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

Paola Pellegrini[1], Thomas Stützle[2], and Mauro Birattari[2]

[1] Dipartimento di Matematica Applicata
Università Ca' Foscari Venezia, Venezia, Italia
paolap@pellegrini.it
[2] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{stuetzle,mbiro}@ulb.ac.be

Abstract. Stochastic local search algorithms require finding an appropriate setting of their parameters in order to reach high performance. The parameter tuning approaches that have been proposed in the literature for this task can be classified into two families: *on-line* and *off-line* tuning. In this paper, we compare the results we 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. We observe the performance of on-line parameter tuning for different parameter adaptation schemes and for different numbers of parameters to be tuned. Our results indicate that, under the experimental conditions chosen here, off-line tuned parameter settings are preferable.

1 Introduction

The performance of stochastic local search (SLS) algorithms [15], depends on the appropriate setting of numerical and categorical parameters [7]. Methods that find good parameter settings in an automatic way have recently received strong attention by the research community [4,9,7,10,16,19]. A main contribution of those methods is to alleviate algorithm designers from the tedious and error-prone task of hands-on parameter adaptation.

The available approaches for automated parameter tuning can be classified into either *off-line* or *on-line* approaches. 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 with the selected parameter setting. Off-line approaches are typically black-box and they do not require any modification of the algorithm at hand. Examples of off-line approaches are F-Race [9], Iterated F-Race [3], CALIBRA [1] and ParamILS [16]. The main cost of off-line tuning is due to the 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 [4]. These approaches typically adapt very few key parameters of an algorithm and require substantial insight into algorithm behavior for their development. On-line parameter adaptation has also received a strong interest in the evolutionary computation community [19], where often 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) [24] to the traveling salesman problem (TSP). Our experimental setup is based on the initial conjectures that (i) for homogeneous instance sets off-line tuning should result in excellent performance; (ii) the higher is the number of parameters adapted the worse should get the performance of on-line tuned algorithms; and (iii) for heterogeneous instance sets on-line tuning should have an advantage over off-line.

As an off-line tuning method we use F-Race [5] on the set of candidate configurations we considered. The on-line tuning approaches are given the same set of candidate configurations and we test 5 on-line approaches. Each of the on-line approaches is tested for various numbers of parameters that are to be adapted online. Our results indicate that, in the setting considered, 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 worsens as the number of parameters to be adjusted increases.

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

$\mathcal{MM}$AS [24] is one of the most prominent ACO algorithms. It extends ant system [12] 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 [23]. 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 [24] 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 [11], 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 n indicates how many cities are considered as candidates for the next city.

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 five on-line approaches.

For off-line tuning we apply F-Race [7,9]. 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 configurations 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. Thanks to this progressive elimination, F-Race uses all the available resources for focusing on the most promising configurations. For more details on F-Race we refer to [7].

The first on-line approaches tested follow the self-adaptive approach [13], that is, the determination of the parameter values is integrated into the algorithm's search process. This is done by associating pheromone trails to each possible value of a parameter and by using the ants' construction mechanism to choose which parameter value to adopt. After the solution construction, the pheromone update rule is applied to the trail associated to both parameter values and solution components. Various authors have proposed variants of such a self-adaptive approach [20,22,14,18]. The existing 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. In our study, parameters are treated as interdependent.

Typically, the self-adaptive approaches manage parameters at the ant-level, i.e., each ant selects its own parameter setting [20,22,14]. However, if each ant uses a different parameter setting, the speed-up techniques used in the ACOTSP code (essentially pre-computations of values required in the solution construction) cannot be used, leading to high computation time. Therefore we consider also the case in which parameters are managed at the colony-level, i.e., one parameter setting is fixed for all ants at the beginning of each iteration [18]. In this framework, we analyze three variants of the adaptive algorithm, that differ for the parameter values on which pheromone is deposited after the colony has completed its activity. In all cases, pheromone is deposited on the edges connecting the parameter values used in one specific iteration. In the first case, the parameter settings that receive reinforcement are either the ones of the current iteration or those used for generating the best-so-far solution. In the second case, the parameter settings reinforced are the ones for which the best solution was generated across all previous 25 iterations. In the third case, the parameter settings reinforced are the ones for which the best average solution cost was found across all previous 25 iterations. When the adaptation is made at the colony-level, all six parameters described in Section 2 can be adapted $(q_0, \beta, \rho, m, \alpha, n)$. When it is done at the ant-level, the on-line tuning can be applied only to the parameters involved in the solution construction (q_0, β, α, n).

The second on-line tuning mechanism we examine uses a search-based procedure for selecting the best values of parameters in the run of the algorithm [2]. 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 the center of the neighborhood for the next iteration. Being parameters associated to groups of ants, the speed-up procedures used in the ACOTSP code cannot be fully exploited. With this mechanism, the four parameters involved in solution construction are adapted, that is, α, β, n, q_0.

All the five on-line tuning procedures have been implemented for the experimental analysis. For the sake of brevity we consider in the following only one of the self-adaptive approaches, the one that gives the best results. The whole analysis is available in [21].

4 Experimental Setup

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

- *literature* (L): parameter values are set as suggested in the literature [12]. 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 at colony-level is used, starting from the parameter setting suggested in the literature [12];
- *off-line + self-adaptive on-line* (OFF+SA): the self-adaptive mechanism at colony-level is used, starting from the parameter setting returned by F-Race.

The same analysis has been done with the other adaptation schemes.

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 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 added on top of the ACOTSP software [23].

Table 1. Values that can be chosen for each parameter. The values reported in bold type are the ones suggested in the literature [12]. 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.

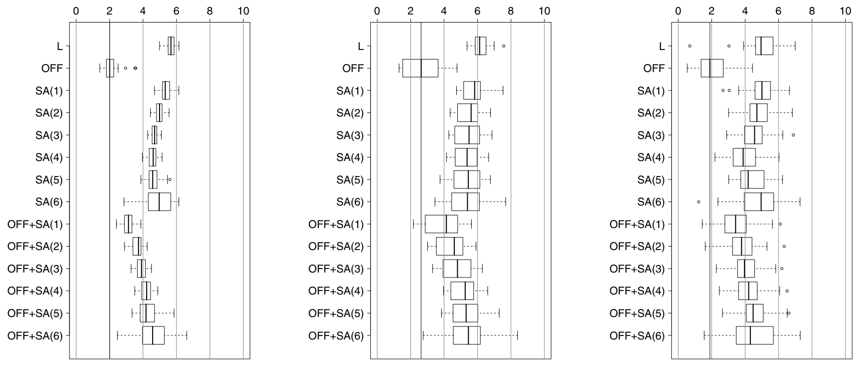
set	number of nodes	spatial distribution	F-Race selection					
			α	β	ρ	q_0	m	n
1	2000	uniform	1	5	0.75	0.5	25	20
2	2000	clustered	2	1	0.25	0.75	25	40
3	2000	uniform and clustered	1	1	0.25	0.9	25	20
4	$U(1000, 2000)$	uniform	1	5	0.75	0.25	50	20
5	$U(1000, 2000)$	clustered	2	2	0.25	0.75	50	40
6	$U(1000,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

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 [17]. 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. In this environment, the ACOTSP code generates about 2 500-3 000 solutions for instances of set 1. The results presented in Section 5 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 [6,8].



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

Fig. 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.

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 convenient in the setting considered. This result is not due to the time overhead implied by adaptation, but due to the nature of the adaptation itself.

5 Experimental Results

The results for all sets of instances described in Section 4 and all the adaptation schemes described in Section 3 are reported in [21]. Due to the limited space available, we focus here on the results of the self-adaptation mechanism at colony-level 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. The on-line tuning approach shown here achieves the best results: the qualitative conclusions that can be drawn are very similar for all the adaptation schemes. We compare the two tuning approaches simulating different levels of knowledge on which are the most relevant parameters to be tuned on-line.

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, where the boxplots summarize the average results in terms of percentage error.

The performance of OFF is the best for all 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; this conjecture is confirmed in Experiment 4 on long run times.

Depending on whether L or OFF settings are used as initial parameter values, different behavior of the on-line parameter adaptation schemes can 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 tuning 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 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 average solution quality reached. Remarkably, this conclusion remains the same for different levels of the heterogeneity of the instance sets.

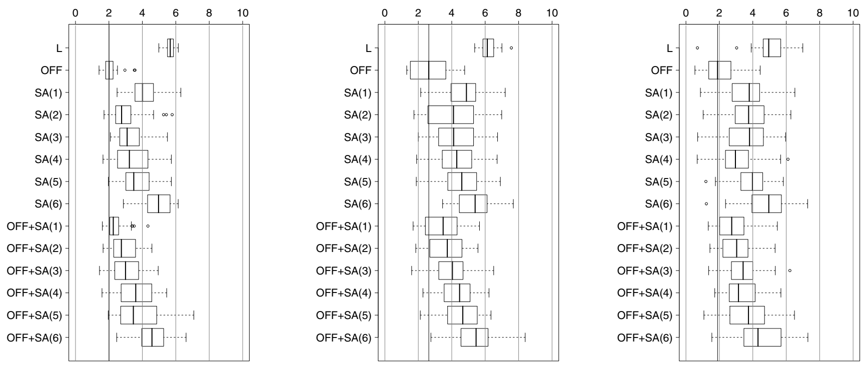
Experiment 2: perfect *a posteriori* knowledge on parameter importance for on-line tuning. Here we simulate the case in which the algorithm designer knows exactly which are the most important parameters to be tuned. This is done by considering the *a posteriori* best configuration for each cardinality 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, as shown below, the off-line version remains preferable. Hence, 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 but OFF+SA(2) for set 2. Hence, even for the most heterogeneous class of instances, set 6, OFF is performing better than the on-line tuned version. L is always significantly worse than all the other versions.

The quality of the final results, as a function of the number of parameters adapted, follows the same trend observed in Figure 1. It also confirms 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.

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 it was already the best in the previous two experiments. For what SA is concerned, the results are



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

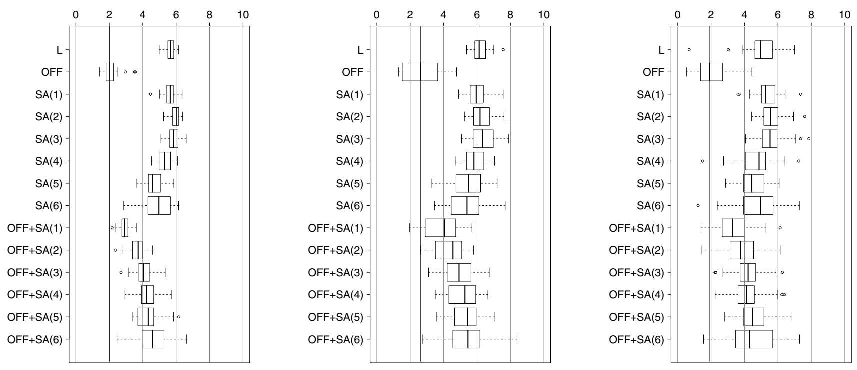
Fig. 2. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one.

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 lead 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 lead close to the best possible results.

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.

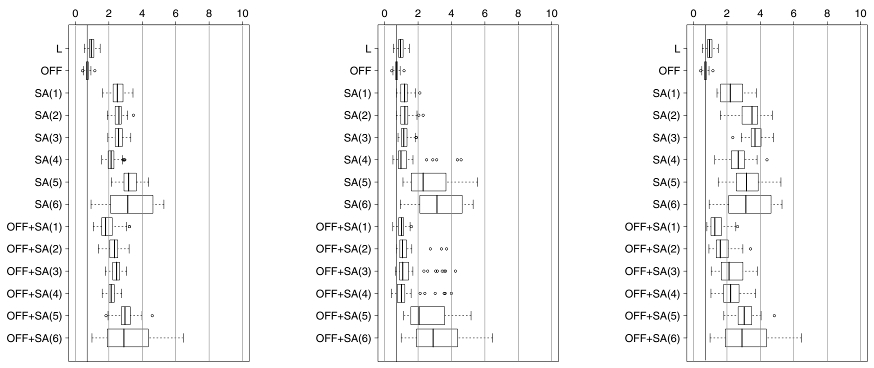
The conclusions that can be drawn are very much in line with those for the shorter computational time: Off-line tuning allows $\mathcal{MM}$AS to achieve the best performance with respect to all the other versions. The differences are statistically significant (checked using the Wilcoxon rank-sum test). A major improvement is experienced by the *literature* version: as we expected, longer runs allow the parameter setting suggested in [12] to achieve good results.

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



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

Fig. 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.



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

Fig. 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.

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(1), SA(4) and OFF+SA(4); the difference is significant in favor 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 strengthens the claim that tuning on-line just one or two parameters, instead of a large number, is the most convenient choice: Not only we can expect the approach to achieve 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 potential of the tuning is actually exploited.

Summary of results. From the results just described we can conclude that:

– off-line tuning performs better than on-line tuning under all the experimental conditions tested;
– it is preferable to apply on-line tuning to few parameters than to many;
– if the initial parameter setting is a well-behaving configuration, our intuition on the most important parameters to adapt allows to exploit the on-line tuning more efficiently than if the setting suggested in the literature is used as starting configuration.

The heterogeneity of the class of TSP 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.

We have proposed an experimental analysis based on one off-line tuning and five on-line tuning procedures. We have considered $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the TSP, and we have solved instances of six sets, differing in the heterogeneity of the instances included. Within this setting we have tested three main conjectures on the quality of the results achievable by tuning parameters off-line *vs.* on-line: two of them have been confirmed by the experimental evidence, while one has actually been contradicted. In particular, (i) as expected, for homogeneous instance sets off-line tuning has resulted in excellent performance; (ii) as expected, the higher the number of parameters adapted the worse the performance of on-line tuned algorithms; and (iii) contrarily to what expected, for heterogeneous instance sets on-line tuning has not had an advantage over off-line: also on these instances off-line tuning has resulted in excellent performance. In this paper we have reported the results achieved by the best one, while the complete analysis is shown in [21]. The conclusions that can be drawn are equivalent regardless the specific approach considered.

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 tackled 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. Thanks to a social experiment, we could observe that researchers' intuition on the most important parameters to tune on-line allows to get better results if an optimized parameter setting is used for starting the adaptation, rather than if the default setting is used.

A further possible direction of future research consists in using an off-line tuning approach for setting the value of the meta-parameters that drive the adaptation in on-line tuning. Examples of such meta-parameters are the number of iterations used in the self-adaptation scheme with multiple-colony comparison, or the number of neighbor-configurations in the search-based adaptation.

Acknowledgments. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Mauro Birattari and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are Research Associates. The authors thank the colleagues that answered the survey described in the paper.

References

1. Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54(1), 99–114 (2006)
2. 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., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 411–420. Springer, Heidelberg (2008)
3. 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. LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg (2007)
4. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Operations Research/Computer Science Interfaces, vol. 45. Springer, Berlin (2008)
5. Birattari, M.: Race. R package (2003), `http://cran.r-project.org`
6. Birattari, M.: On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Tech. Rep. TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2004)
7. Birattari, M.: Tuning Metaheuristics: A Machine Learning Perspective. Springer, Berlin (2009)
8. 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), 309–311 (2007)
9. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W., et al. (eds.) GECCO 2002, pp. 11–18. Morgan Kaufmann Publishers, San Francisco (2002)
10. Coy, S., Golden, B., Runger, G., Wasil, E.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics 7(1), 77–97 (2001)

11. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
12. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
13. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter control in evolutionary algorithms. In: [19], pp. 19–46
14. Förster, M., Bickel, B., Hardung, B., Kókai, G.: Self-adaptive ant colony optimisation applied to function allocation in vehicle networks. In: GECCO 2007, pp. 1991–1998. ACM Press, New York (2007)
15. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco (2005)
16. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
17. Johnson, D., McGeoch, L., Rego, C., Glover, F.: 8th DIMACS implementation challenge (2001), `http://www.research.att.com/~dsj/chtsp/`
18. Khichane, M., Albert, P., Solnon, C.: A reactive framework for ant colony optimization. In: Stützle, T. (ed.) LION 3. LNCS, vol. 5851, pp. 119–133. Springer, Heidelberg (2009)
19. Lobo, F., Lima, C.F., Michalewicz, Z.: Parameter Setting in Evolutionary Algorithms. Springer, Berlin (2007)
20. 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), 651–665 (2007)
21. Pellegrini, P., Stützle, T., Birattari, M.: Companion of off-line and on-line tuning: a study on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for TSP (2010) IRIDIA Supplementary page, `http://iridia.ulb.ac.be/supp/IridiaSupp2010-008/`
22. Randall, M.: Near Parameter Free Ant Colony Optimisation. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 374–381. Springer, Heidelberg (2004)
23. Stützle, T.: ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem (2002), `http://www.aco-metaheuristic.org/aco-code`
24. Stützle, T., Hoos, H.H.: MAX–MIN ant system. Future Generation Computer Systems 16(8), 889–914 (2000)