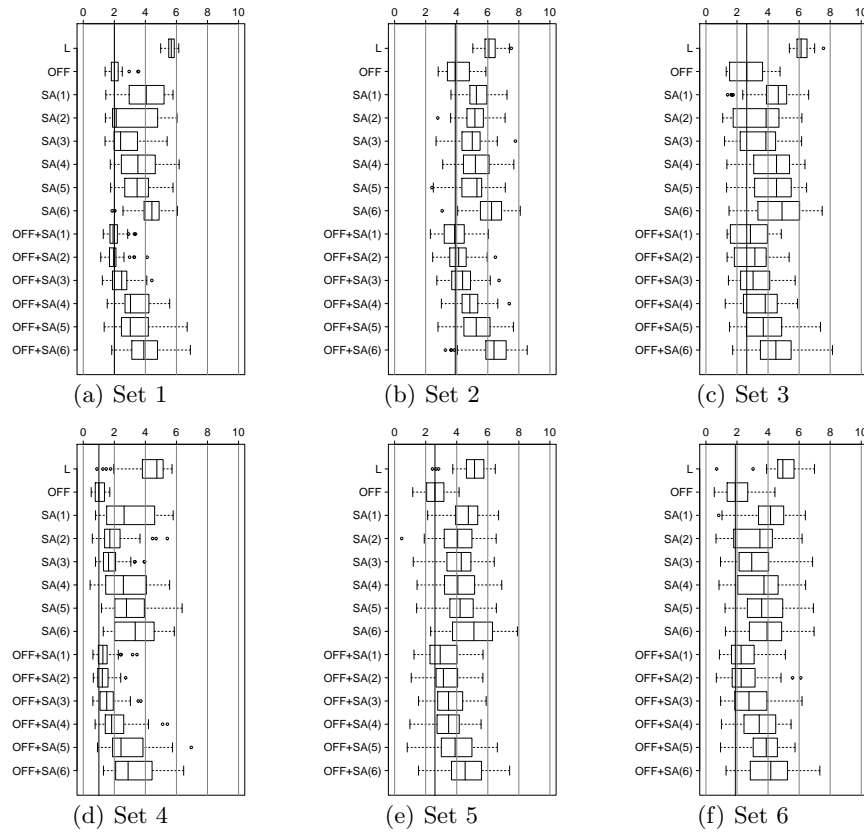


Fig. 26. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. 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.

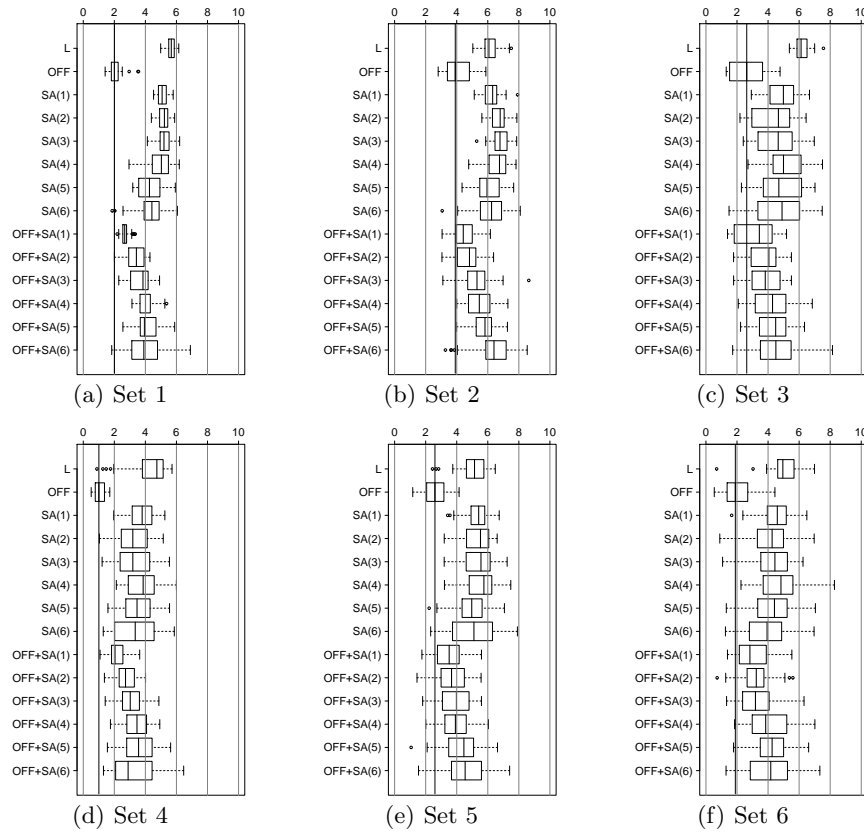


Fig. 27. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. 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.

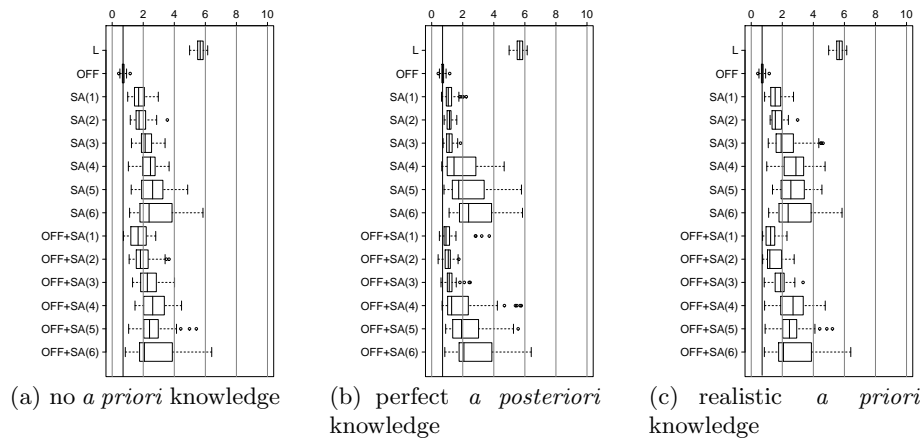


Fig. 28. Results in long runs. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 60 seconds. Self-adaptation mechanism with multiple-colony comparison, best solution. Instances of set 1. The horizontal axis represents the percentage error. 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.

2.8 Self-adaptation with multiple-colony comparison, average solution: number of objective function evaluations as stopping criterion

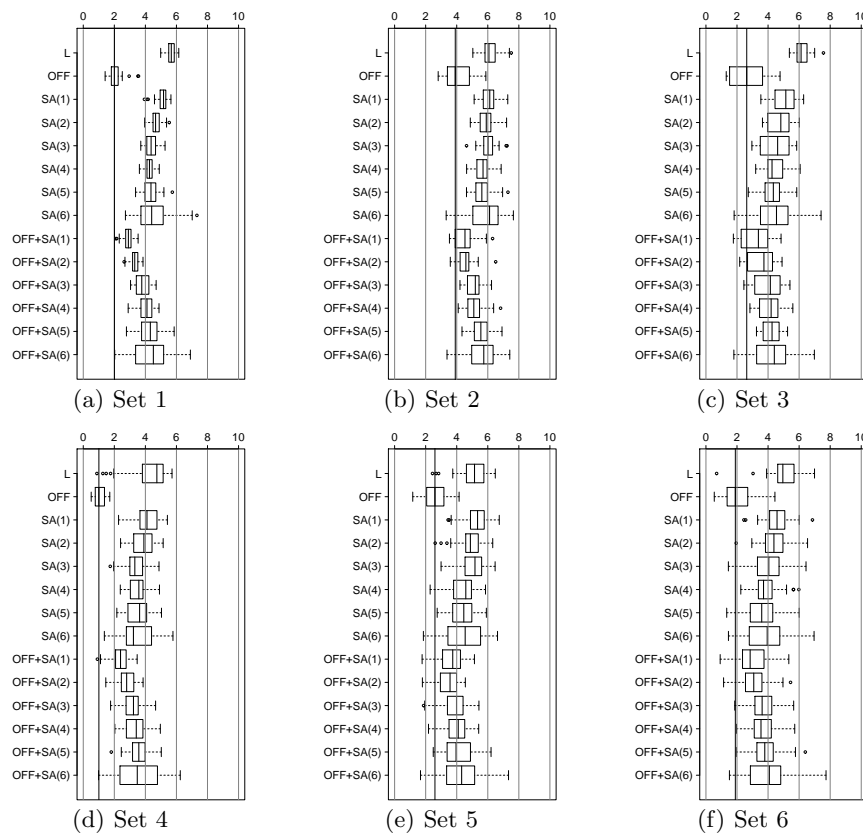


Fig. 29. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. 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.

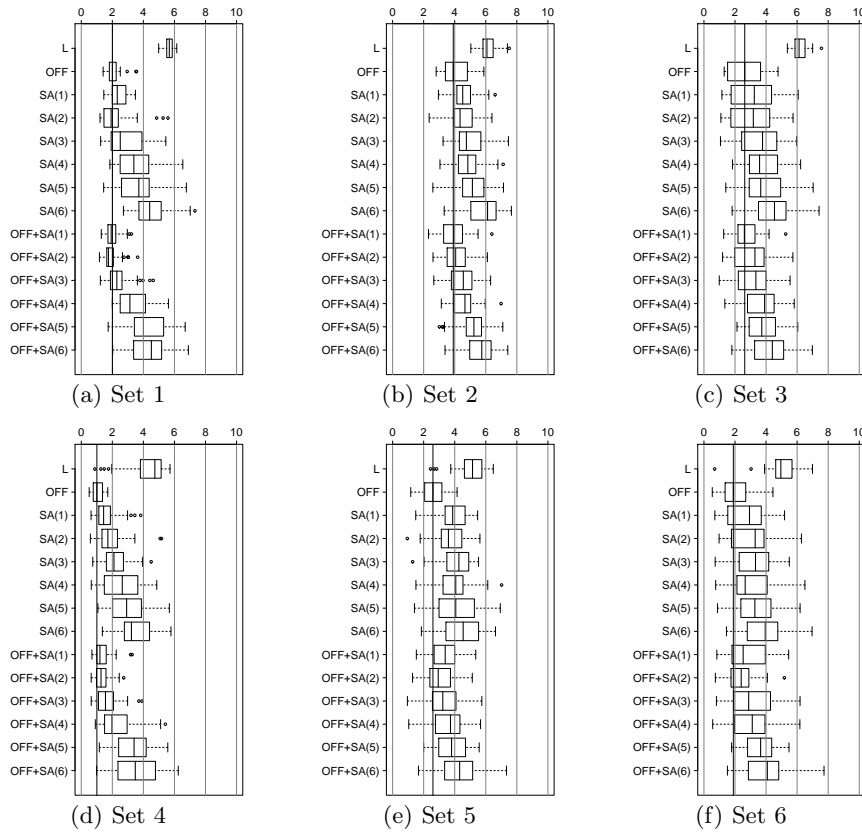


Fig. 30. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. 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.

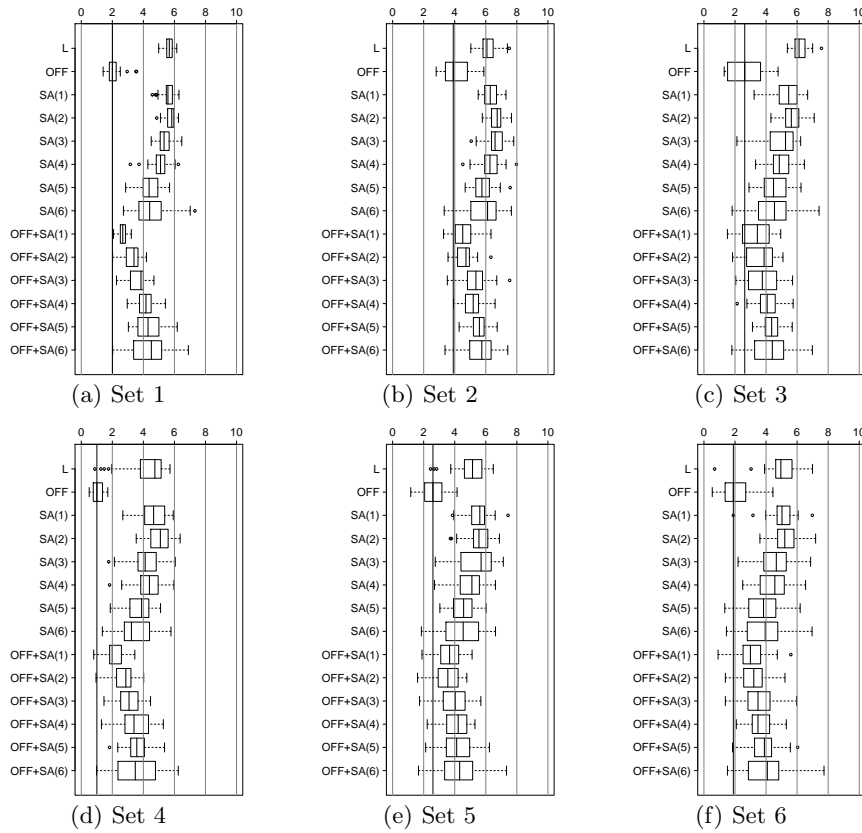


Fig. 31. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. 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.

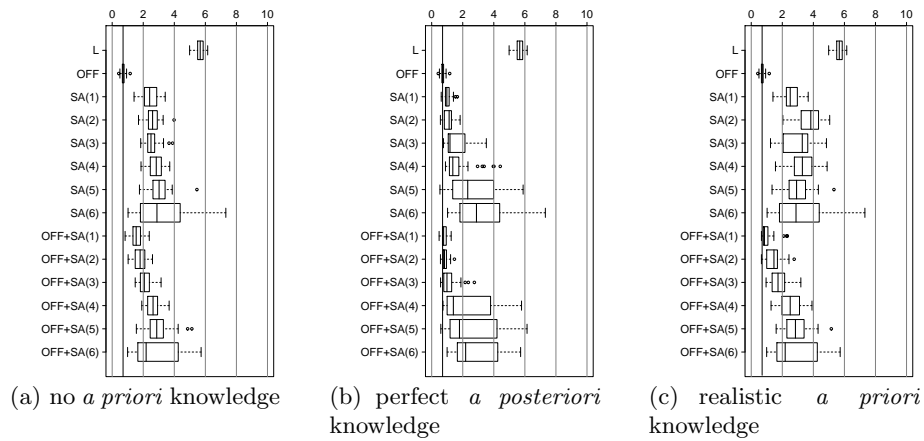


Fig. 32. Results in long runs. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 60 seconds. Self-adaptation mechanism with multiple-colony comparison, average solution. Instances of set 1. The horizontal axis represents the percentage 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.

2.9 Self-adaptation with ant-level parameters: number of objective function evaluations as stopping criterion

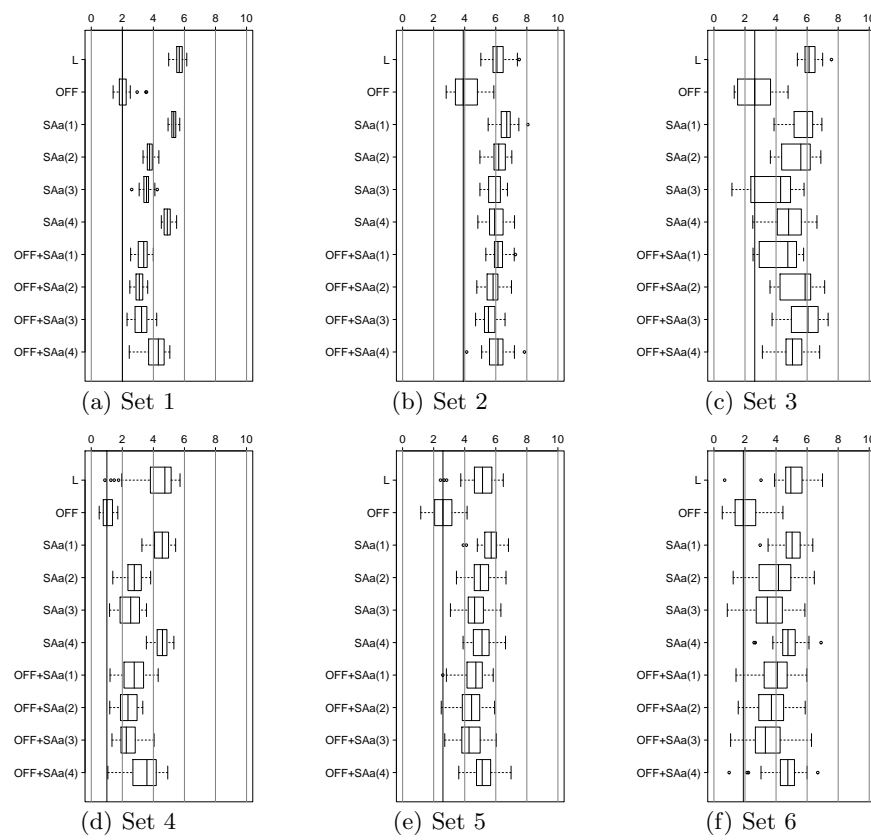


Fig. 33. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with ant-level parameters. The horizontal axis represents 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.

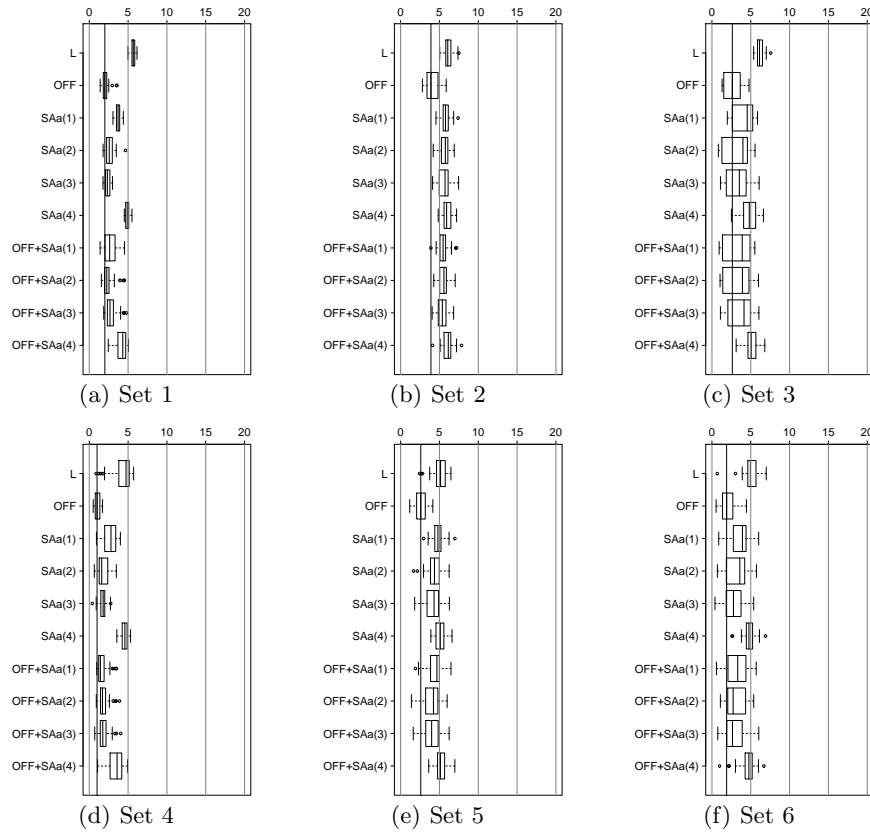


Fig. 34. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with ant-level parameters. The horizontal 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.

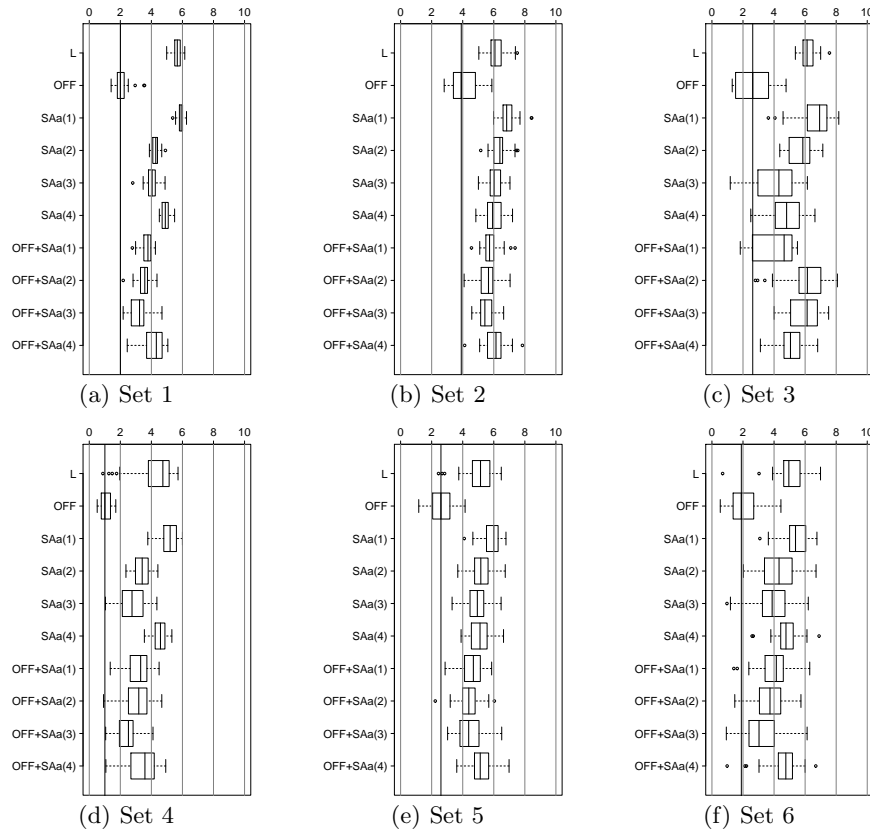


Fig. 35. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Self-adaptation mechanism with ant-level parameters. The horizontal axis 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.

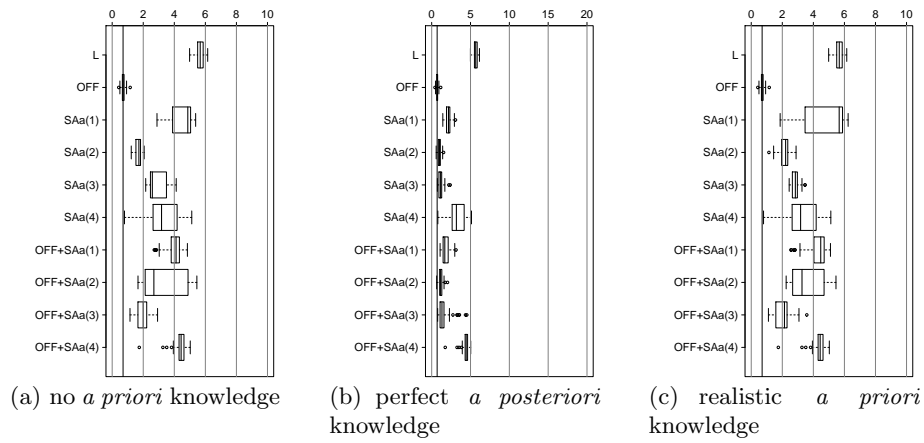


Fig. 36. Results in long runs. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 60 seconds. Self-adaptation mechanism with ant-level parameters. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

2.10 Search-based adaptation: number of objective function evaluations as stopping criterion

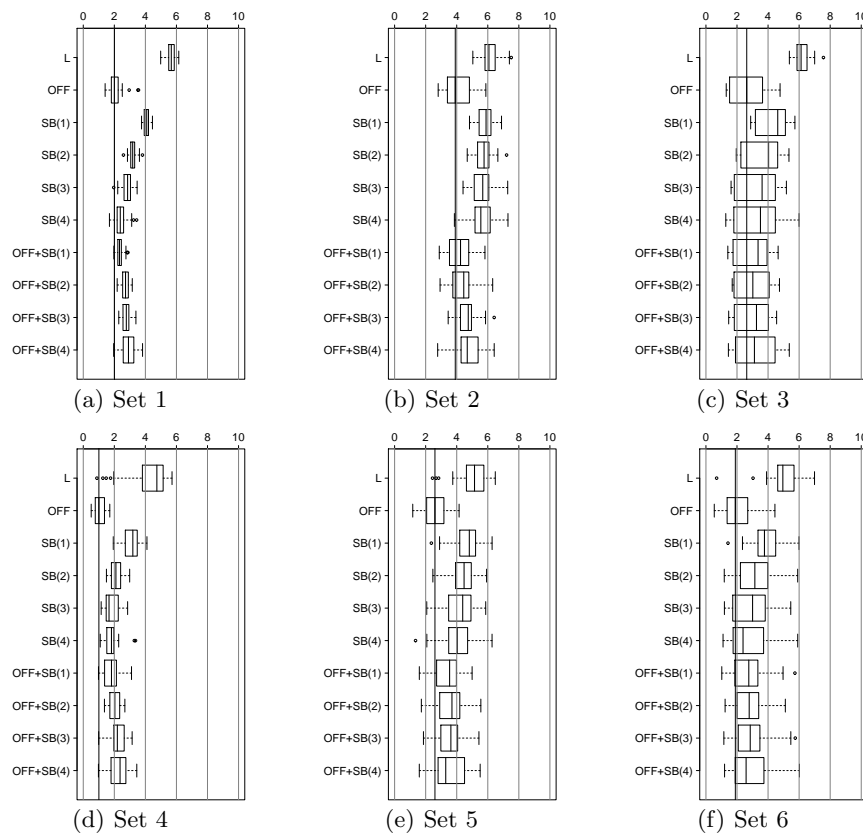


Fig. 37. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage error. 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.

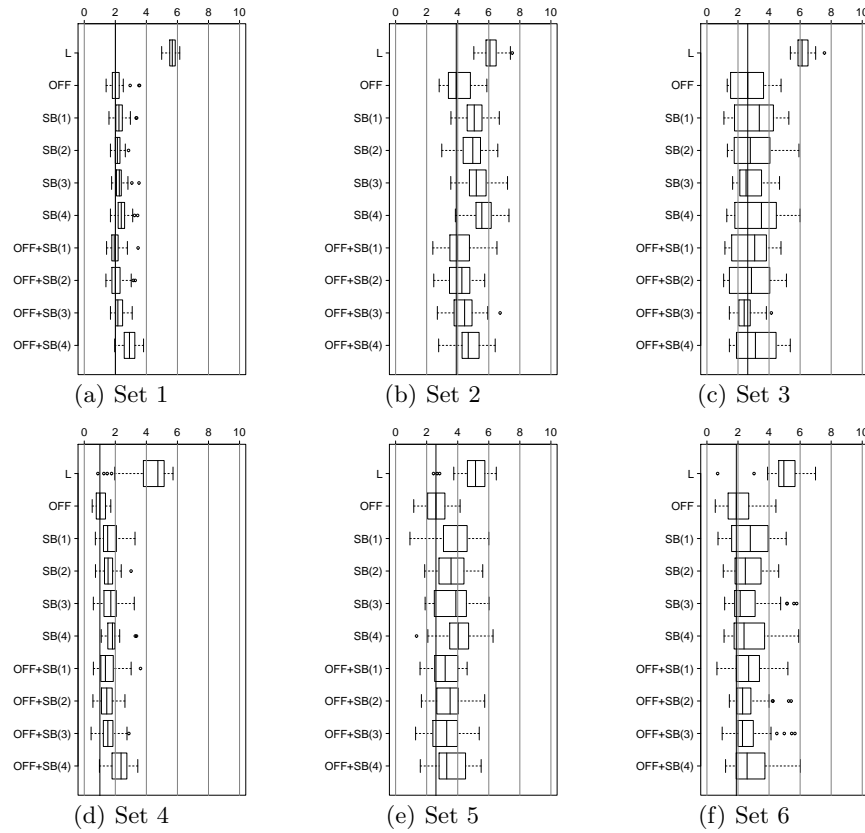


Fig. 38. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage 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.

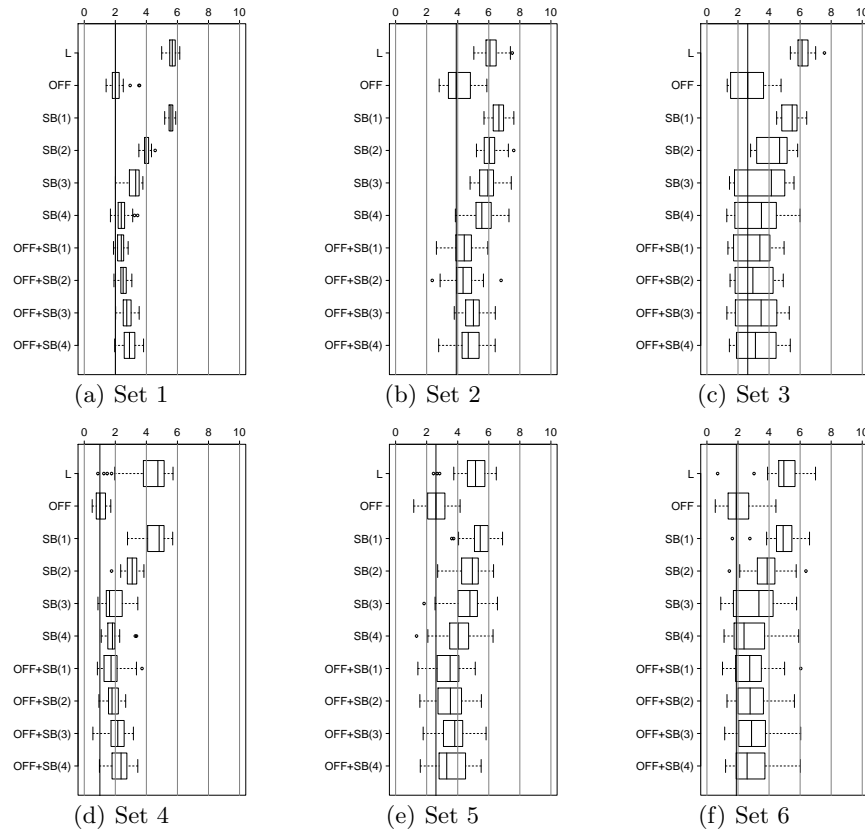


Fig. 39. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 10 seconds. Search-based mechanism. The horizontal axis represents the percentage 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.

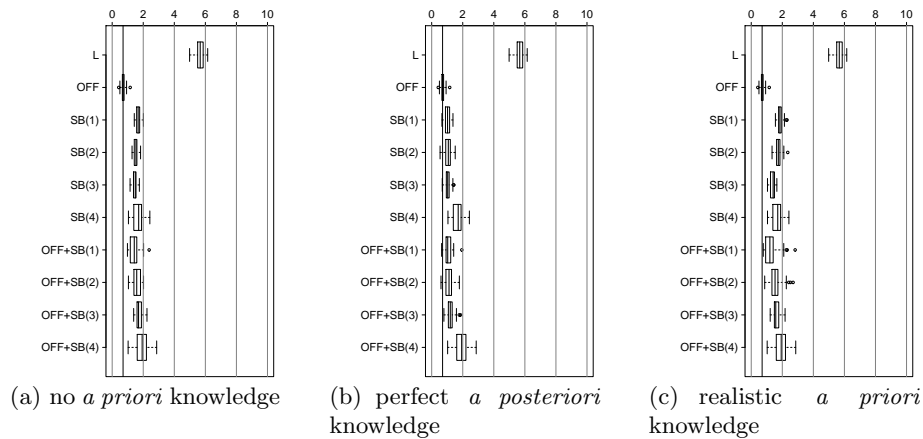


Fig. 40. Results in long runs. Maximum number of objective function evaluations equal to evaluation performed by the *off-line* version in runs of 60 seconds. Search-based mechanism. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one. The vertical line corresponds to the median percentage error made by the *off-line* version.

References

1. Pellegrini, P., Stützle, T., Birattari, M.: Off-line and on-line tuning: a study on *MAX-MIN* Ant System for TSP (2010)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)
3. Birattari, M.: Tuning Metaheuristics: A machine learning perspective. Springer, Berlin, Germany (2009)
4. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In Langdon, W., ed.: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, USA, Morgan Kaufmann Publishers (2002) 11–18
5. 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
6. Anghinolfi, D., Boccalatte, A., Paolucci, M., Vecchiola, C.: Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In Li, X., Kirley, M., Zhang, M., Green, D.G., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y., eds.: SEAL. Volume 5361 of Lecture Notes in Computer Science., Springer (2008) 411–420
7. 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)
8. Johnson, D., McGeoch, L., Rego, C., Glover, F.: 8th dimacs implementation challenge. <http://www.research.att.com/~dsj/chtsp/> (2001)

9. Birattari, M., Dorigo, M.: How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? *Optimization Letters* **1**(3) (2007) 309–311