Swarm Intell (2012) 6:23–48
DOI 10.1007/s11721-011-0061-0

# A critical analysis of parameter adaptation in ant colony optimization

**Paola Pellegrini · Thomas Stützle · Mauro Birattari**

Received: 11 November 2010 / Accepted: 13 September 2011 / Published online: 21 October 2011


**Abstract** Applying parameter adaptation means operating on parameters of an algorithm while it is tackling an instance. For ant colony optimization, several parameter adaptation methods have been proposed. In the literature, these methods have been shown to improve the quality of the results achieved in some particular contexts. In particular, they proved to be successful when applied to novel ant colony optimization algorithms for tackling problems that are not a classical testbed for optimization algorithms. In this paper, we show that the adaptation methods proposed so far do not improve, and often even worsen the performance when applied to high performing ant colony optimization algorithms for some classical combinatorial optimization problems.

**Keywords** Ant colony optimization · Parameter adaptation · Traveling salesman problem · Quadratic assignment problem

## 1 Introduction

Many swarm intelligence techniques, such as ant colony optimization (ACO) (Dorigo and Stützle 2004) and particle swarm optimization (Clerc 2006), have a number of numerical and categorical parameters that can have a crucial impact on performance. The appropriate setting of these parameters depends on the problem to be tackled and, for a given problem, on the problem instances.

Parameter adaptation methods (Angeline 1995), also known as parameter control methods (Eiben et al. 2007), operate on the parameter setting during the execution of the algorithm. These methods constantly modify the parameter setting on an instance-per-instance

P. Pellegrini (✉) · T. Stützle · M. Birattari
IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
e-mail: paola.pellegrini@ulb.ac.be

T. Stützle
e-mail: stuetzle@ulb.ac.be

M. Birattari
e-mail: mbiro@ulb.ac.be

basis. The use of adaptation methods is motivated by the fact that an instance-optimal parameter setting always obtains better results than any other setting. Adapting the parameter setting on an instance-per-instance basis is an intriguing possibility that, in principle, could allow using the instance-optimal setting in each case. Aiming at using an instance-optimal setting, several authors have proposed adaptation methods.

For ACO, several parameter adaptation methods have been proposed. Some of these methods are self-adaptive, following the classification proposed by Eiben et al. (2007): the space of the parameter settings is coupled with the solution space of the instance to be tackled, and the algorithm operates in the joint space so obtained. Four main applications of self-adaptive ant colony optimization have been published so far. They are all based on original variants of existing algorithms. Randall (2004) tackled 12 instances of both the traveling salesman problem and the quadratic assignment problem through a self-adaptive variant of ant colony system. Martens et al. (2007) proposed a self-adaptive variant of AntMiner for tackling classification problems. Förster et al. (2007) dealt with function allocation in vehicle networks using a self-adaptive variant of a multi-colony ant algorithm. Finally, Khichane et al. (2009) introduced a self-adaptive variant of an ant-solver for tackling constraint satisfaction problems. Another parameter adaptation method that has been proposed in the literature is based on local search. Anghinolfi et al. (2008) applied it in an adaptive variant of ant colony system for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. For a review of these and other adaptation methods that have been proposed for ACO, we refer the reader to Stützle et al. (2011). In the literature, no direct comparison among different adaptation methods has been proposed. Thus, it is not possible to formally identify any method as the state-of-the-art one. Nonetheless, the self-adaptive ones can be considered, to some extent, the state-of-the-art in the field of parameter adaptation in ACO: the results reported in the papers in which they are proposed indicate that they positively contribute to performance.

In this paper, we implement into ACO algorithms four self-adaptation methods that are based on the work of Martens et al. (2007), Randall (2004), Förster et al. (2007) and Khichane et al. (2009), and one local-search-based adaptation method that is based on the work of Anghinolfi et al. (2008). We consider an experimental setup that has been adopted in several analyses reported in the literature. In this way, we capture the contribution of adaptation methods independently of the particular algorithms and problems for which they were proposed. We use $\mathcal{MAX}$–$\mathcal{MIN}$ ant system ($\mathcal{MM}$AS) (Stützle and Hoos 2000), and we solve both the traveling salesman problem (TSP) (Lawler et al. 1985) and the quadratic assignment problem (QAP) (Lawler 1963). For each problem, we run the algorithm under multiple experimental setups. We vary the experimental setup with respect to four factors:

(i) the number of parameters adapted,
(ii) the quality of the results achieved by the algorithm,[1]
(iii) the heterogeneity of the instances to be tackled,
(iv) the runtime.

Intuitively, these factors affect the impact of adaptation methods on the performance of the algorithm. First, a large number of parameters adapted corresponds to a large space of the parameter settings, the effective exploration of which may be quite difficult. Second, an algorithm achieving high quality results has already been optimized: little margin remains for further improvements. Third, very heterogeneous instances are much better tackled with

[1] Without parameter adaptation.

a parameter setting selected on an instance-per-instance basis than with a common parameter setting. Fourth, a short runtime may be too stringent for having both the adaptation methods selecting the appropriate setting, and the algorithm finding a good quality solution to the problem instance. Thus, we conjecture that:

(i) the more parameters adapted,
(ii) the higher the quality of the results achieved by the algorithm,
(iii) the lower the heterogeneity of the set of instances to be tackled, and
(iv) the shorter the runtime,

the smaller the improvement on the performance of the algorithm that adaptation methods can produce. For either refuting or corroborating one or more of these conjectures, we consider for both, the TSP and the QAP, (i) different numbers of parameters adapted, (ii) multiple settings for statically assigned parameters,[2] (iii) several sets of instances with different levels of heterogeneity, and (iv) short and long runtimes. By assigning different values to the statically assigned parameters, we capture the impact of the quality of the results achieved by the algorithm on the improvement obtained by adaptation methods. In particular, by setting statically assigned parameters as suggested in the literature, the algorithm may achieve only relatively low quality results. This is due to the fact that the suggested parameter setting is not necessarily the most appropriate one in a context that is different from the one for which it was originally proposed. We have the algorithm achieve high quality solutions by setting statically assigned parameters as resulting from the application of an off-line tuning procedure (Birattari 2009). In this context, off-line tuning is a means that we use to create algorithms of different levels of performance.

The results we obtain in this extensive experimental analysis show that, in the strong majority of the cases, adaptation methods not only fail to improve the performance $\mathcal{MM}$AS, but they even worsen it. This happens also in cases in which we would have expected to obtain good performance with an adaptation method, such as, for example, with heterogeneous instances, long runtime and few parameters to be adapted. The only exception is when the setting of statically assigned parameters leads the algorithm to achieve relatively low quality results under specific experimental conditions. This case is anyway of little interest: virtually any sensible refinement of the algorithm is likely to increase the quality of the results.

The rest of the paper is organized as follows. In Sect. 2, we describe the main characteristics of $\mathcal{MM}$AS and of the specific algorithms that we implement for solving the two problems tackled. In Sect. 3, we present the adaptation methods we tested. In Sects. 4 and 5, we report the experimental setup and the results, respectively. Finally, in Sect. 6, we draw some conclusions.

## 2 $\mathcal{MAX}$–$\mathcal{MIN}$ ant system

In ACO, a colony of artificial ants explores the search space iteratively. Ants are independent agents that construct solutions incrementally, component by component. Ants communicate indirectly with each other through pheromone trails, biasing the search toward regions of the search space containing high quality solutions.

When applying an ACO algorithm, an optimization problem is typically mapped to a construction graph $G = (V, E)$, with $V$ being the set of nodes and $E$ being the set of edges

[2]That is, those that are not handled by the adaptation mechanism.

connecting the nodes. Solution components may be represented either by nodes or by edges of this graph. In the following, we describe the main procedures that characterize $\mathcal{MM}$AS, considering solution components associated to edges, and supposing that a minimization problem is to be solved.

A pheromone trail $\tau_{ij}$ is associated to each edge $(i, j) \in E$; it represents the cumulated knowledge of the colony on the convenience of choosing solution component $(i, j)$. At the end of each iteration, in which $m$ ants construct one solution each, this cumulated knowledge is enriched by applying a pheromone update rule. Some pheromone evaporates from each edge, and some is deposited on the edges belonging to the best solution, considering either the last iteration (iteration-best solution), the whole run (best-so-far solution), or the best since a re-initialization of the pheromone trails (restart-best solution) (Stützle and Hoos 2000). This pheromone update rule is implemented as

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \begin{cases} F(s_{best}) & \text{if edge } (i, j) \text{ is part of the best solution } s_{best}; \\ 0 & \text{otherwise;} \end{cases} \quad (1)$$

where $F(s_{best})$ is the inverse of the cost of the best solution (recall that we are dealing with minimization problems here), and $\rho$ is a parameter of the algorithm called the evaporation rate ($0 < \rho < 1$). In addition, the pheromone strength is constantly maintained in the interval $[\tau_{min}, \tau_{max}]$. These bounds are functions of the state of the search. When pheromone trails are excessively concentrated, they are re-initialized uniformly on all edges to favor exploration (Stützle and Hoos 2000).

Exploiting the common knowledge represented by pheromone trails, ants construct solutions independently of each other, by selecting at each construction step the edge to traverse. In $\mathcal{MM}$AS, this selection is done according to the random-proportional rule: ant $k$, being in node $i$, moves to node $j \in N^k$ with probability

$$p_{ij} = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{h \in N_i^k} [\tau_{ih}]^\alpha \cdot [\eta_{ih}]^\beta}, \quad (2)$$

where $\alpha$ and $\beta$ are parameters of the algorithm, $\eta_{ij}$ is a heuristic measure representing the desirability of using edge $(i, j)$ from a greedy point of view, and $N_i^k$ is the set of nodes to which the ant can move to, when being in node $i$.

Typically, the $m$ solutions generated by ants at each iteration are used as starting points for local search runs, one for each initial solution. The solutions returned are then used in the pheromone update as if they had been built by the ants.

The procedures just described may be used for designing algorithms for virtually any optimization problem. In Sects. 2.1 and 2.2, we describe the specific implementation we use for the TSP and the QAP, respectively.

### 2.1 $\mathcal{MM}$AS for the TSP

The TSP consists in finding a minimum-cost tour for visiting a set of cities exactly once, starting and ending at the same location. The cost of going from one city to another is fixed. A TSP instance is mapped to a graph by associating a node to each city, and an edge with a predefined cost to the connection between each pair of cities. The objective of the problem is to construct a minimum cost Hamiltonian tour (Lawler et al. 1985).

For solving the TSP, we use the $\mathcal{MM}$AS algorithm included in ACOTSP (Stützle 2002). The heuristic measure $\eta_{ij}$ is the inverse of the cost of traversing the edge $(i, j)$. The set $N_i^k$ of nodes to which ant $k$ can move includes all nodes that belong to the candidate list associated to node $i$ and that have not been visited yet. The candidate list contains the *nn*

nearest neighbors of node $i$, where $nn$ is a parameter of the algorithm. The remaining nodes are included in $N_i^k$ only if all nodes in the candidate list have already been visited.

The performance of $\mathcal{MM}$AS for the TSP can possibly be improved by using the pseudo-random proportional rule that is used by another ACO algorithm, namely the ant colony system (ACS) (Dorigo and Gambardella 1997). Thus, we exploit this rule here: with a probability $q_0$, the next node $j$ to be inserted in an ant's partial solution is the most attractive node according to pheromone and heuristic measure:

$$j = \arg\max_{h \in N_i^k} \left\{ \tau_{ih}^{\alpha} \eta_{ih}^{\beta} \right\}. \tag{3}$$

With probability $1 - q_0$, the ant uses the traditional random proportional rule of (2). Thus, $q_0$ is a further parameter of the algorithm, with $0 \leq q_0 < 1$. A 2-opt local search is applied to all solutions constructed by ants.

### 2.2 $\mathcal{MM}$AS for the QAP

The QAP consists in finding a minimum-cost assignment of a set of $n$ facilities to a set of $n$ locations (Lawler 1963). A flow $f_{ij}$ is associated to each pair of facilities $i, j = 1, \ldots, n$, and a distance $d_{hk}$ is given for each pair of locations $h, k = 1, \ldots, n$. A solution corresponds to an assignment of each facility to a location. It can be represented as a permutation $\pi$: the value in position $i$ of the permutation, $\pi(i)$, corresponds to the facility that is assigned to location $i$. The cost of a solution is equal to the sum over all pairs of locations of the product of the distance between them, and the flow between their assigned facilities:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{\pi(i)\pi(j)} d_{ij}. \tag{4}$$

A QAP instance is mapped to a graph by associating nodes to facilities and locations: using edge $(i, j)$ in the solution construction corresponds to the assignment of facility $i$ to location $j$.

The implementation of $\mathcal{MM}$AS for the QAP used in this paper is described by Stützle and Hoos (2000). Pheromone trail is associated to edges, and no heuristic measure is used; thus, parameter $\beta$ is not considered here.

The algorithm follows the general framework of $\mathcal{MM}$AS described at the beginning of the current section. Depending on the characteristics of the set of instances tackled, the best performing local search may vary (Stützle and Hoos 2000). Thus, local search is a further parameter of the algorithm. In the following, it will be referred to as $l$.

## 3 The adaptation methods considered

We study the results achieved by $\mathcal{MM}$AS when applying state-of-the-art parameter adaptation methods proposed for ACO. In the following, let $P$ be the set of all parameters that may be adapted, and $A$ be the set of parameters to be adapted, $A \subseteq P$. Adaptation methods operate in a parameter setting space determined a priori. Each possible setting of a parameter $p$ is given by a set $S_p$. Here we only deal with numerical parameters. We assume that the possible values are obtained by a discretization of the feasible intervals in the case of real-valued parameters, or by a selection of representative values in the case of integer parameters; the possible values are then listed in $S_p$ in increasing order. The space of the parameter setting to be explored by adaptation methods is then given by all possible combinations of elements

of set $S_p$, for all $p \in A$. One can extend this concept to the case of categorical parameters in a straightforward way. A possible strategy is defining an arbitrary ordering of the elements of $S_p$.

### 3.1 Self-adaptive methods

The first four methods we consider are self-adaptive. In self-adaptive methods, the mechanism that is used to adapt parameters is the same as that underlying the optimization algorithm (Eiben et al. 2007). This can be realized by associating to each parameter $p \in A$ a set of nodes $V'_p$ in the construction graph, corresponding to all possible settings in $S_p$. This results in an additional set of nodes,

$$V' = \bigcup_{p \in A} V'_p. \tag{5}$$

After defining an ordering on set $A$, each node associated to a parameter is connected through an edge to each node associated to the following parameter in $A$, resulting in edge set $E'$. Each node associated to the last parameter is connected to all nodes in the original set $V$. Pheromone trails are associated to all edges in $E \cup E'$ and they are all managed with the update rule reported in (1). The parameter settings to be used are selected by the search mechanism that is used in the ACO algorithm, by choosing one node in each set $V'_p$.

Multiple attempts have been made for exploiting efficiently self-adaptive methods in ACO algorithms. They differ in two main points:

Dependent/Independent parameters. Parameters are considered either independently or interdependently from one another. If parameters are considered independent of one another, pheromone trails are associated to the nodes in $V'$. In this way, a setting for one parameter is chosen independently of the other settings. If parameters are considered dependent on one another, pheromone is on the edges. In this way, interactions among parameters may be taken into account.

Colony-level/Ant-level parameters. Parameters are managed either at the colony or at the ant level. When they are managed at the colony level, at each iteration the same setting is used for the whole colony. When parameters are managed at the ant level, at each iteration the ants may use a different setting.

In the four self-adaptive methods used in this paper, parameters are considered as interdependent. Following the literature (Martens et al. 2007; Khichane et al. 2009), the ordering defined on set $A$, that is, the order in which parameter settings are selected, is fixed a priori and maintained constant throughout the runs. The methods differ in the level at which they manage parameters and in the selection of the setting on which pheromone is reinforced:

- SAc: colony-level parameters; it reinforces pheromone on the parameter setting with which the best-so-far, the iteration-best, or the restart-best solution was found according to the schedule defined by Stützle and Hoos (2000);
- SAcb: colony-level parameters; it reinforces pheromone on the parameter setting with which the algorithm found the best solution in the last 25 iterations;
- SAcm: colony-level parameters; it reinforces pheromone on the parameter setting with which the algorithm found the set of solutions with the lowest mean cost in the last 25 iterations;
- SAa: ant-level parameters; similarly to SAc, it reinforces pheromone on the parameter setting with which the best-so-far, the iteration-best, or the restart-best solution was found according to the schedule defined by Stützle and Hoos (2000).

Both $\rho$ and $m$ are colony-wise parameters. Thus, they cannot be adapted by the methods that use multiple settings in each iteration of $\mathcal{MM}$AS, that is, the methods where parameters are handled at the ant level. Hence SAc, SAcb and SAcm may adapt all the parameters described in Sect. 2: $P = (\alpha, \beta, \rho, m, q_0, n)$ for the TSP, and $P = (\alpha, \rho, m)$ for the QAP. SAa, instead, adapts fewer parameters: $P = (\alpha, \beta, q_0, n)$ for the TSP, and $P = (\alpha)$ for the QAP. The order in which these parameters are listed reflects the ordering we define for $A$ in our experiments.

### 3.2 Local-search-based method

The fifth adaptation method, LS, is based on the exploration of the space of the parameter setting using a naive local search approach (Anghinolfi et al. 2008). LS evaluates at each step one reference setting and the neighbors of it. The neighborhood includes all settings that differ in exactly one parameter $p$ from the reference one. If this parameter in the reference setting is the element in position $i$ in the set $S_p$ of possible ones, the neighbors are the ones taking the elements in position $(i-1)$ and $(i+1)$. If either the elements in position $(i-1)$ or $(i+1)$ do not exist in $S_p$, which means that the reference setting includes either the first or the last element in $S_p$, one neighbor is not valid, and LS doubles the reference setting. The reference setting is replaced by one of the neighbors if it is not the best performing.

LS evaluates the settings by splitting the ant colony in groups, and by having each group of ants build solutions using a different setting. The number of groups is equal to the size of the neighborhood. The performance of each setting is defined as the value of the best solution found by the corresponding group of ants. LS evaluates each candidate setting for 10 iterations of $\mathcal{MM}$AS before deciding whether to change the reference setting or not, as done by Anghinolfi et al. (2008). Ties are resolved randomly.

Analogously to SAa, LS adapts only parameters that are appropriate for the ant level, that is, $P = (\alpha, \beta, q_0, n)$ for the TSP and $P = (\alpha)$ for the QAP.

## 4 Experimental setup

In the experimental analysis, we empirically show that the state-of-the-art adaptation methods proposed for ant colony optimization do not achieve good performance under several experimental conditions.

We apply five adaptation methods to $\mathcal{MM}$AS for both the TSP and the QAP. We analyze the performance of the methods as a function of the cardinality of the set of parameters adapted, of the setting of statically assigned parameters, of the heterogeneity of the set of instances tackled, and of the runtime.

The results reported for each adaptation method are obtained with one single run of $\mathcal{MM}$AS on 100 instances for each set and for each runtime (Birattari 2004). We perform the experiments on Xeon E5410 quad core 2.33 GHz processors with $2 \times 6$ MB L2-Cache and 8 GB RAM, running under the Linux Rocks Cluster Distribution, after compiling the code with `gcc`, version 3.4.

### 4.1 Number of parameters adapted

We evaluate the adaptation methods in correspondence to different numbers of parameters adapted, that is, to different cardinalities of the sets $A$. For each cardinality, we test all sets

**Table 1** Possible parameter settings for the TSP and the QAP. The settings reported in bold type are the ones suggested in the literature (Stützle and Hoos 2000; Dorigo and Stützle 2004)

| TSP | |
|---|---|
| parameter | settings ($S$) |
| $q_0$ | **0.0**, 0.25, 0.5, 0.75, 0.9 |
| $\beta$ | 1, **2**, 3, 5, 10 |
| $\rho$ | 0.1, **0.2**, 0.3, 0.5, 0.7 |
| $m$ | 5, 10, **25**, 50, 100 |
| $\alpha$ | 0.5, **1**, 1.5, 2, 3 |
| $nn$ | 10, **20**, 40, 60, 80 |
| QAP | |
| parameter | settings ($S$) |
| $m$ | 1, 2, **5**, 8, 16 |
| $\rho$ | **0.2**, 0.4, 0.6, 0.8 |
| $\alpha$ | 0.5, **1**, 1.5, 2, 3 |

that can be extracted from $P$. The total number of combinations, and thus the total number of sets $A$ for each cardinality $|A|$ is:

$$\binom{|P|}{|A|} = \frac{|P|!}{|A|!(|P| - |A|)!}. \tag{6}$$

In the experiments, we consider the same space of the parameter settings for all adaptation methods to avoid any possible bias. In Table 1, we report the set $S$ of possible settings of each parameter. The space of the parameter settings includes all combinations of the different settings; it amounts to 15,625 candidates for the TSP, and 100 for the QAP. The settings suggested in the literature (Stützle and Hoos 2000; Dorigo and Stützle 2004) are reported in boldface.

For the QAP, the literature suggests to use 2-opt with best improvement for structured instances, and tabu search relying on 2-opt for unstructured instances (Stützle and Hoos 2000): it is not possible to identify a single local search that dominates the other. Thus, we initially considered different possible settings of parameter $l$: first improvement using don't look bits (0), first improvement without using don't look bits (1), best improvement (2), tabu search runs of length $2n$ (3), and tabu search runs of length $6n$ (4). Preliminary results show that the methods implemented are penalized by the presence of the type of local search in the set of parameters to be adapted. In fact, both the computation time and the solution quality differ for runs of different local search procedures. So, there is a trade-off between computation time and solution quality. Adjusting adaptation methods to take this trade-off into account is beyond the scope of this research work.

### 4.2 Setting of statically assigned parameters

The setting of statically assigned parameters may have a strong impact on the quality of the results achieved by the algorithms. We consider multiple settings of statically assigned parameters. We apply the off-line tuning method named F-Race (Birattari 2009; Birattari et al. 2002) for selecting the appropriate setting of all parameters. In the experiments, we use the so-obtained setting for all parameters that are not handled through an adaptation method, that is, that are not included in set $A$. Off-line tuning selects the appropriate parameter setting

on an instance class basis: an instance class is formally defined as a probability measure over the space of the instances of the optimization problem at hand (Birattari 2009). The appropriate setting is the one that achieves the best expected performance on the instances of a class. For testing different settings of statically assigned parameters, we vary the tuning effort required for selecting them. The tuning effort is represented by the maximum number of experiments performed in the off-line tuning phase, where, in our context, an experiment is one run of $\mathcal{MM}$AS.

For both, the TSP and the QAP, we say that a null effort has been devoted to tuning when the maximum number of experiments equals zero, and the parameter setting is the one suggested in the literature, as reported in Table 1. On the other extreme, we say that a high effort has been devoted to tuning when the maximum number of experiments equals ten (15 for the QAP) (Birattari 2009; Birattari et al. 2002) multiplied by the number of combinations obtainable by considering all the settings reported in Table 1 (times the five possible settings for the local search for the QAP reported in Sect. 4.1). The maximum number of experiments is 156,250 for the TSP, and 7,500 for the QAP. For the TSP, we perform three off-line tunings with intermediate levels of effort by proportionally decreasing the order of magnitude of the maximum number of experiments: 15,625; 1,562; and 156. In these cases, off-line tuning operates on a randomly sampled subspace of the parameter settings.[3] For the QAP, we do not consider these intermediate levels of tuning effort. As we will see in Sect. 5.2, these intermediate levels are not necessary for supporting our conclusions.

We perform a separate off-line tuning for each set of instances, and for each runtime. For each off-line tuning, we use 1000 instances. These instances do not include those used in the evaluations of the tuning methods (Birattari et al. 2006). F-Race terminates when one among three stopping criteria is met: (i) all instances are used, (ii) a fixed number of experiments are executed, (iii) only one parameter setting survives. In our experiments, F-Race always terminates due to either the second or the third stopping criterion.

### 4.3 Instances and runtimes

We assess the performance achieved when applying parameter adaptation on multiple sets of instances for each problem. These sets are characterized by different heterogeneities of the instances included. Intuitively, when the instances to be tackled are very heterogeneous, adapting parameter settings on an instance-per-instance basis should strongly improve the performance of the algorithm. We perform both short and long runs, conjecturing that long runs allow adaptation methods to properly handle parameter settings and the algorithm to solve the instance to be tackled.

*Traveling salesman problem* To examine the impact of the heterogeneity of instance sets on adaptation methods, for the TSP we define six sets of instances with different numbers of cities and different spatial distributions of the cities. All instances are generated through portgen, the instance generator used in the 8th DIMACS Challenge on the TSP (Johnson et al. 2001). The characteristics of each set are described in Table 2. On the right-hand side of this table, we report the setting selected through off-line tuning for short and long runs. Hereafter, we will refer to a set of TSP instances as *TSP* followed by a parenthesis indicating the number of cities included and their spatial distribution. When either the number

[3]To randomly sample the space of the parameter settings, we use an automatic procedure implemented for Iterated F-Race. The size of the sampled subspace is a function of the tuning effort, as described by López-Ibáñez et al. (2011).

**Table 2** Sets of instances considered for the TSP. $U(a, b)$ indicates that a number was randomly drawn between $a$ and $b$ for each instance, according to a uniform probability distribution. The right-hand side of the table reports the setting selected through off-line tuning for short and long runs, and for each tuning effort

| set | number of nodes | spatial distribution | Tuning selection for short runtime | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | effort | $\alpha$ | $\beta$ | $\rho$ | $q_0$ | $m$ | $nn$ |
| $TSP(2000, u)$ | 2000 | uniform | 156,250 | 1 | 5 | 0.75 | 0.5 | 25 | 20 |
| | | | 15,625 | 1 | 5 | 0.75 | 0.5 | 50 | 20 |
| | | | 1,562 | 1 | 2 | 0.75 | 0.25 | 50 | 10 |
| | | | 156 | 1.5 | 1 | 0.75 | 0.25 | 50 | 10 |
| $TSP(2000, c)$ | 2000 | clustered | 156,250 | 2 | 1 | 0.25 | 0.75 | 25 | 40 |
| | | | 15,625 | 2 | 1 | 0.25 | 0.75 | 10 | 40 |
| | | | 1,562 | 3 | 3 | 0.25 | 0.75 | 25 | 60 |
| | | | 156 | 2 | 1 | 0.25 | 0.25 | 50 | 40 |
| $TSP(2000, x)$ | 2000 | uniform & | 156,250 | 1 | 1 | 0.25 | 0.9 | 25 | 20 |
| | | clustered | 15,625 | 2 | 1 | 0.25 | 0.75 | 100 | 20 |
| | | | 1,562 | 3 | 1 | 0.25 | 0.25 | 50 | 20 |
| | | | 156 | 3 | 3 | 0.5 | 0.25 | 25 | 10 |
| $TSP(x, u)$ | $U(1000, 2000)$ | uniform | 156,250 | 1 | 5 | 0.75 | 0.25 | 50 | 20 |
| | | | 15,625 | 1 | 3 | 0.75 | 0.0 | 100 | 10 |
| | | | 1,562 | 1 | 2 | 0.75 | 0.25 | 10 | 10 |
| | | | 156 | 2 | 1 | 0.25 | 0.75 | 100 | 20 |
| $TSP(x, c)$ | $U(1000, 2000)$ | clustered | 156,250 | 2 | 2 | 0.25 | 0.75 | 50 | 40 |
| | | | 15,625 | 3 | 5 | 0.5 | 0.5 | 50 | 40 |
| | | | 1,562 | 1.5 | 1 | 0.25 | 0.9 | 25 | 20 |
| | | | 156 | 1 | 5 | 0.5 | 0.5 | 25 | 20 |
| $TSP(x, x)$ | $U(1000, 2000)$ | uniform & | 156,250 | 1 | 1 | 0.25 | 0.9 | 50 | 20 |
| | | clustered | 15,625 | 1.5 | 3 | 0.25 | 0.75 | 50 | 20 |
| | | | 1,562 | 2 | 3 | 0.25 | 0.75 | 100 | 40 |
| | | | 156 | 1.5 | 3 | 0.75 | 0.9 | 100 | 40 |

| set | number of nodes | spatial distribution | Tuning selection for long runtime | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | effort | $\alpha$ | $\beta$ | $\rho$ | $q_0$ | $m$ | $nn$ |
| $TSP(2000, u)$ | 2000 | uniform | 156,250 | 1 | 3 | 0.5 | 0.5 | 100 | 20 |
| | | | 15,625 | 1 | 2 | 0.5 | 0.5 | 100 | 20 |
| | | | 1,562 | 0.5 | 3 | 0.75 | 0.9 | 50 | 20 |
| | | | 156 | 1 | 2 | 0.75 | 0.25 | 100 | 60 |

of cities or their spatial distribution are not the same in all instances, we report an $x$ in the corresponding position. For example, if a set includes instances with 2000 cities that can be either uniformly distributed in the space or grouped in clusters, we use the acronym $TSP(2000, x)$. The different sets have different levels of heterogeneity:

– We call a set homogeneous if all instances have the same number of cities and the same spatial distribution, as in $TSP(2000, u)$ and $TSP(2000, c)$;

**Table 3** Sets of instances considered for the QAP. The right-hand side of the table reports the setting selected through off-line tuning for short and long runs

| set | size | type | Tuning selection | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | effort | | $\alpha$ | $\rho$ | $l$ | $m$ |
| $QAP(80, RR)$ | 80 | unstructured | 7,500 | short | 1 | 0.6 | 3 | 2 |
| | | | 7,500 | long | 1 | 0.4 | 3 | 1 |
| $QAP(80, ES)$ | 80 | structured | 7,500 | short | 1.5 | 0.4 | 0 | 5 |
| | | | 7,500 | long | 1.5 | 0.2 | 0 | 5 |
| $QAP(80, x)$ | 80 | unstructured or | 7,500 | short | 1 | 0.4 | 3 | 2 |
| | | structured | 7,500 | long | 1 | 0.4 | 3 | 2 |
| $QAP(x, RR)$ | $\in \{60, 80, 100\}$ | unstructured | 7,500 | short | 1 | 0.6 | 3 | 2 |
| | | | 7,500 | long | 1 | 0.8 | 2 | 3 |
| $QAP(x, ES)$ | $\in \{60, 80, 100\}$ | structured | 7,500 | short | 1.5 | 0.4 | 0 | 2 |
| | | | 7,500 | long | 1.5 | 0.4 | 0 | 5 |
| $QAP(x, x)$ | $\in \{60, 80, 100\}$ | unstructured or | 7,500 | short | 1 | 0.4 | 2 | 5 |
| | | structured | 7,500 | long | 1 | 0.4 | 2 | 5 |

– We call a set heterogeneous if neither the number of cities nor their spatial distribution is the same in all instances, as in set $TSP(x, x)$;
– At an intermediate level between those two extremes, we define instance sets where either the number of cities or their spatial distribution is not the same in all instances, as in sets $TSP(2000, x)$, $TSP(x, u)$ and $TSP(x, c)$. For convenience we refer to these sets as semi-heterogeneous.

The runtime is 10 and 60 CPU seconds for short and long runs, respectively. In short runs, the literature version completes between 100 and 120 iterations on instances of set $TSP(2000, u)$. Long runs last 60 CPU seconds and are performed only on instances of set $TSP(2000, u)$. In long runs, the literature version completes between 560 and 600 iterations. We limit the experiments in this sense due to the extremely long computation time that would have been required for replicating the analysis on all the sets.

*Quadratic assignment problem* For the QAP, we consider six sets of instances. We generate instances of three sizes: 60, 80 and 100. Moreover, we generate both unstructured (RR) and structured (ES) instances. The former are among the hardest QAP instances to solve exactly, but they do not have practical relevance; the latter are instances that show a structure (in particular, of the flow matrix) similar to those occurring in real-world QAP instances (Taillard 1995). In unstructured instances, the entries of both distance and flow matrices are random numbers uniformly distributed in the interval $[0, 99]$ (Taillard 1991). For the structured instances, we follow Hussin and Stützle (2010). In structured instances, the entries of the distance matrix are the Euclidean distances of points positioned in a square $100 \times 100$ according to a uniform distribution. The entries are rounded to the nearest integer. For what concerns the flow matrix, first a set of points are randomly located in a square of size $100 \times 100$. For each pair of points, if the distance is longer than a predefined threshold $t$, then the flow is set to zero; otherwise it is equal to a value $x$ resulting from the following procedure: first, consider a random number $x_1 \in [0, 0.7]$; next, compute $x_2 = -\ln x_1$; finally, set $x = \min\{100x_2^{2.5}, 3000\}$. By setting the parameters of the instance generator in this way, the expected quartiles of the distribution of the values are 4, 14 and 50, respectively. The result of this procedure is a matrix with an asymmetric distribution of values (high frequency

of low values and low frequency of high values), which we consider an interesting testbed for our experiments. We generate instances by setting parameter $t$ equal to 72. The peculiarities of each set are described in Table 3. On the right-hand side of Table 3, we report the setting selected through tuning for short and long runs. Hereafter, we will refer to the sets of QAP instances as *QAP* followed by a parenthesis indicating the size of the instances and the characteristics of the matrices. As in the TSP, when the instances in a set have different characteristics, an $x$ is reported in the corresponding position. For example, the set of instances generated inserting random entries in the distance and flow matrices of size 60, 80 or 100, is indicated as $QAP(x, RR)$. For the QAP, we define a scale of heterogeneity for the sets of instances, analogously to the TSP case:

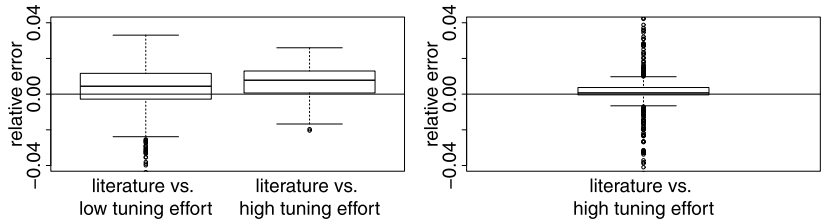
- An instance set is homogeneous if all instances have the same size and characteristics of the distance and the flow matrices, as in sets $QAP(80, RR)$ and $QAP(80, ES)$;
- An instance set is heterogeneous if neither the size nor the characteristics of the distance and the flow matrices are the same in all instances, as in set $QAP(x, x)$;
- Semi-heterogeneous refers to instance sets where either the size or the instance characteristics, either *RR* or *ES*, are not the same in all instances, as in sets $QAP(80, x)$, $QAP(x, ES)$ and $QAP(x, RR)$.

The runtime is 17 and 29 CPU seconds for short and long runs, respectively. To determine these values, we run the literature version with 2-opt local search on instances of size 60, 80 and 100, considering as stopping criterion the number of iterations performed. Short runs are stopped after 200 iterations; long runs after 600. In the experiments, the time available for solving one instance is the average computational time used in these runs.

## 5 Experimental results

In this section, we assess the performance of $\mathcal{MM}$AS for the TSP and for the QAP when applying a parameter adaptation method. We consider all possible numbers of parameters to be adapted, and all possible sets $A$ of each number of parameters. For each problem, we graphically present the results of one adaptation method for two sets of problem instances. We report in the following both the analysis concerning the adaptation of a single parameter, and the results obtained by the adaptation of more than one parameter. When the number of parameters adapted is greater than one, we report the best case results for the adaptation method: for each instance set, runtime, and tuning effort, we report the results of the set $A$ for which the algorithm achieved the best average result. These results are those that one could obtain if one had an oracle that perfectly predicts which parameters should be included into the set $A$, so that the best performance is obtained with respect to any other possible choice of parameters to be adapted. We call this the a posteriori best results. These a posteriori best results do not realistically represent the performance of adaptation methods, since they would require knowing the right composition of $A$ before running the experiments. In this sense, by reporting the a posteriori best case results, we overestimate the quality of the results obtained by adaptation methods. As we will show in this section, this overestimation is not strong enough to improve the performance of the adaptation methods in a qualitatively relevant way. For the TSP, we show here only the results obtained when the setting of statically assigned parameters is either:

- The one suggested in the literature; or
- The one selected through off-line tuning with low tuning effort (maximum number of experiments set to 156).




(a) TSP (b) QAP

**Fig. 1** No parameter adapted. Relative error made when using the literature setting with respect to the one selected through off-line tuning

For the QAP, we show the results obtained when the setting of statically assigned parameters is either:

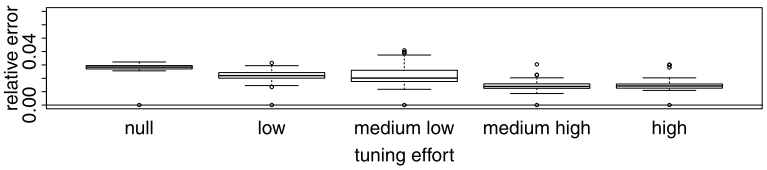
– The one suggested in the literature; or
– The one selected through off-line tuning with high tuning effort (maximum number of experiments set to 7,500).

These results refer to a portion of the experiments described in Sect. 4. This portion of results allows drawing the conclusions of our analysis. The results of the whole study are available in Pellegrini et al. (2010a), which further confirm our conclusions.

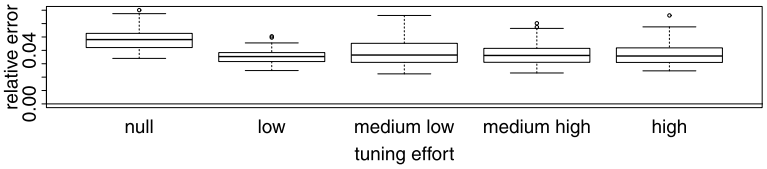
Figure 1 reports the relative performance of $\mathcal{MM}$AS when no parameter is adapted, with the different settings of statically assigned parameters. For each problem, it shows the summary over all sets of instances and all runtimes. The values plotted represent the relative error: for example, in *literature* vs. *low tuning effort*, we compute the difference of the results obtained using the literature setting minus the results obtained using the setting selected through off-line tuning with low tuning effort, divided by the latter. A value greater than zero indicates that the literature setting performs worse than the low tuning effort one (recall we are tackling minimization problems). These results show that in the TSP the use of the three settings (that is, when the statically assigned parameters are set as suggested in the literature, as selected through off-line tuning with low effort, or as selected through off-line tuning with high effort) lead to different performance: as expected, the literature one is worse than both settings returned by off-line tuning, and the setting selected with low tuning effort is worse than the one selected with high tuning effort. This can be seen by observing that the relative error made by the literature setting with respect to the high tuning effort is larger than to the low tuning effort. These differences are statistically significant at the 95% confidence level, according to the Wilcoxon rank-sum test (Wilcoxon 1945). In the QAP, instead, tuning does not improve much with respect to the results obtained with the literature setting, which is quite well performing in itself. The Wilcoxon rank-sum test does not detect any significant difference in the results.

In Sects. 5.1 and 5.2, we will assess the adaptation methods by using the just described way for computing the relative error, and the Wilcoxon rank-sum test at the 95% confidence level for performing statistical tests.

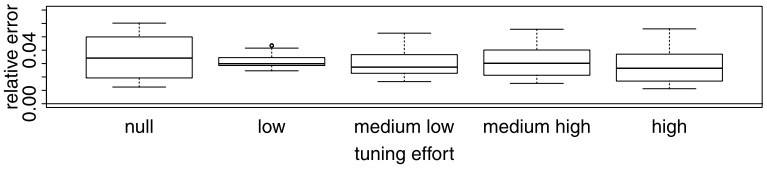
The results reported in Fig. 2 and Fig. 3 indicate the quality of the results achieved by the various versions of $\mathcal{MM}$AS on the TSP and the QAP, respectively. In these figures, we report the relative error with respect to the optimal solutions for the TSP, and with respect to the best-known solutions for the QAP. For the TSP, the optima were determined by using the publicly available Concorde solver (Applegate et al. 2003). For the QAP, it is unfeasible to compute the optimal solution of the instances we use. We therefore consider as a good upper



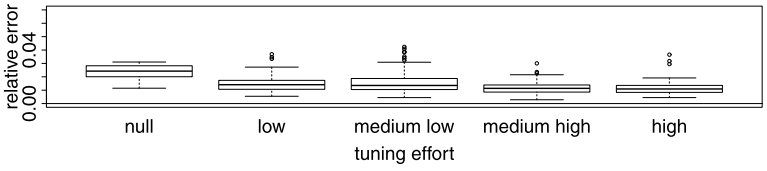

(a) set $TSP(2000, u)$ short runtime



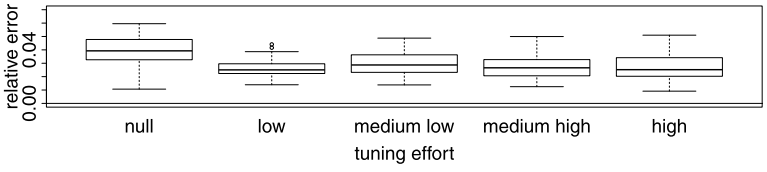

(b) set $TSP(2000, c)$ short runtime



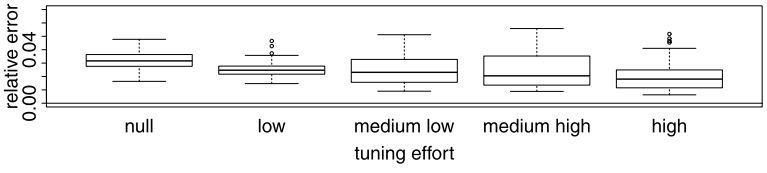

(c) set $TSP(2000, x)$ short runtime




(d) set $TSP(x, u)$ short runtime



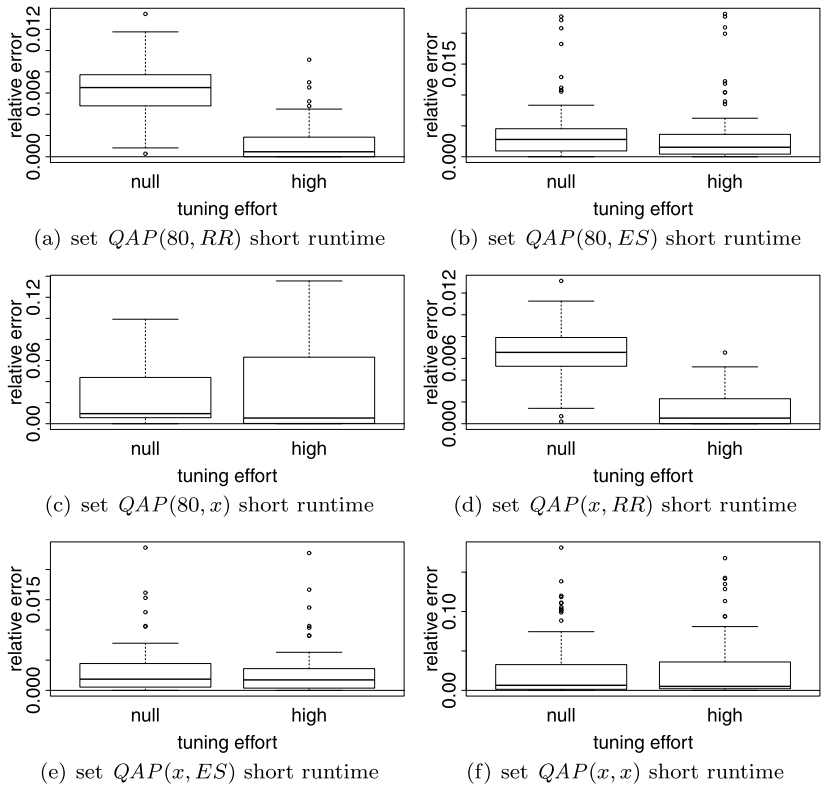

(e) set $TSP(x, c)$ short runtime




(f) set $TSP(x, x)$ short runtime

**Fig. 2** Boxplots of the relative error with respect to the optimal solution when no parameter is adapted. Short runtime on TSP instances. The *x axis* reports the tuning effort. Null tuning effort corresponds to the literature version

bound for the optimal value the result obtained by $\mathcal{MM}$AS when the computation time available is one order of magnitude larger than the longest runtime fixed in the experiments,




(a) set $QAP(80, RR)$ short runtime (b) set $QAP(80, ES)$ short runtime

(c) set $QAP(80, x)$ short runtime (d) set $QAP(x, RR)$ short runtime

(e) set $QAP(x, ES)$ short runtime (f) set $QAP(x, x)$ short runtime

**Fig. 3** Boxplots of the relative error with respect to the best-known solution when no parameter is adapted. Short runtime on QAP instances. The *x axis* reports the tuning effort. Null tuning effort corresponds to the literature version

that is, 290 seconds. As it can be seen in these figures, there is a quite clear difference in the performance of the $\mathcal{MM}$AS configurations from different tuning budgets. Moreover, under all the experimental conditions we consider, there is margin for adaptation methods to improve the performance. Only the results achieved in the short runtime are reported here. The results obtained for the long runtime are available in Pellegrini et al. (2010a), and they confirm these observations.

### 5.1 Traveling salesman problem

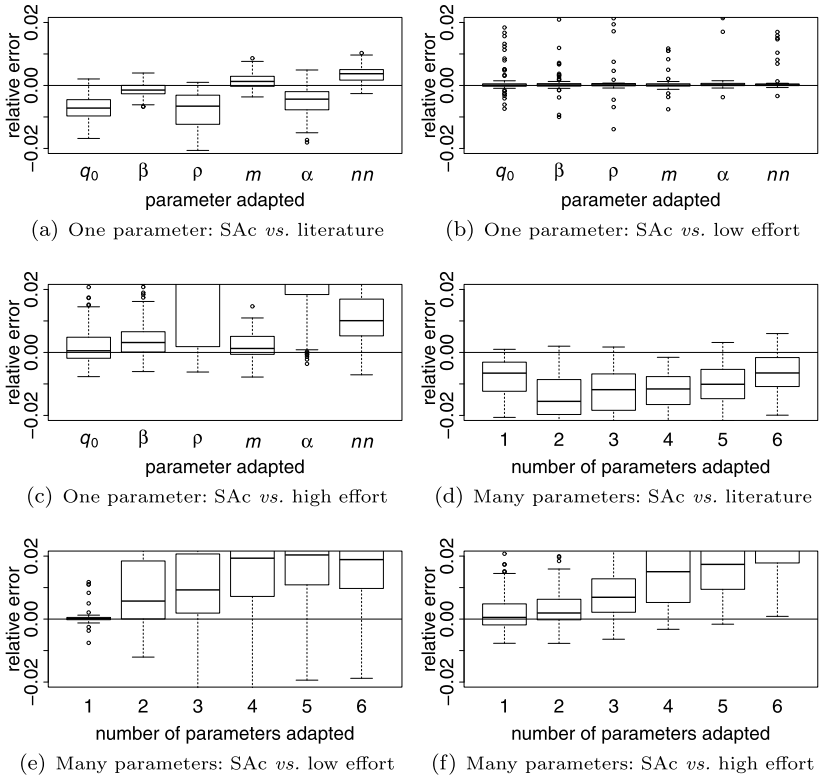
In Table 4 and Table 5, we show the average relative error made on the TSP by the five methods described in Sect. 3 when adapting one parameter. The reference results for the computation are the ones obtained by $\mathcal{MM}$AS when no parameter is adapted. We show in the two tables the error made when the setting of statically assigned parameters is either the one suggested in the literature, or the one selected through off-line tuning with low tuning effort. We compute the average relative error for each set of instances and each runtime. In Fig. 4 and Fig. 5, we report the boxplots of these relative errors for short runs on the instance sets $TSP(2000, u)$ and $TSP(x, x)$ for adaptation method SAc. SAc usually gets the best performance compared to the other adaptation methods. Still, the results achieved with the other methods appear qualitatively equivalent to those of SAc and they are available in

**Table 4** Average relative error for each set of instances of the TSP. Comparison between $\mathcal{MM}$AS applying each of the five parameter adaptation methods to one single parameter, and the version with static parameter settings (no adaptation). Statically assigned parameters are set as suggested in the literature. Given is the relative error for each set of instances between the version with parameter adaptation and the statically assigned parameter settings for $\mathcal{MM}$AS. When the relative error is followed by a bullet, no adaptation is statistically better than adaptation. When it is followed by a star, no adaptation is statistically worse than adaptation

| instance set | runs | $q_0$ | $\beta$ | $\rho$ | $m$ | $\alpha$ | $nn$ |
|---|---|---|---|---|---|---|---|
| SAc | | | | | | | |
| $TSP(2000, u)$ | short | $-0.0072^{\star}$ | $-0.0013^{\star}$ | $-0.0080^{\star}$ | $0.0014^{\bullet}$ | $-0.0050^{\star}$ | $0.0034^{\bullet}$ |
| $TSP(2000, c)$ | short | $-0.0024^{\star}$ | $0.0007^{\star}$ | $-0.0023^{\star}$ | $0.0015^{\bullet}$ | $-0.0019^{\star}$ | $0.0030^{\bullet}$ |
| $TSP(2000, x)$ | short | $-0.0053^{\star}$ | $-0.0007^{\star}$ | $-0.0049^{\star}$ | $0.0015^{\bullet}$ | $-0.0035^{\star}$ | $0.0028^{\bullet}$ |
| $TSP(x, u)$ | short | $-0.0101^{\star}$ | $-0.0009^{\star}$ | $-0.0099^{\star}$ | $0.0009^{\bullet}$ | $-0.0060^{\star}$ | $0.0031^{\bullet}$ |
| $TSP(x, c)$ | short | $-0.0035^{\star}$ | $-0.0001$ | $-0.0035^{\star}$ | $0.0000$ | $-0.0030^{\star}$ | $0.0020^{\bullet}$ |
| $TSP(x, x)$ | short | $-0.0066^{\star}$ | $0.0000$ | $-0.0067^{\star}$ | $0.0012^{\bullet}$ | $-0.0043^{\star}$ | $0.0026^{\bullet}$ |
| $TSP(2000, u)$ | long | $0.0008$ | $0.0025$ | $0.0025^{\bullet}$ | $0.0005^{\bullet}$ | $0.0033$ | $0.0024^{\bullet}$ |
| SAcb | | | | | | | |
| $TSP(2000, u)$ | short | $-0.0062^{\star}$ | $-0.0008^{\star}$ | $-0.0044^{\star}$ | $0.0005$ | $-0.0050^{\star}$ | $0.0031^{\bullet}$ |
| $TSP(2000, c)$ | short | $-0.0027^{\star}$ | $0.0013^{\star}$ | $-0.0016^{\star}$ | $0.0014^{\bullet}$ | $-0.0018^{\star}$ | $0.0028^{\bullet}$ |
| $TSP(2000, x)$ | short | $-0.0044^{\star}$ | $-0.0002$ | $-0.0029^{\star}$ | $0.0006$ | $-0.0043^{\star}$ | $0.0027^{\bullet}$ |
| $TSP(x, u)$ | short | $-0.0079^{\star}$ | $-0.0004$ | $-0.0057^{\star}$ | $-0.0010$ | $-0.0069^{\star}$ | $0.0031^{\bullet}$ |
| $TSP(x, c)$ | short | $-0.0041^{\star}$ | $-0.0002$ | $-0.0017^{\star}$ | $0.0000$ | $-0.0031^{\star}$ | $0.0013^{\bullet}$ |
| $TSP(x, x)$ | short | $-0.0054^{\star}$ | $0.0002$ | $-0.0025^{\star}$ | $-0.0003$ | $-0.0049^{\star}$ | $0.0025^{\bullet}$ |
| $TSP(2000, u)$ | long | $0.0038^{\bullet}$ | $0.0036^{\bullet}$ | $0.0067$ | $0.0033^{\bullet}$ | $0.0052$ | $0.0051^{\bullet}$ |
| SAcm | | | | | | | |
| $TSP(2000, u)$ | short | $-0.0122^{\star}$ | $-0.0006^{\star}$ | $-0.0060^{\star}$ | $0.0035^{\bullet}$ | $-0.0018^{\star}$ | $0.0028^{\bullet}$ |
| $TSP(2000, c)$ | short | $-0.0041^{\star}$ | $0.0007$ | $-0.0008$ | $0.0019^{\bullet}$ | $-0.0017^{\star}$ | $0.0029^{\bullet}$ |
| $TSP(2000, x)$ | short | $-0.0069^{\star}$ | $0.0000$ | $-0.0037^{\star}$ | $0.0022^{\bullet}$ | $-0.0026^{\star}$ | $0.0031^{\bullet}$ |
| $TSP(x, u)$ | short | $-0.0141^{\star}$ | $0.0000$ | $-0.0081^{\star}$ | $0.0027^{\bullet}$ | $-0.0017^{\star}$ | $0.0028^{\bullet}$ |
| $TSP(x, c)$ | short | $-0.0050^{\star}$ | $-0.0001$ | $-0.0020^{\star}$ | $0.0009^{\bullet}$ | $-0.0031^{\star}$ | $0.0015^{\bullet}$ |
| $TSP(x, x)$ | short | $-0.0090^{\star}$ | $0.0005$ | $-0.0042^{\star}$ | $0.0020^{\bullet}$ | $-0.0018^{\star}$ | $0.0025^{\bullet}$ |
| $TSP(2000, u)$ | long | $0.0072$ | $0.0089^{\bullet}$ | $0.0086$ | $0.0071^{\bullet}$ | $0.0096$ | $0.0081^{\bullet}$ |
| SAa | | | | | | | |
| $TSP(2000, u)$ | short | $-0.0095^{\star}$ | $-0.0014^{\star}$ | | $-0.0105^{\star}$ | $0.0024$ | |
| $TSP(2000, c)$ | short | $-0.0013^{\star}$ | $0.0029^{\bullet}$ | | $-0.0014^{\star}$ | $0.0029^{\bullet}$ | |
| $TSP(2000, x)$ | short | $-0.0053^{\star}$ | $0.0000$ | | $-0.0062^{\star}$ | $0.0023^{\bullet}$ | |
| $TSP(x, u)$ | short | $-0.0113^{\star}$ | $-0.0004$ | | $-0.0126^{\star}$ | $0.0016^{\bullet}$ | |
| $TSP(x, c)$ | short | $-0.0038^{\star}$ | $0.0007$ | | $-0.0049^{\star}$ | $0.0011^{\bullet}$ | |
| $TSP(x, x)$ | short | $-0.0067^{\star}$ | $0.0008$ | | $-0.0081^{\star}$ | $0.0019^{\bullet}$ | |
| $TSP(2000, u)$ | long | $0.0154$ | $0.0022^{\bullet}$ | | $0.0141$ | $0.0023^{\bullet}$ | |

**Table 4** (*Continued*)

| instance set | runs | $q_0$ | $\beta$ | $\rho$ | $m$ | $\alpha$ | $nn$ |
|---|---|---|---|---|---|---|---|
| LS | | | | | | | |
| $TSP(2000, u)$ | short | $-0.0095^{\bigstar}$ | $-0.0014^{\bigstar}$ | | $-0.0105^{\bigstar}$ | 0.0024 | |
| $TSP(2000, c)$ | short | $-0.0013^{\bigstar}$ | $0.0029^{\bullet}$ | | $-0.0014^{\bigstar}$ | $0.0029^{\bullet}$ | |
| $TSP(2000, x)$ | short | $-0.0053^{\bigstar}$ | 0.0000 | | $-0.0062^{\bigstar}$ | $0.0023^{\bullet}$ | |
| $TSP(x, u)$ | short | $-0.0113^{\bigstar}$ | $-0.0004$ | | $-0.0126^{\bigstar}$ | $0.0016^{\bullet}$ | |
| $TSP(x, c)$ | short | $-0.0038^{\bigstar}$ | 0.0007 | | $-0.0049^{\bigstar}$ | $0.0011^{\bullet}$ | |
| $TSP(x, x)$ | short | $-0.0067^{\bigstar}$ | 0.0008 | | $-0.0081^{\bigstar}$ | $0.0019^{\bullet}$ | |
| $TSP(2000, u)$ | long | 0.0154 | $0.0022^{\bullet}$ | | 0.0141 | $0.0023^{\bullet}$ | |




(a) One parameter: SAc *vs.* literature (b) One parameter: SAc *vs.* low effort

(c) One parameter: SAc *vs.* high effort (d) Many parameters: SAc *vs.* literature

(e) Many parameters: SAc *vs.* low effort (f) Many parameters: SAc *vs.* high effort

**Fig. 4** Relative error of $\mathcal{MM}$AS with parameter adaptation, adapting parameters with respect to adapting none in TSP. Short runs on instances of set $TSP(2000, u)$. Plots (**a**) to (**c**) refer to the case of a single parameter adapted: the *x axis* reports the specific parameter adapted. Plots (**d**) to (**f**) refer to the case of many parameters adapted: the *x axis* reports the number of parameters adapted. Adaptation method: SAc

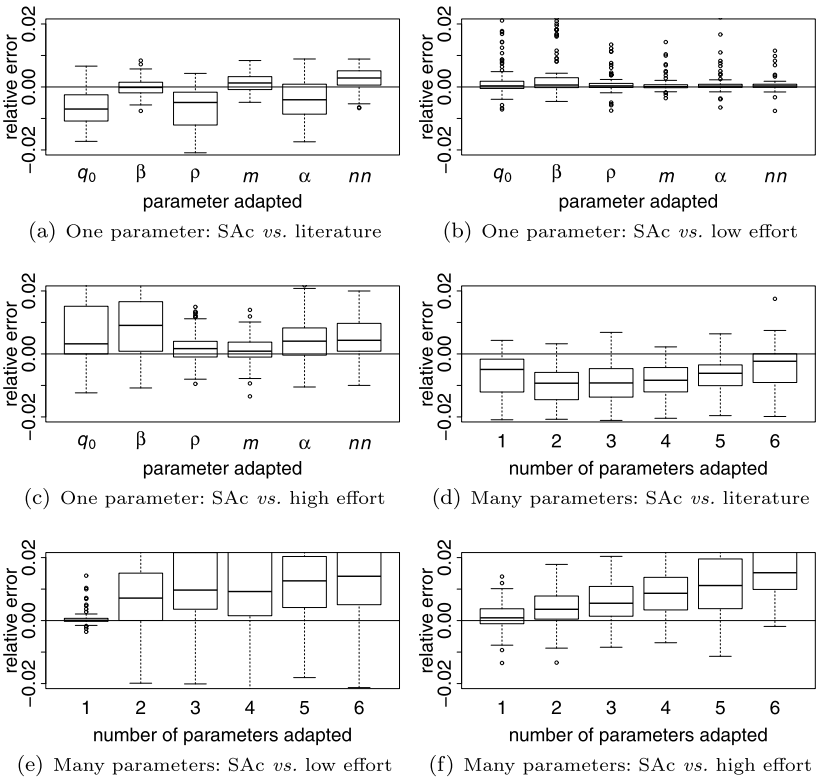
Pellegrini et al. (2010a). In the same figures, we also report the relative error made when several parameters are adapted. For each number of parameters to be adapted, we plot the results obtained with the best set $A$. We show also the results achieved when statically assigned parameters are set as selected by off-line tuning with high tuning effort. We report

**Table 5** Average relative error for each set of instances of the TSP. Comparison between $\mathcal{MM}$AS applying each of the five parameter adaptation methods to one single parameter, and the version with static parameter settings (no adaptation). Statically assigned parameters are set as selected through off-line tuning with low tuning effort. Given is the relative error for each set of instances between the version with parameter adaptation and the statically assigned parameter settings for $\mathcal{MM}$AS. When the relative error is followed by a bullet, no adaptation is statistically better than adaptation. When it is followed by a star, no adaptation is statistically worse than adaptation

| instance set | runs | $q_0$ | $\beta$ | $\rho$ | $m$ | $\alpha$ | $nn$ |
|---|---|---|---|---|---|---|---|
| SAc | | | | | | | |
| $TSP(2000, u)$ | short | 0.0012$^\bullet$ | 0.0011$^\bullet$ | 0.0034$^\bullet$ | 0.0004$^\bullet$ | 0.0036$^\bullet$ | 0.0012$^\bullet$ |
| $TSP(2000, c)$ | short | 0.0023$^\bullet$ | 0.0021$^\bullet$ | 0.0014$^\bullet$ | 0.0005$^\bullet$ | 0.0013$^\bullet$ | 0.0014$^\bullet$ |
| $TSP(2000, x)$ | short | 0.0046$^\bullet$ | 0.0040$^\bullet$ | 0.0011$^\bullet$ | 0.0002$^\bullet$ | 0.0016$^\bullet$ | 0.0017$^\bullet$ |
| $TSP(x, u)$ | short | 0.0004 | 0.0011$^\bullet$ | 0.0072$^\bullet$ | 0.0006$^\bullet$ | 0.0050$^\bullet$ | 0.0016$^\bullet$ |
| $TSP(x, c)$ | short | 0.0014$^\bullet$ | 0.0018$^\bullet$ | 0.0012$^\bullet$ | 0.0000 | 0.0013$^\bullet$ | 0.0008$^\bullet$ |
| $TSP(x, x)$ | short | 0.0022$^\bullet$ | 0.0040$^\bullet$ | 0.0010$^\bullet$ | 0.0007$^\bullet$ | 0.0010$^\bullet$ | 0.0006$^\bullet$ |
| $TSP(2000, u)$ | long | 0.0009$^\bullet$ | 0.0025$^\bullet$ | 0.0025$^\bullet$ | 0.0006$^\bullet$ | 0.0043$^\bullet$ | 0.0025$^\bullet$ |
| SAcb | | | | | | | |
| $TSP(2000, u)$ | short | 0.0029$^\bullet$ | 0.0029$^\bullet$ | 0.0087$^\bullet$ | 0.0028$^\bullet$ | 0.0057$^\bullet$ | 0.0035$^\bullet$ |
| $TSP(2000, c)$ | short | 0.0052$^\bullet$ | 0.0059$^\bullet$ | 0.0058$^\bullet$ | 0.0048$^\bullet$ | 0.0048$^\bullet$ | 0.0056$^\bullet$ |
| $TSP(2000, x)$ | short | 0.0144$^\bullet$ | 0.0072$^\bullet$ | 0.0066$^\bullet$ | 0.0070$^\bullet$ | 0.0100$^\bullet$ | 0.0120$^\bullet$ |
| $TSP(x, u)$ | short | 0.0022$^\bullet$ | 0.0038$^\bullet$ | 0.0117$^\bullet$ | 0.0024$^\bullet$ | 0.0065$^\bullet$ | 0.0039$^\bullet$ |
| $TSP(x, c)$ | short | 0.0027$^\bullet$ | 0.0032$^\bullet$ | 0.0028$^\bullet$ | 0.0030$^\bullet$ | 0.0035$^\bullet$ | 0.0036$^\bullet$ |
| $TSP(x, x)$ | short | 0.0084$^\bullet$ | 0.0048$^\bullet$ | 0.0039$^\bullet$ | 0.0057$^\bullet$ | 0.0066$^\bullet$ | 0.0073$^\bullet$ |
| $TSP(2000, u)$ | long | 0.0036$^\bullet$ | 0.0039$^\bullet$ | 0.0077$^\bullet$ | 0.0036$^\bullet$ | 0.0062$^\bullet$ | 0.0053$^\bullet$ |
| SAcm | | | | | | | |
| $TSP(2000, u)$ | short | 0.0054$^\bullet$ | 0.0056$^\bullet$ | 0.0091$^\bullet$ | 0.0055$^\bullet$ | 0.0092$^\bullet$ | 0.0069$^\bullet$ |
| $TSP(2000, c)$ | short | 0.0055$^\bullet$ | 0.0067$^\bullet$ | 0.0060$^\bullet$ | 0.0055$^\bullet$ | 0.0061$^\bullet$ | 0.0062$^\bullet$ |
| $TSP(2000, x)$ | short | 0.0065$^\bullet$ | 0.0106$^\bullet$ | 0.0066$^\bullet$ | 0.0070$^\bullet$ | 0.0082$^\bullet$ | 0.0083$^\bullet$ |
| $TSP(x, u)$ | short | 0.0031$^\bullet$ | 0.0048$^\bullet$ | 0.0117$^\bullet$ | 0.0034$^\bullet$ | 0.0105$^\bullet$ | 0.0057$^\bullet$ |
| $TSP(x, c)$ | short | 0.0049$^\bullet$ | 0.0069$^\bullet$ | 0.0059$^\bullet$ | 0.0050$^\bullet$ | 0.0069$^\bullet$ | 0.0052$^\bullet$ |
| $TSP(x, x)$ | short | 0.0074$^\bullet$ | 0.0095$^\bullet$ | 0.0070$^\bullet$ | 0.0065$^\bullet$ | 0.0076$^\bullet$ | 0.0077$^\bullet$ |
| $TSP(2000, u)$ | long | 0.0078$^\bullet$ | 0.0084$^\bullet$ | 0.0096$^\bullet$ | 0.0076$^\bullet$ | 0.0106$^\bullet$ | 0.0087$^\bullet$ |
| SAa | | | | | | | |
| $TSP(2000, u)$ | short | 0.0070$^\bullet$ | 0.0070$^\bullet$ | | 0.0070$^\bullet$ | 0.0070 | |
| $TSP(2000, c)$ | short | 0.0120$^\bullet$ | 0.0121$^\bullet$ | | 0.0120$^\bullet$ | 0.0119$^\bullet$ | |
| $TSP(2000, x)$ | short | 0.0066$^\bullet$ | 0.0067$^\bullet$ | | 0.0066$^\bullet$ | 0.0066$^\bullet$ | |
| $TSP(x, u)$ | short | 0.0064$^\bullet$ | 0.0064$^\bullet$ | | 0.0065$^\bullet$ | 0.0065$^\bullet$ | |
| $TSP(x, c)$ | short | 0.0056$^\bullet$ | 0.0057$^\bullet$ | | 0.0057$^\bullet$ | 0.0056$^\bullet$ | |
| $TSP(x, x)$ | short | 0.0035$^\bullet$ | 0.0036$^\bullet$ | | 0.0036$^\bullet$ | 0.0036$^\bullet$ | |
| $TSP(2000, u)$ | long | 0.0027$^\bullet$ | 0.0027$^\bullet$ | | 0.0028$^\bullet$ | 0.0027$^\bullet$ | |

**Table 5** (*Continued*)

| instance set | runs | $q_0$ | $\beta$ | $\rho$ | $m$ | $\alpha$ | $nn$ |
|---|---|---|---|---|---|---|---|
| LS | | | | | | | |
| $TSP(2000, u)$ | short | 0.0069• | 0.0074• | | 0.0095• | 0.0072• | |
| $TSP(2000, c)$ | short | 0.0118• | 0.0119• | | 0.0110• | 0.0118• | |
| $TSP(2000, x)$ | short | 0.0064• | 0.0071• | | 0.0055• | 0.0056• | |
| $TSP(x, u)$ | short | 0.0062• | 0.0070• | | 0.0110• | 0.0064• | |
| $TSP(x, c)$ | short | 0.0054• | 0.0049• | | 0.0052• | 0.0053• | |
| $TSP(x, x)$ | short | 0.0037• | 0.0035• | | 0.0036• | 0.0039• | |
| $TSP(2000, u)$ | long | 0.0026• | 0.0042• | | 0.0072• | 0.0029• | |




(a) One parameter: SAc *vs.* literature (b) One parameter: SAc *vs.* low effort

(c) One parameter: SAc *vs.* high effort (d) Many parameters: SAc *vs.* literature

(e) Many parameters: SAc *vs.* low effort (f) Many parameters: SAc *vs.* high effort

**Fig. 5** Relative error of $\mathcal{MM}$AS with parameter adaptation, adapting parameters with respect to adapting none in TSP. Short runs on instances of set $TSP(x, x)$. Plots (**a**) to (**c**) refer to the case of a single parameter adapted: the *x axis* reports the specific parameter adapted. Plots (**d**) to (**f**) refer to the case of many parameters adapted: the *x axis* reports the number of parameters adapted. Adaptation method: SAc

these latter results for showing that, when varying the tuning effort, the general pattern of the relative error obtained by adaptation methods as a function of the cardinality of $A$ remains the same, but the magnitude of this relative error increases as a function of the tuning effort.

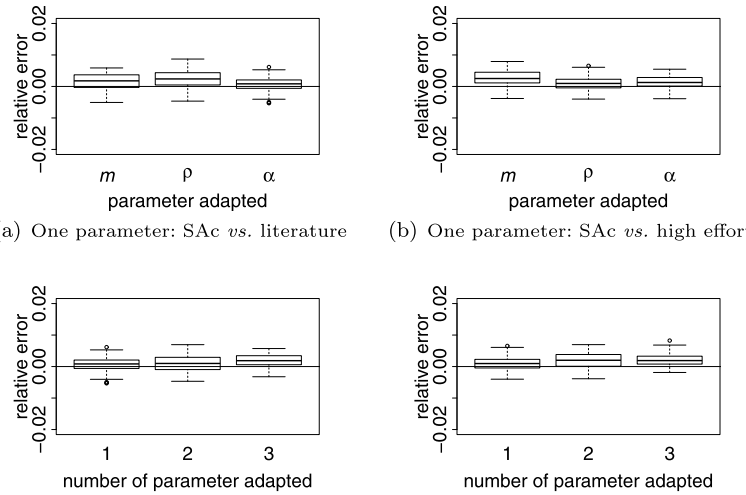
**Table 6** Parameters that allow each adaptation method to achieve the best and the worse overall performance on TSP instances

| adaptation | best performance parameter | | | worst performance parameter | | |
|---|---|---|---|---|---|---|
| method | literature | low effort | high effort | literature | low effort | high effort |
| SAc | $q_0$ | $m$ | $m$ | $nn$ | $\alpha$ | $\alpha$ |
| SAcb | $q_0$ | $m$ | $m$ | $nn$ | $\rho$ | $\alpha$ |
| SAcm | $q_0$ | $q_0, m$ | $m$ | $nn$ | $\alpha$ | $\alpha$ |
| SAa | $\alpha$ | $q_0, \beta, \alpha, m$ | $nn$ | $m$ | $q_0, \beta, \alpha, m$ | $\alpha$ |
| LS | $\alpha$ | $q_0, nn$ | $nn$ | $m$ | $\alpha$ | $\alpha$ |

The results show that the relative performance achieved by $\mathcal{MM}$AS when one of its parameters is adapted strongly depends on the setting of the statically assigned parameters. In particular, when the setting of the statically assigned parameters makes the algorithm achieve quite poor results, the application of an adaptation method may improve the quality of the results: when the literature setting is used for short runs on the classes of instances considered here (Pellegrini et al. 2010b), the adaptation methods achieve better performance than the statically assigned version, as shown in Table 4, Figs. 4(a), 4(d) and Figs. 5(a), 5(d). When the setting of the statically assigned parameters makes the algorithm achieve rather good results, that is, when off-line tuning selects the setting of statically assigned parameters, even with an extremely low tuning effort, the adaptation methods proposed in the literature for ACO perform very poorly. This can be seen in Table 5, Figs. 4(b), 4(e) and Figs. 5(b), 5(f). Even if it is not evident from the boxplots in either Fig. 4(b) or Fig. 5(b), Table 4 shows that the difference is statistically significant in favor of the $\mathcal{MM}$AS version when no parameter is adapted, except when adapting parameter $q_0$ on instances of set $TSP(x, u)$ and parameter $m$ on instances of set $TSP(x, c)$. The statistical significance of the differences is, in this context, more relevant than their absolute size. The fact that the degradation of the performance brought by the adaptation methods is statistically significant guarantees that we are indeed facing a degradation, even if sometimes rather small, and not a neutral contribution (or even a slight improvement) hidden by the experimental noise. If the tuning effort increases, thus in principle if the expected quality of the selected setting improves, the difference between adapting parameters or not increases as well (Fig. 4(c), 4(f), and Figs. 5(c), 5(f)). In fact, in the strong majority of the cases we analyzed, the difference in the performance is statistically significant in favor of the version with static parameter settings. This result is irrespective of the instance set, the cardinality of the set of parameters adapted, the adaptation method, and the tuned setting of the statically assigned parameters.

By analyzing the results achieved when adapting more than one parameter, we can see that the more parameters are adapted, the worse the performance (Figs. 4(d)–4(f) and Figs. 5(d)–5(f)). This result holds regardless the instance set, runtime, setting of the statically assigned parameters, and adaptation method. Only in few cases, adapting either two or three parameters is better than adapting one, but in all the experiments adapting five or six parameters is disadvantageous. In all cases, if adapting one parameter is worse than adapting none, the same holds when adapting two or three. This is true even if we overestimate the quality of the results obtainable by adaptation methods, namely by considering the a posteriori best case.

The best set of parameters to be adapted depends on the setting of statically assigned parameters. Focusing on the case in which $|A|$ is one, Table 6 reports the parameter for which each adaptation scheme achieves the best and the worst overall performance, in terms




(a) One parameter: SAc *vs.* literature (b) One parameter: SAc *vs.* high effort

(c) Many parameters: SAc *vs.* literature (d) Many parameters: SAc *vs.* high effort

**Fig. 6** Relative error of $\mathcal{MM}$AS with parameter adaptation, adapting parameters with respect to adapting none in QAP. Short runs on instances of set $QAP(80, RR)$. Plots (**a**) and (**b**) refer to the case of a single parameter adapted: the *x axis* reports the specific parameter adapted. Plots (**c**) and (**d**) refer to the case of many parameters adapted: the *x axis* reports the number of parameters adapted. Adaptation method: SAc

of average relative error computed over all instance sets and runtimes for each setting of statically assigned parameters. There is no single parameter that, when adapted, resulted in the best performance across all the experimental setups and sets of instances.

The heterogeneity of the set of instances does not appear to have a predictable impact on the results, as visible in Table 4 and Table 5, and in Fig. 4 and Fig. 5. This observation contradicts the intuition according to which the more heterogeneous the set of instances, the more advantageous the use of an adaptation method. The same observation holds for the use of different runtimes.

## 5.2 Quadratic assignment problem

In Table 7, we show the average relative error made on the QAP by the five methods described in Sect. 3 when adapting one parameter. The reference results for the computation are the ones obtained by $\mathcal{MM}$AS when no parameter is adapted. In the two parts of the table, we show the error made when the setting of the statically assigned parameters is either the one suggested in the literature, or the one selected through off-line tuning with high tuning effort. We compute the average relative error for each set of instances and each runtime. We report only the results for the short runtime, since the long runtime does not lead to qualitatively different conclusions; the results for the long runtime are available in Pellegrini et al. (2010a). In Fig. 6 and Fig. 7, we report the boxplots of these relative errors for short runs on the sets of instances $QAP(80, RR)$ and $QAP(x, x)$ for the adaptation method SAc, which is in general the best performing. The results achieved with the other methods appear qualitatively equivalent; we refer to Pellegrini et al. (2010a) for the full results. In the same figures, we report the relative error made when several parameters are adapted, plotting for each number of parameters to be adapted the results for the best set $A$.

Differently from the TSP, applying an adaptation method when solving the QAP is always a disadvantage in terms of solution quality (see Table 7). For all instance sets, for any

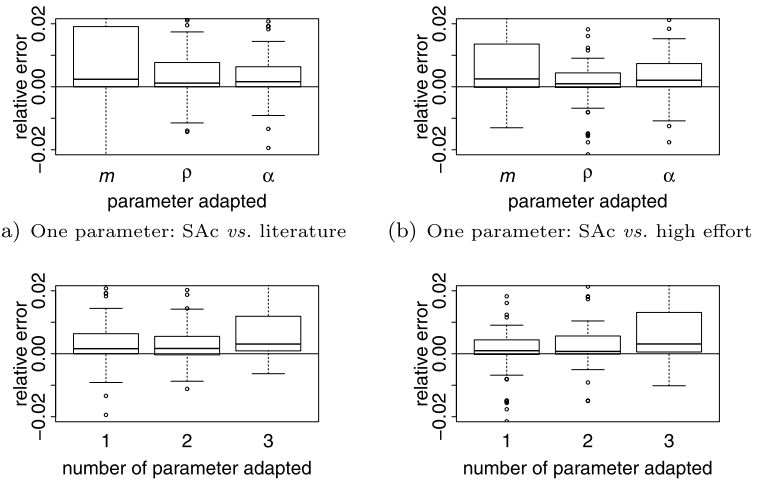
**Table 7** Average relative error for each set of instances of the QAP. Comparison between $\mathcal{MM}$AS applying each of the five parameter adaptation methods to one single parameter, and the version with static parameter settings (no adaptation). Given is the relative error for each set of instances between the version with parameter adaptation and the statically assigned parameter settings for $\mathcal{MM}$AS. When the relative error is followed by a bullet, no adaptation is statistically better than adaptation. When it is followed by a star, no adaptation is statistically worse than adaptation

| | | Setting of statically assigned parameters: literature | | | Setting of statically assigned parameters: high effort | | |
|---|---|---|---|---|---|---|---|
| instance set | runs | $m$ | $\rho$ | $\alpha$ | $m$ | $\rho$ | $\alpha$ |
| SAc | | | | | | | |
| $QAP(80, RR)$ | short | 0.0016$^\bullet$ | 0.0024$^\bullet$ | 0.0007$^\bullet$ | 0.0026$^\bullet$ | 0.0010$^\bullet$ | 0.0013$^\bullet$ |
| $QAP(80, ES)$ | short | 0.0033$^\bullet$ | 0.0035$^\bullet$ | 0.0026$^\bullet$ | 0.0018$^\bullet$ | 0.0009$^\bullet$ | 0.0011$^\bullet$ |
| $QAP(80, x)$ | short | 0.0185$^\bullet$ | 0.0217$^\bullet$ | 0.0126$^\bullet$ | 0.0242$^\bullet$ | 0.0029$^\bullet$ | 0.0102$^\bullet$ |
| $QAP(x, RR)$ | short | 0.0035$^\bullet$ | 0.0020$^\bullet$ | 0.0013$^\bullet$ | 0.0027$^\bullet$ | 0.0017$^\bullet$ | 0.0017$^\bullet$ |
| $QAP(x, ES)$ | short | 0.0030$^\bullet$ | 0.0028$^\bullet$ | 0.0022$^\bullet$ | 0.0010$^\bullet$ | 0.0008$^\bullet$ | 0.0008$^\bullet$ |
| $QAP(x, x)$ | short | 0.0154$^\bullet$ | 0.0094$^\bullet$ | 0.0070$^\bullet$ | 0.0188$^\bullet$ | 0.0053$^\bullet$ | 0.0082$^\bullet$ |
| SAcb | | | | | | | |
| $QAP(80, RR)$ | short | 0.0017$^\bullet$ | 0.0016$^\bullet$ | 0.0004$^\bullet$ | 0.0022$^\bullet$ | 0.0008$^\bullet$ | 0.0013$^\bullet$ |
| $QAP(80, ES)$ | short | 0.0033$^\bullet$ | 0.0031$^\bullet$ | 0.0018$^\bullet$ | 0.0011$^\bullet$ | 0.0007$^\bullet$ | 0.0015$^\bullet$ |
| $QAP(80, x)$ | short | 0.0186$^\bullet$ | 0.0087$^\bullet$ | 0.0077$^\bullet$ | 0.0201$^\bullet$ | 0.0079$^\bullet$ | 0.0113$^\bullet$ |
| $QAP(x, RR)$ | short | 0.0030$^\bullet$ | 0.0018$^\bullet$ | 0.0014$^\bullet$ | 0.0028$^\bullet$ | 0.0015$^\bullet$ | 0.0016$^\bullet$ |
| $QAP(x, ES)$ | short | 0.0030$^\bullet$ | 0.0027$^\bullet$ | 0.0019$^\bullet$ | 0.0009$^\bullet$ | 0.0006$^\bullet$ | 0.0011$^\bullet$ |
| $QAP(x, x)$ | short | 0.0189$^\bullet$ | 0.0079$^\bullet$ | 0.0082$^\bullet$ | 0.0187$^\bullet$ | 0.0061$^\bullet$ | 0.0072$^\bullet$ |
| SAcm | | | | | | | |
| $QAP(80, RR)$ | short | 0.0016$^\bullet$ | 0.0036$^\bullet$ | 0.0006$^\bullet$ | 0.0024$^\bullet$ | 0.0011$^\bullet$ | 0.0010$^\bullet$ |
| $QAP(80, ES)$ | short | 0.0031$^\bullet$ | 0.0038$^\bullet$ | 0.0023$^\bullet$ | 0.0012$^\bullet$ | 0.0009$^\bullet$ | 0.0013$^\bullet$ |
| $QAP(80, x)$ | short | 0.0248$^\bullet$ | 0.0252$^\bullet$ | 0.0103$^\bullet$ | 0.0252$^\bullet$ | 0.0057$^\bullet$ | 0.0104$^\bullet$ |
| $QAP(x, RR)$ | short | 0.0027$^\bullet$ | 0.0016$^\bullet$ | 0.0016$^\bullet$ | 0.0027$^\bullet$ | 0.0016$^\bullet$ | 0.0013$^\bullet$ |
| $QAP(x, ES)$ | short | 0.0028$^\bullet$ | 0.0036$^\bullet$ | 0.0021$^\bullet$ | 0.0009$^\bullet$ | 0.0011$^\bullet$ | 0.0012$^\bullet$ |
| $QAP(x, x)$ | short | 0.0178$^\bullet$ | 0.0122$^\bullet$ | 0.0065$^\bullet$ | 0.0224$^\bullet$ | 0.0055$^\bullet$ | 0.0077$^\bullet$ |
| SAa | | | | | | | |
| $QAP(80, RR)$ | short | | | 0.0020$^\bullet$ | | | 0.0033$^\bullet$ |
| $QAP(80, ES)$ | short | | | 0.0045$^\bullet$ | | | 0.0010$^\bullet$ |
| $QAP(80, x)$ | short | | | 0.0228$^\bullet$ | | | 0.0341$^\bullet$ |
| $QAP(x, RR)$ | short | | | 0.0031$^\bullet$ | | | 0.0041$^\bullet$ |
| $QAP(x, ES)$ | short | | | 0.0042$^\bullet$ | | | 0.0007$^\bullet$ |
| $QAP(x, x)$ | short | | | 0.0135$^\bullet$ | | | 0.0226$^\bullet$ |

runtime, for all adaptation methods, for all cardinalities of the set of parameters adapted, and for all the tuning efforts applied for setting statically assigned parameters, the difference is statistically significant in favor of the $\mathcal{MM}$AS version in which no parameter is adapted.

**Table 7** (*Continued*)

| instance set | runs | Setting of statically assigned parameters: literature | | | Setting of statically assigned parameters: high effort | | |
|---|---|---|---|---|---|---|---|
| | | $m$ | $\rho$ | $\alpha$ | $m$ | $\rho$ | $\alpha$ |
| LS | | | | | | | |
| $QAP(80, RR)$ | short | | | 0.0020$^\bullet$ | | | 0.0061$^\bullet$ |
| $QAP(80, ES)$ | short | | | 0.0103$^\bullet$ | | | 0.0084$^\bullet$ |
| $QAP(80, x)$ | short | | | 0.0495$^\bullet$ | | | 0.0490$^\bullet$ |
| $QAP(x, RR)$ | short | | | 0.0031$^\bullet$ | | | 0.0064$^\bullet$ |
| $QAP(x, ES)$ | short | | | 0.0098$^\bullet$ | | | 0.0085$^\bullet$ |
| $QAP(x, x)$ | short | | | 0.0595$^\bullet$ | | | 0.0585$^\bullet$ |




(a) One parameter: SAc *vs.* literature (b) One parameter: SAc *vs.* high effort

(c) Many parameters: SAc *vs.* literature (d) Many parameters: SAc *vs.* high effort

**Fig. 7** Relative error of $\mathcal{MM}$AS with parameter adaptation, adapting parameters with respect to adapting none in QAP. Short runs on instances of set $QAP(x, x)$. Plots (**a**) and (**b**) refer to the case of a single parameter adapted: the *x axis* reports the specific parameter adapted. Plots (**c**) and (**d**) refer to the case of many parameters adapted: the *x axis* reports the number of parameters adapted. Adaptation method: SAc

The adoption of an adaptation method is disadvantageous even in the case of a null tuning effort, that is, for the literature version. We expect this latter to be the context in which adaptation methods perform the best. Thus, using parameter adaptation will increase the relative error when any tuning effort is devoted to selecting the setting of statically assigned parameters: it is not necessary to test several levels of tuning effort, as we did for the TSP.

Even if it is less evident than in the TSP case, increasing the cardinality of the set of parameters adapted is not advantageous (Figs. 6(c), 6(d), Figs. 7(c), 7(d)).

As in the TSP, it is not possible to identify either the best adaptation method, or the best set of parameters to be adapted: they vary as a function of both the set of instances tackled and the runtime, as it can be seen in Table 7. Moreover, neither the heterogeneity of the set of instances nor the runtime influence the results in a predictable way.

## 6 Conclusions

In this paper, we empirically showed that the state-of-the-art parameter adaptation methods often worsen the performance of ACO algorithms in case they are applied out of the particular context for which they were proposed. The context that we considered is based on a high performing state-of-the-art ACO algorithm and two classical combinatorial optimization problems. We applied five adaptation methods to $\mathcal{MM}$AS for both the traveling salesman problem and the quadratic assignment problem. We ran an extensive experimental analysis, considering several experimental setups. We exploited these setups for either refuting or corroborating four conjectures. Two of them were corroborated by the results, and two were refuted. We verified that:

(i) the more parameters adapted, and
(ii) the higher the quality of the results achieved by the algorithm,

the smaller the improvement (or rather the larger the worsening) of the performance of the algorithm that adaptation methods can produce. We could not verify that:

(iii) the lower the heterogeneity of the set of instances to be tackled, and
(iv) the shorter the runtime,

the smaller the improvement (or the larger the worsening) of the performance of the algorithm that adaptation methods can produce.

The ineffectiveness of the adaptation methods is evident in the results. We could observe only one exception to this conclusion, when the setting of statically assigned parameters leads the algorithm to achieve relatively low quality results. Otherwise, we could obtain better quality results without adapting any parameter during the runs. The number of parameters adapted does not affect these conclusions: even if considering the (unrealistic) a posteriori best case for the adaptation methods, applying an adaptation method worsens the results unless the setting of statically assigned parameters is performing poorly.

We will devote future research to the application of successful methods proposed for other metaheuristics to ACO. An example of such methods is the rank-based multi-armed bandit (Fialho 2010) that was proposed for genetic algorithms. Despite the poor performance achieved by the methods proposed for ACO, in fact, our results do not contradict in absolute terms the merits of adaptation methods. In some specific cases, and when the appropriate method is used for adapting only the appropriate parameter(s), adaptation methods may provide an advantage. In future works, we will try to characterize situations in which parameter adaptation is useful. Moreover, we will devote further studies to the understanding of the impact that each parameter has on the behavior of the algorithms. This may help in identifying the appropriate parameters to be adapted. In particular, we will try to combine off-line tuning and parameter adaptation by using off-line tuning to decide which parameters to adapt, and letting an adaptation method operate only on those parameters during the run.

**Acknowledgements** 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 and by the European Union through the ERC Advanced Grant "E-SWARM: Engineering Swarm Intelligence Systems" (contract 246939). Mauro Birattari and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are Research Associates. The work of Paola Pellegrini is funded by a Bourse d'excellence Wallonie-Bruxelles International.

## References


Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. In M. Palaniswami et al. (Eds.), *Computational intelligence: a dynamic systems perspective* (pp. 152–163). New York: IEEE Press.

Anghinolfi, D., Boccalatte, A., Paolucci, M., & Vecchiola, C. (2008). Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In X. Li et al. (Eds.), *LNCS: Vol. 5361. SEAL* (pp. 411–420). Heidelberg: Springer.

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (2003). Concorde package. www.tsp.gatech.edu/concorde/downloads/codes/src/co031219.tgz.

Birattari, M. (2004). *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.

Birattari, M. (2009). *Tuning metaheuristics: a machine learning perspective*. *Studies of computational intelligence* (Vol. 197). Berlin: Springer.

Birattari, M., Zlochin, M., & Dorigo, M. (2006). Towards a theory of practice in metaheuristics design: A machine learning perspective. *Theoretical Informatics and Applications*, *40*(2), 353–369.

Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. Langdon et al. (Eds.), *GECCO 2002* (pp. 11–18). San Francisco: Morgan Kaufmann.

Clerc, M. (2006). *Particle swarm optimization*. London: ISTE.

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

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Eiben, A. E., Michalewicz, Z., Schoenauer, M., & Smith, J. E. (2007). Parameter control in evolutionary algorithms. In F. G. Lobo et al. (Eds.), *Studies in computational intelligence: Vol. 54. Parameter setting in evolutionary algorithms* (pp. 19–46). Berlin: Springer.

Fialho, A. (2010). *Adaptive operator selection for optimization*. PhD thesis, Université Paris-Sud XI, Orsay, France.

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

Hussin, M. S., & Stützle, T. (2010). *Tabu search vs. simulated annealing for solving large quadratic assignment instances* (Tech. Rep. TR/IRIDIA/2010-20). IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Johnson, D., McGeoch, L., Rego, C., & Glover, F. (2001). 8th DIMACS implementation challenge. http://www.research.att.com/~dsj/chtsp/.

Khichane, M., Albert, P., & Solnon, C. (2009). A reactive framework for ant colony optimization. In T. Stützle (Ed.), *LNCS: Vol. 5851. Learning and intelligent optimizatioN (LION)* (pp. 119–133). Heidelberg: Springer.

Lawler, E. L. (1963). The quadratic assignment problem. *Management Science*, *9*, 586–599.

Lawler, E. L., Lenstra, J. K., Rinnooy, Kan A. H. G., & Shmoys, D. B. (1985). *The traveling salesman problem: a guided tour of combinatorial optimization*. New York: Wiley.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., & Birattari, M. (2011). *The* `irace` *package, iterated race for automatic algorithm configuration* (Tech. Rep. TR/IRIDIA/2011-04). IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

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

Pellegrini, P., Stützle, T., & Birattari, M. (2010a). Companion of a critical analysis of parameter adaptation in ant colony optimization. http://iridia.ulb.ac.be/supp/IridiaSupp2010-013/. IRIDIA Supplementary page, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Pellegrini, P., Stützle, T., & Birattari, M. (2010b). Off-line and on-line tuning: a study on $\mathcal{MAX}$–$\mathcal{MIN}$ ant system for TSP. In M. Dorigo et al. (Eds.), *LNCS: Vol. 6234. ANTS 2010: seventh international conference on swarm intelligence* (pp. 239–250). Heidelberg: Springer.

Randall, M. (2004). Near parameter free ant colony optimisation. In M. Dorigo et al. (Eds.), *LNCS: Vol. 3172. ANTS 2004: fourth international conference on ant colony optimization and swarm intelligence* (pp. 374–381). Heidelberg: Springer.

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

Stützle, T., & Hoos, H. H. (2000). *MAX–MIN* ant system. *Future Generations Computer Systems*, *16*(8), 889–914.

Stützle, T., López-Ibánez, M., Pellegrini, P., Maur, M., Montes de Oca, M. A., Birattari, M., & Dorigo, M. (2011, to appear). Parameter adaptation in ant colony optimization. In Y. Hamadi et al. (Eds.), *Autonomous search*. Berlin: Springer.

Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing*, *17*, 443–455.

Taillard, E. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science*, *3*, 87–105.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83.