Swarm Intelligence manuscript No.
(will be inserted by the editor)

Ant Colony Optimization on a Limited Budget of Evaluations

Leslie Pérez Cáceres · Manuel López-Ibáñez · Thomas Stützle

This is a pre-print version of the article: Pérez Cáceres L, López-Ibáñez M, Stützle T. Ant Colony Optimization on a Limited Budget of Evaluations. *Swarm Intelligence*, 2015, doi: 10.1007/s11721-015-0106-x

Abstract Ant Colony Optimization (ACO) is a successful method for solving difficult combinatorial optimization problems. Following Ant System, the first ACO algorithm, a large number of algorithmic variants have been developed that showed significantly better performance on a wide range of optimization problems. Typically, performance was measured according to the solution quality achieved for a given computation time limit, which usually allowed the evaluation of a very large number of candidate solutions, often in the range of millions. However, there are practical applications where the number of evaluations that can be done is very restricted due to tight real-time constraints or to the high computational cost of evaluating a solution. Since these situations are quite different from those for which ACO algorithms were initially designed, current knowledge on good parameter settings or the most promising search strategies may not be directly applicable. In this paper, we examine the performance of different ACO algorithms under a strongly limited budget of 1,000 evaluations. We do so using default parameter settings from the literature and parameter settings tuned for the limited-budget scenario. In addition, we compare the performance of the ACO algorithms to algorithms that make use of surrogate modeling of the search landscapes. We show that tuning algorithms for the limited-budget case is of uttermost importance, that direct search through the ACO algorithms keeps an edge over techniques using surrogate modeling, and that the ACO variants proposed as improvements over Ant System remain preferable.

1 Introduction

Ant Colony Optimization (ACO) is a well-established metaheuristic that was invented in the beginning of the 1990s (Dorigo, 1992; Dorigo et al, 1991), inspired by real ants' pheromone trail laying and following behavior. After the proposal of

L. Pérez Cáceres · M. López-Ibáñez · T. Stützle
IRIDIA, CoDE, Université libre de Bruxelles, Brussels, Belgium
E-mail: {leslie.perez.caceres, manuel.lopez-ibanez, stuetzle}@ulb.ac.be

Ant System (AS), the first ACO algorithm, a number of other ACO algorithms showing improved performance over AS have been proposed. The main algorithmic improvements included Ant Colony System (Dorigo and Gambardella, 1997), Max-Min Ant System (Stützle and Hoos, 1997, 2000), Rank-based Ant System (Bullnheimer et al, 1999) and a few others (see Dorigo and Stützle (2004) for a review).

The development of ACO algorithms has usually focused on situations in which the computation time that is available allows for the generation of a large number of candidate solutions to be evaluated. A typical example in this context is the "First International Contest on Evolutionary Optimisation" (Bersini et al, 1996), where the competing algorithms were allowed to evaluate up to $10{,}000 \cdot n$ candidate solutions for the symmetric Traveling Salesman Problem (TSP), where n is the number of cities. This situation is common in the vast majority of the ACO literature. Often, the candidate solutions generated by the ants are improved by means of local search algorithms (Dorigo and Gambardella, 1997; Gambardella et al, 2012; Stützle and Hoos, 1997), which results in many additional solutions that need to be evaluated.

However, there are many situations in which an algorithm may evaluate only very few solutions. One such situation arises if the algorithm operates in a setting where tight real-time constraints exist such as in robotics applications: Even if the evaluation of candidate solutions is very fast, the computation time may be too restricted to allow the evaluation of many solutions. Another such situation arises in applications where the evaluation of a candidate solution requires intensive computation such that, within feasible computation times, only a very small number of candidate solutions can be evaluated. The latter is a typical setting that arises in the area of simulation-optimization (April et al, 2003; López-Ibáñez et al, 2008; Teixeira et al, 2012; Zeng and Yang, 2009), but also in industrial settings, when the computation of the cost function requires significant computational effort (Fernandez et al, 2014).

If only few candidate solutions can be evaluated, a first research question is how to transfer the knowledge available in the ACO literature to this new situation. This is an interesting question, as most ACO research has considered evaluation budgets where the number of solutions evaluated is in the order of tens of thousands or more. In this paper, we consider the evaluation budgets to be much smaller. Concretely, we consider a budget of 1,000 evaluations in most of the experiments. Obviously, one may wonder what would happen for even more restricted budgets (say, 100 evaluations) or slightly larger budgets (say, 10,000 evaluations). However, as we will see, the setting we examine is enough to show that our initial research question leads to interesting new insights. We remark that similar questions have been posed in different contexts on single ACO algorithms such as MAX-MIN Ant System (Pellegrini et al, 2006) but using much higher evaluation budgets than the ones we consider here. In addition, here we examine the question of which of the algorithmic ACO variants are the most performing in very low-budget cases. In a preliminary conference version of this research (Pérez Cáceres et al, 2014), we already explored ACO algorithms in reduced budget conditions. The experiments reported there showed that the default parameter settings of popular ACO algorithms are not appropriate for reduced budget conditions. Here, we confirm and extend those earlier computational results presenting additional experimental results.

In this paper, we also explore the possibility of using surrogate modeling approaches to help the search for high-quality solutions in low-budget cases (Jones et al, 1998; Knowles et al, 2009). These methods iteratively build a model of the solution landscape, which is less expensive to evaluate and can be searched to find promising solutions. This approach is widely used in continuous optimization in the form of, for example, the efficient global optimization (EGO) approach (Jones et al, 1998). Recently, both Zaefferer et al (2014b) and Moraglio and Kattan (2011) have proposed adaptations of the EGO method to tackle combinatorial problems. In this paper, we combine the best performing of these adaptations of EGO (Zaefferer et al, 2014b) with various ACO algorithms for optimizing the surrogate model. We then compare the performance of some of the main ACO algorithms —Ant System (AS), Elitist Ant System (EAS), Rank-based Ant System (RAS), MAX-MIN Ant System (MMAS), and Ant Colony System (ACS)— when applied directly to the problem and when used for optimizing the surrogate model of EGO. In both these cases, we compare the default ACO parameter settings recommended in the literature (Dorigo and Stützle, 2004) and the parameters settings obtained after tuning with irace (López-Ibáñez et al, 2011), an automatic algorithm configuration tool.

As benchmark problems, we use the TSP and the Quadratic Assignment Problem (QAP), which are the main benchmark problems that have been used in the early research on ACO (Bullnheimer et al, 1999; Dorigo and Gambardella, 1997; Dorigo et al, 1996; Stützle and Hoos, 2000). However, here we consider them under the constraint that at most 1,000 candidate solutions can be evaluated during an algorithm run.

The article is organized as follows. In Section 2, we present the algorithm details and the experimental setup. Section 3 presents detailed experimental results. Finally, we summarize the main findings of our work in Section 4.

2 Experimental Setting

2.1 Problems

The main reasons for choosing the TSP and the QAP as our test problems are that both problems are standard problems on which many ACO algorithms have been tested (Dorigo and Stützle, 2004), that they are very difficult to solve (NP-hard), that they have different structure w.r.t. their landscape properties (Stützle and Hoos, 2000), and that they are classical combinatorial problems representative of many other routing or assignment problems. In addition, the solution evaluation in both problems is very fast in practice: $O(n)$ and $O(n^2)$ for the TSP and the QAP, respectively. This fast evaluation allows us to perform comprehensive experiments within a feasible amount of time, and in particular, it allows us to consider the tuning of the involved ACO algorithms by automatic algorithm configuration tools. The latter is relevant, as it gives a hint on how parameter settings should be chosen if the evaluation budget is small and can, thus, provide more general guidelines.

For both problems, we have generated a set of benchmark instances to be used in the evaluation of the algorithms. In the TSP case, we used Random Uniform Euclidean (RUE) instances. These are generated by first placing points uniformly at random in a square of dimension 10,000 $\times$ 10,000 and then computing the

distances between the points (rounded to the closest integer). We have generated 55 instances for each number of cities $n \in \{50, 60, 70, 80, 90, 100\}$. 50 instances of each size are used for tuning, while the others are used as test set. For the QAP, the instances used by Pellegrini et al (2014) are taken. These instances have a structure similar to that of instances occurring in practical applications. We have used 60 instances for each size $n \in \{60, 80, 100\}$, using 50 of each size for tuning and 10 of each size as test set.

2.2 ACO Algorithms

Our study comprises five of the best-known ACO algorithms: AS (Dorigo et al, 1996), EAS (Dorigo et al, 1996), RAS (Bullnheimer et al, 1999), MMAS (Stützle and Hoos, 2000) and ACS (Dorigo and Gambardella, 1997). A detailed description of the above algorithms is given by Dorigo and Stützle (2004), as well as in the original articles; therefore, we only give a concise description of the main algorithm features so that all the parameters that are later tuned are explained.

Each ant in an ACO algorithm can be seen as a probabilistic construction procedure, which makes usage of one or two types of numerical information during the construction process. The first is the pheromone trail information, which is a numerical value associated with construction decisions that is adapted during the ACO algorithm run to bias the search toward high-quality solutions. The second is the heuristic information related to specific construction decisions. Heuristic information is available (and useful) for many combinatorial problems, and thus, it helps to improve the quality of the solutions generated by ACO algorithms; however, if no heuristic information is available, an ACO algorithm may proceed without it.

Many ACO algorithms use the random-proportional rule introduced with AS. This rule chooses at each decision point i the next feasible element j to be added to an ant's partial solution s_p with a probability given by

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N(s_p)^n} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} \quad \text{if } j \in N(s_p), \tag{1}$$

where $N(s_p)$ is the set of elements j that can extend a partial solution s_p maintaining feasibility of the partial solution; τ_{ij} is the pheromone trail associated with the decision, η_{ij} is the related heuristic information, and α and β are two parameters that regulate the influence of pheromones and heuristic information.

ACS constructs solutions using a rule that allows for a more deterministic choice: With a probability q_0, the element j is chosen as the one that maximizes the term $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$ (ties being broken randomly), while with a probability $1 - q_0$ a random choice following Equation 1 is made.

In this article, we use two minor variations of the ACO algorithms compared to how they were defined in the original papers. The first variation is that all ACO algorithms may use the action choice rule of ACS, that is, make a deterministic choice with probability q_0. The default version of these algorithms is simply obtained by setting $q_0 = 0$. The second is that we allow ACS to use a setting of $\alpha \neq 1$ (the parameter α does not appear in the original version of ACS). The original version of ACS is obtained by setting $\alpha = 1$.

Once the m ants have terminated their solution construction, where m is a parameter that is also called colony size, pheromone update takes place. AS and its direct variants EAS, RAS, and MMAS do so by first reducing the amount of pheromone (a process called pheromone evaporation) by a fixed factor ρ, where ρ is a parameter ($0 \leq \rho < 1$), and then depositing pheromone of an amount that is inversely proportional to the solution cost. This update can be written as

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij}^{alg} \tag{2}$$

where $\Delta\tau_{ij}^{alg}$ is the amount of pheromone deposited in an algorithm-specific way for each of the algorithms $alg \in \{\text{AS}, \text{EAS}, \text{RAS}, \text{MMAS}\}$. In all these four algorithms, the amount of pheromone deposited by a solution s_k considered in the pheromone update is equal to $1/L_k$, where L_k is the solution quality of s_k. In AS, each ant that has generated a solution in the current algorithm iteration deposits its corresponding amount of pheromone on the edges that it has traversed. In EAS, in addition to the deposit of AS, the best solution found since the beginning of the algorithm run, s_{gb} (for global-best solution), deposits an amount of pheromone equal to e/L_{gb}, where e is an integer parameter giving the weight of s_{gb}. In RAS, the best $r - 1$ solutions of each iteration are assigned weights decreasing from $r - 1$ (for the best ranked solution) to one (for the solution ranking $r - 1$) and then deposit an amount of pheromone proportional to their rank; additionally, also solution s_{gb} deposits pheromone using a weight r, a parameter of RAS. In MMAS, only one solution is chosen for pheromone deposit after each iteration; this may be the iteration-best, the global-best, or the restart-best solution.

ACS uses a different pheromone deposition rule; it only deposits pheromone on the global-best solution using the formula

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_{ij}^{\text{gb}}.$$

In addition, in ACS, each ant applies immediately after each construction step the so-called local pheromone update rule, where we have

$$\tau_{ij} = (1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0;$$

τ_0 is a very small constant. This rule has the effect of removing pheromone at each of the construction steps, and thus, it favors exploration of the search.

In contrast to the pheromone update rule of AS, the other ACO algorithms use more explicitly the best solutions found (iteration-best, global-best, or restart-best) during the search, which favors a stronger convergence. However, depending on the used parameter settings, this stronger convergence may happen only after a rather long computation time (Dorigo and Stützle, 2004).

Here, we use the implementation of the ACO algorithms as provided by the ACOTSP software (Stützle, 2002). We neither use candidate sets nor local search. The former is excluded to avoid usage of problem-specific knowledge that may not be available in expensive optimization problems; the latter is excluded as the small number of evaluations does not even allow to complete a single neighborhood scan for the problem sizes we consider. As ACOTSP provides only the code for tackling the TSP, we have adapted the software also to the QAP so that we can use the same implementation of the ACO algorithms for both. However, in the QAP case, we have not considered the usage of heuristic information. In the TSP case, the exclusion of heuristic information is simply handled by setting $\beta = 0$.

2.3 Efficient Global Optimization

When the evaluation of solutions is very costly, the use of surrogate models in the optimization process may prove beneficial (Jones et al, 1998; Knowles et al, 2009). This approach fits a model $\widehat{M}$ of the search landscape using an available set of candidate solutions together with their computed objective function values. The approach then uses the model as an inexpensive estimation (inexpensive w.r.t. the exact evaluation of a candidate solution) of the objective function and computes estimates $\widehat{M}(s_i)$ for each candidate solutions s_i. Only the best (or a subset of the best) candidate solutions as judged by the model are then actually evaluated using the exact objective function. This evaluation, in turn, is used to update the model $\widehat{M}$, refining it in this way and the process then iterates.

This approach has become rather popular for continuous optimization problems; a prominent example is the Efficient Global Optimization (EGO) algorithm by Jones et al (1998). This algorithm iteratively fits a stochastic process model (Kriging) to approximate the solution space. This is done by estimating the parameters of the model using maximum likelihood estimation. Initially, the model is created using a limited set of data points. At each iteration, EGO searches for a solution that maximizes the Expected Improvement (EI), which is a measure that takes into account the mean prediction for a solution and the uncertainty about the estimation. Maximizing EI instead of the model prediction gives EGO a way to balance exploration and exploitation, avoiding a too fast convergence of the algorithm.

Recently, these surrogate modeling approaches have been applied to the solution of combinatorial optimization problems (Moraglio and Kattan, 2011) and of permutation problems (Moraglio et al, 2011; Zaefferer et al, 2014a,b). This is done by using an appropriate definition of a distance measure between solutions in order to calculate the correlation between data points. The distance measure must be selected according to the problem at hand; examples of distance measures and correlations between these measures can be found in (Schiavinotto and Stützle, 2007; Zaefferer et al, 2014a). In our experiments, we use the bi-directional adjacency distance when solving the TSP. This measure defines the distance between two TSP tours as n minus the number of times two cities are direct neighbors in both tours. When solving the QAP, we use a generalized Hamming distance, which defines the distance between two QAP solutions as n minus the number of positions in the two permutations that are the same. Otherwise, our implementation of the EGO approach follows the Kriging model that was found to be the best performing implementation of the EGO principle by Zaefferer et al (2014b). Our implementation is written in `C` in order to speed up computation.

When the number of data points increases, the computation of the model becomes increasingly expensive. We therefore limited the number of solutions that are used to compute the model $\widehat{M}$ to a maximum of 300. When the number of available solutions becomes larger than 300, we eliminate the solution with the smallest average distance from the others in the set of data points. The generation of the solutions that are to be evaluated by the model $\widehat{M}$ is done in our case by the same ACO algorithms that we apply in our analysis. As a result, we have five variants of the EGO algorithms according to which of the ACO algorithms (AS, EAS, RAS, MMAS, or ACS) is chosen to generate candidate solutions. We refer to the five resulting algorithms as EGO-AS, EGO-EAS, etc.; if we want to

refer generically to all of them, we use EGO-ACO. Finally, note that candidate solutions can be generated in a straightforward way by the ACO algorithms even when using the model $\widehat{M}$ for evaluating them: The main necessary change is to replace the direct evaluation of a candidate solution through the objective function by the estimation through the model $\widehat{M}$.

2.4 Automatic Configuration

The comparison of ACO and EGO algorithms is done in various stages, where we first consider default parameter settings and then parameter settings tuned for the scenario with strongly limited evaluation budget. The comparison using default parameter settings is interesting because algorithms are often applied using parameter settings known from the literature even when tackling different scenarios. Parameter tuning reveals that, as maybe expected, this way of proceeding may result in poor performance for several of the ACO algorithms. The main reason is that default parameter settings were usually devised considering scenarios where a large number of solution evaluations can be done. For tuning, we apply the `irace` software (López-Ibáñez et al, 2011) that implements Iterated F-race and other racing methods (Balaprakash et al, 2007). The scenarios we considered for tuning are the following.

tune-ACOTSP, tune-ACOQAP: These scenarios tune the ACO algorithms using a maximum budget for the run of each algorithm of 1,000 solution evaluations. These scenarios require the tuning of four parameters (α, β, m, q_0) common to all ACO algorithms for the QAP and the TSP, plus the tuning of β in the TSP case where the usage of β is considered, plus the tuning of algorithm-specific parameters for EAS (parameter e) and RAS (parameter r). The parameters and their ranges are given in Table 1. The budget considered for each tuning is 10,000 runs of the corresponding ACO algorithm. Note that this budget refers to how many times during a tuning run an algorithm may be executed; each algorithm is executed for 1,000 evaluations. For each benchmark problem, the first 50 instances of each size n (see Sec. 2.1) are used as a training set for the tuning.

tune-EGOTSP, tune-EGOQAP: These scenarios tune the settings of the ACO algorithms when they are used inside EGO. This tuning is only done when EGO is allowed a maximum of 100 evaluations instead of the default 1,000. The reason for this is twofold. First, a single execution of EGO for a total of 1,000 evaluations requires about three hours of CPU time on our hardware.[1] Second, our initial experiments with EGO (reported in the next section) show that its use is beneficial only when a very small number of evaluations is allowed. The total configuration budget for each tuning run was 5,000 runs of the EGO algorithm using as training set the same instances as in the scenarios tune-ACOTSP and tune-ACOQAP.

[1] All experiments were performed on a single core of cluster nodes each equipped with two AMD Opteron 6272 16 cores CPUs running at 2.1 GHz and with 64 GB RAM. Note that the heavy computation needs for the model update, and other computations in EGO make this approach an option in case the actual real evaluation of a candidate solution involves very high computation times but not for the case of very tight real-time constraints.

Table 1 Range of possible parameter values considered in the tuning of the parameter settings.

Common parameters for all scenarios				RAS	EAS	TSP scenario
m	α	ρ	q_0	r	e	β
$[5, 100]$	$[0, 10]$	$[0.01, 1]$	$[0, 1]$	$[1, 100]$	$[1, 750]$	$[0, 10]$

3 Experimental Results

In this section, we study the performance of the considered ACO and EGO-ACO algorithms when solving TSP and QAP instances. All analyses will be done considering default and tuned parameter settings of the algorithms. We first compare the five ACO algorithms in Section 3.1 and next the EGO-ACO algorithms in Section 3.2. The results of this latter comparison suggested to have a closer look at even smaller budgets than the default 1,000 evaluations we considered, which is done in Section 3.3. We further examine in Section 3.4 whether the availability of heuristic information would change our initial conclusions and explore the range of parameter settings that are obtained in repeated tuning runs. Finally, in Section 3.6, we consider cross-benchmark comparisons where parameter settings tuned on one problem are applied to the other and vice versa.

In the remainder of the paper, statistical significance tests refer to Wilcoxon signed-rank tests at the 0.05 significance level with Bonferroni correction for multiple testing and pairing on the instances. For the experiments, we use one single run for each of 30 test instances for the TSP and the QAP.

3.1 Ant colony optimization algorithms

3.1.1 Default parameter settings

The first comparison uses the ACO algorithms with their default parameter settings as recommended in the relevant literature (see Dorigo and Stützle, 2004) or the ACOTSP code (Stützle, 2002) and uses an evaluation budget of 1,000. The experiments reported in this section do not make use of heuristic information: For ACOQAP, no heuristic information was implemented, while for ACOTSP we set $\beta = 0$. The reason to exclude heuristic information is to simulate a real-world situation where such kind of information may not be available or be too expensive to compute. The results are presented using AS as a baseline; in particular, we computed, for each instance separately, the relative percentage deviation of the solutions' quality obtained by each ACO algorithm when compared to the one obtained by AS. (By definition, the percentage deviation from AS to itself is zero.) The results are then presented as boxplots showing the distribution of these percentage deviations across the 30 test instances.

Figure 1 gives the resulting plots. A value larger (smaller) than zero indicates worse (better) performance than AS. The figure shows that the performance of the ACO algorithms relative to AS is similar in the TSP and the QAP cases. In both cases, the results obtained by MMAS are not statistically different from the AS ones, while ACS obtains the best results followed by EAS; RAS obtains statistically significantly better results than AS only on the TSP.

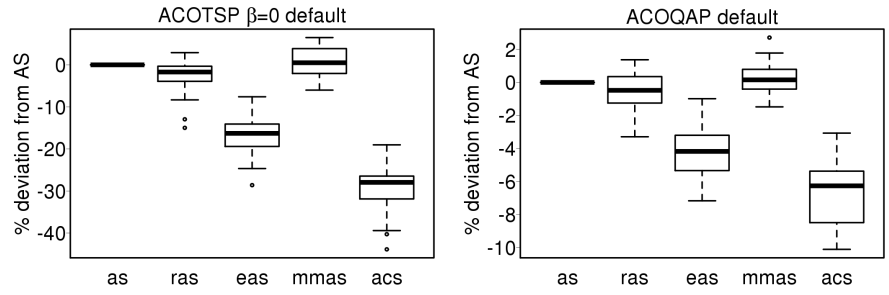

Fig. 1 Boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms taking AS as reference, using their default parameter settings. The line for AS is by definition the zero-line. Positive values indicate that AS obtained a better solution. The results are given across all test instances.

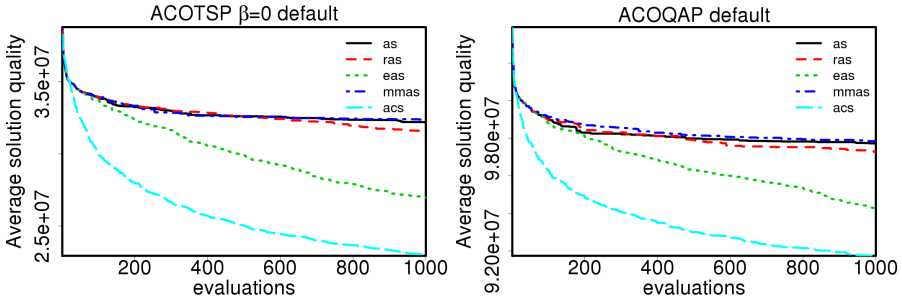

Fig. 2 Average solution quality as a function of the number of solution evaluations. Compared are the five ACO algorithms for ACOTSP (left) and ACOQAP (right) using default parameter settings. The average is computed across all test instances.

Figure 2 shows the average solution quality as a function of the number of evaluations across the tested instances for the five ACO algorithms. These curves show that ACS and EAS intensify the search more effectively than the other algorithms.

3.1.2 Tuned parameter settings

Tuning the parameters of an algorithm may significantly improve it. The tuning of the ACO algorithms parameters using the `irace` software confirms this fact. Figure 3 shows the improvement of each ACO algorithm comparing the results obtained with parameter settings from the tune-ACOTSP and tune-ACOQAP scenarios to default parameter settings. Negative values indicate improved quality. In all cases except AS, the improvement is statistically significant and in many cases substantial. MMAS and RAS benefit clearly the most from the tuning.

The performance improvement through tuning has also a direct impact on the relative ranking of the ACO algorithms. Figure 4 shows the percentage deviation of the results obtained by the tuned ACO algorithms w.r.t. those obtained by AS with tuned parameter settings, which is used as a baseline. For both problems, EAS, RAS, MMAS, and ACS improve significantly over the performance of AS—all differences being so large that no statistical test is needed. For the TSP, the

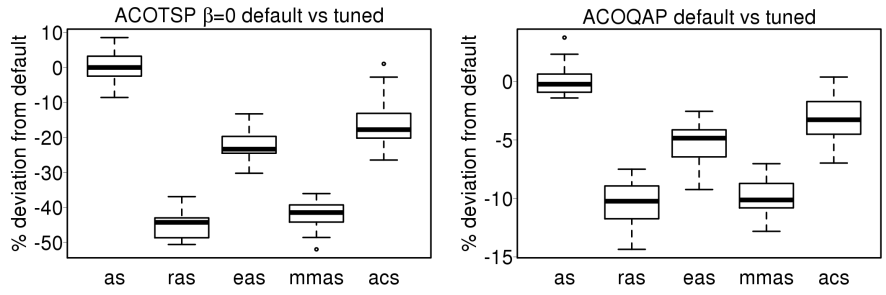

Fig. 3 Boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms with tuned parameter settings taking as reference the results obtained by the respective ACO algorithms with default parameter settings. As an example, the boxplot of RAS indicates the difference between RAS using tuned parameter settings and using its default parameter settings. Positive values indicate that the default version obtained a better solution. The results are given across all test instances.

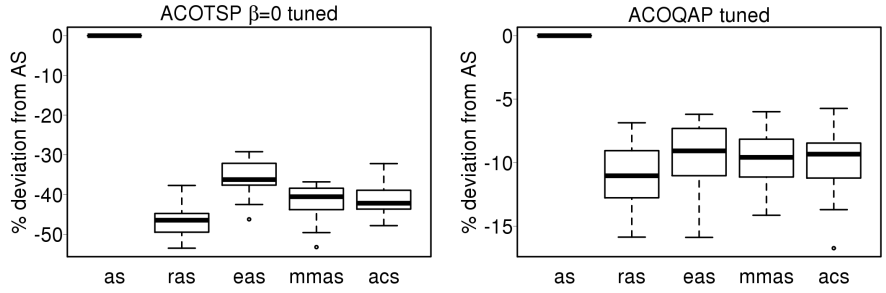

Fig. 4 Boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms taking AS as reference, using their tuned parameter settings. The line for AS is by definition the zero-line. Positive values indicate that AS obtained a better solution. The results are given across all test instances.

improvement is substantial, being in the range of 30% to more than 50%. The best performance among the algorithms is obtained by RAS, which is statistically significantly better than the other algorithms for both the TSP and the QAP.

The default and tuned parameter settings for each of the ACO algorithms on the TSP and the QAP are shown in Table 2. Repeating the tuning procedure results in similar parameter settings found (see Section 3.5). For each of the five algorithms, there are settings that differ strongly from the recommended default settings. For example, for MMAS, the largest differences are for ρ and m, while for ACS, these are for the parameters q_0 and α. These differences arise mainly from the fact that the default ACO algorithm settings were designed for a scenario where ample computation time is available. The tuned parameter settings, especially for RAS and MMAS, imply a much stronger exploitation of the best solutions generated: The high evaporation rate quickly shifts the search focus around the best solutions found, and the smaller number of ants allows performing more iterations so that appropriate pheromone distributions reflecting the best found solutions can be established. Maybe surprising is the low setting of q_0 for ACS, but this is compensated by the very high setting of α, which biases the choice

Table 2 Tuned and default parameter settings of the five ACO algorithms for the TSP and QAP scenarios. The values are given as tuned ACOTSP / tuned ACOQAP / default (the same default is used for both problems, TSP and QAP). The '*specific*' column corresponds to the parameters specific to EAS (e) and RAS (r).

	m	α	ρ	q_0	*specific*
AS	44 / 71 / n	1.08 / 0.53 / 1	0.44 / 0.32 / 0.5	0.07 / 0.53 / 0.0	–
EAS	8 / 9 / n	0.76 / 0.51 / 1	0.32 / 0.34 / 0.5	0.05 / 0.68 / 0.0	221 / 148 / n
RAS	35 / 32 / n	1.31 / 0.78 / 1	0.72 / 0.74 / 0.1	0.16 / 0.36 / 0.0	10 / 8 / 6
MMAS	9 / 14 / n	1.51 / 1.07 / 1	0.23 / 0.60 / 0.02	0.00 / 0.36 / 0.0	–
ACS	10 / 5 / 10	6.87 / 2.24 / 1	0.08 / 0.41 / 0.1	0.25 / 0.03 / 0.9	–

strongly to the decision associated with the highest pheromone level. We provide a more detailed analysis of the parameter settings in Section 3.5.

3.2 Efficient Global Optimization

In this section, we analyze the performance of the EGO variants using first ACO default parameter settings and next ACO tuned parameter settings.

3.2.1 Default EGO-ACO parameter settings

We first compare the results obtained by EGO using each of the five ACO algorithms for optimizing the Kriging model; this results in five algorithms that are in the following identified as EGO-AS, EGO-EAS, etc., depending on which ACO algorithm is used to optimize the Kriging model. Figure 5 shows the resulting percentage deviations using as reference the results obtained by EGO-AS. Except for EGOTSP-ACS[2], none of the algorithms show statistically significant differences for the TSP or the QAP. Hence, one may conjecture that the choice of the particular ACO algorithm for optimizing the Kriging model has no strong impact on the final results. Even more surprisingly, EGO-ACS with default settings gives somewhat worse results than the others, even though ACS was clearly the best performing ACO algorithm when applied directly to the problem (see Fig. 1).

A first question is whether the usage of surrogate modeling improves over the direct application of ACO algorithms to the problem. The results of this comparison are given in Fig. 6. Although the performance of EGOTSP-AS, EGOTSP-RAS and EGOTSP-MMAS is statistically significantly better compared to AS, RAS, and MMAS, respectively, the difference is not large in absolute terms. Moreover, in the cases of EAS and ACS, the usage of surrogate modeling results in significantly worse performance. If we take into account that the tuned ACO algorithms reached, with the exception of AS, much better performance than their default versions, it is clear that the EGO variants are not competitive to tuned ACO algorithms under the settings tested here.

[2] In what follows, we use the problem name to identify results specific to a problem, e.g. EGOTSP-AS refers to the TSP results obtained by using AS to search the landscape of the expected improvement.

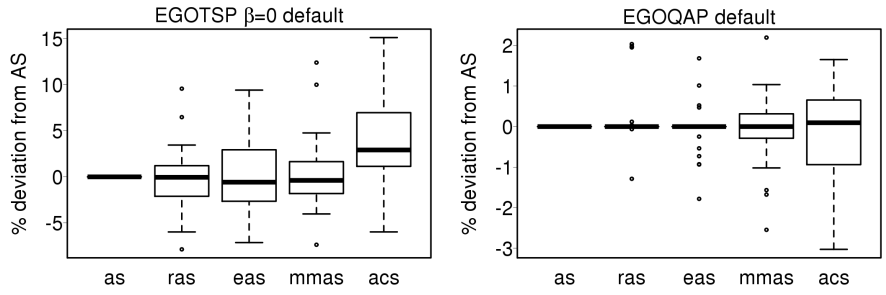

Fig. 5 Boxplots of the percentage deviation of the solution quality obtained by the EGO-ACO algorithms taking EGO-AS as reference, using their default parameter settings. The line for EGO-AS is by definition the zero-line. Positive values indicate that EGO-AS obtained a better solution. The results are given across all test instances.

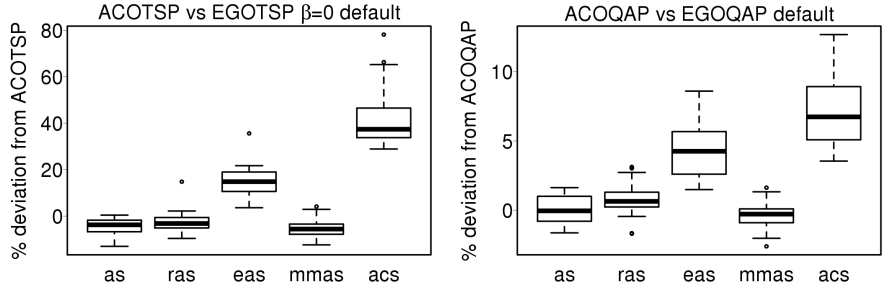

Fig. 6 Boxplots of the percentage deviation of the solution quality obtained by the EGO-ACO algorithms with default parameter settings taking as reference the results obtained by the respective ACO algorithms with default parameter settings. As an example, the boxplot of RAS indicates the difference between EGO-RAS and RAS using default parameter settings. Positive values indicate that the ACO algorithm obtained a better solution. The results are given across all test instances.

3.2.2 Tuned EGO-ACO parameter settings

The EGO model implicitly has a diversifying effect as it tends to explore unseen parts of the search space. Hence, the culprit of the relatively poor performance of the EGO-ACO algorithms we observed may be that the default ACO parameter settings lead to a too strongly exploratory behavior. To test this conjecture, we applied the parameter settings of the ACO algorithms tuned for 1,000 evaluations from Table 2 to EGO-ACO; these parameter settings favor intensification of the search, as shown by the strongly improved performance shown in Section 3.1.2. The results of these experiments are shown in Figure 7, where we compare the solution quality of the default and tuned settings of the ACO algorithms. For EGOTSP-MMAS, EGOTSP-RAS, and EGOQAP-EAS statistically significantly improved performance is obtained; in the other cases, the results change slightly but not significantly. However, this improvement is not enough to make the EGO-ACO algorithms competitive to the tuned parameter settings of ACO, as it is shown in Figure 8. Thus, once adequate parameter settings are given for the ACO algorithms, better or equivalent results are obtained applying them directly to the

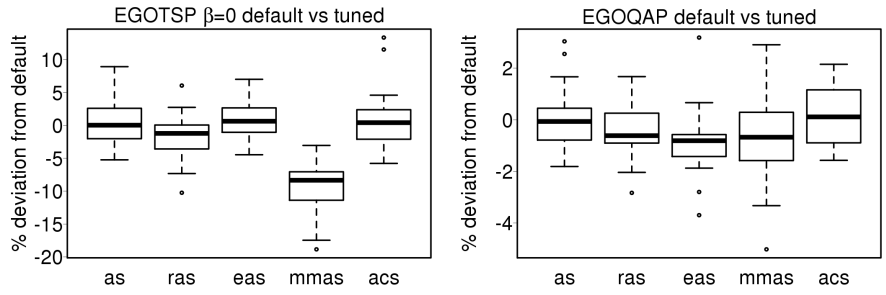

Fig. 7 Boxplots of the percentage deviation of the solution quality obtained by the EGO-ACO algorithms with tuned parameter settings taking as reference the results obtained by the respective EGO-ACO algorithms with default parameter settings. As an example, the boxplot of RAS indicates the difference between EGO-RAS using tuned parameter settings and using its default parameter settings. Positive values indicate that the default version obtained a better solution. The results are given across all test instances.

problem without using surrogate modeling. The only exception is EGOTSP-AS, which results in statistically significantly better quality than AS.

If we compare the development of the solution quality over the number of solutions generated, we can observe that the ACO algorithms obtain much better quality solutions across the largest range of the possible number of evaluations. Only for very few evaluations, the EGO algorithms may compete with the ACO algorithms, which is most visible for the case of EGOTSP-MMAS in Fig. 9 [analogous plots for the other ACO algorithms are available in the supplementary material (Pérez Cáceres et al, 2015)].

3.3 Strongly reduced budget case

Surrogate models are usually used when very few evaluations can be made, and as shown, there may be situations where they can become competitive to ACO algorithms. Here, we examine whether further restrictions of the evaluation budget may reverse the conclusions we have obtained so far. In particular, here we consider a maximum of 100 candidate solution evaluations. This also allows us to directly tune the ACO parameters of the EGO-ACO algorithms (see scenarios tune-EGOTSP and tune-EGOQAP in Sec. 2.4) instead of applying the tuned settings found in the scenarios tune-ACOTSP and tune-ACOQAP to EGO-ACO. The scenarios tune-EGOTSP and tune-EGOQAP should be better to judge EGO performance, as best parameter settings for the ACO algorithms to explore the surrogate model surface may be very different from the ones that are necessary to reach high-quality solutions by ACO alone. However, similarly to what was reported in the previous section, the tuning did not lead to strongly improved EGO-ACO performance over using default parameter settings. For the TSP, tuned parameter settings improve statistically significant performance over the respective ACO algorithms only for EGOTSP-MMAS and EGOTSP-ACS and for the QAP only for the EGOQAP-RAS and EGOQAP-EAS algorithms. Boxplots of these results are given in the supplementary material (Pérez Cáceres et al, 2015).

To make a fair comparison, we also re-tuned the ACO algorithms for a budget of 100 evaluations. The parameter settings can be found in the supplementary

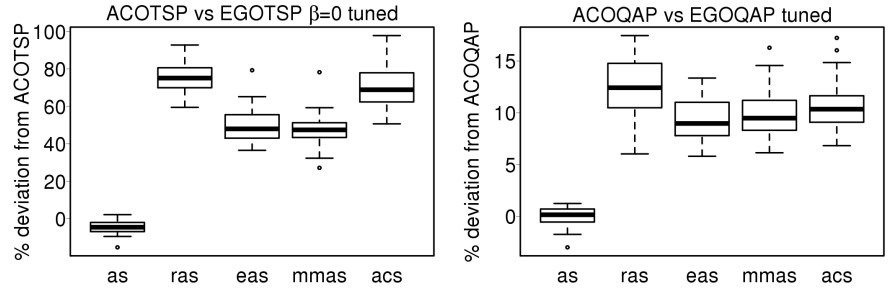

Fig. 8 Boxplots of the percentage deviation of the solution quality obtained by the EGO-ACO algorithms with tuned parameter settings taking as reference the results obtained by the respective ACO algorithms with tuned parameter settings. Positive values indicate that the ACO algorithm obtained a better solution. The results are given across all test instances.

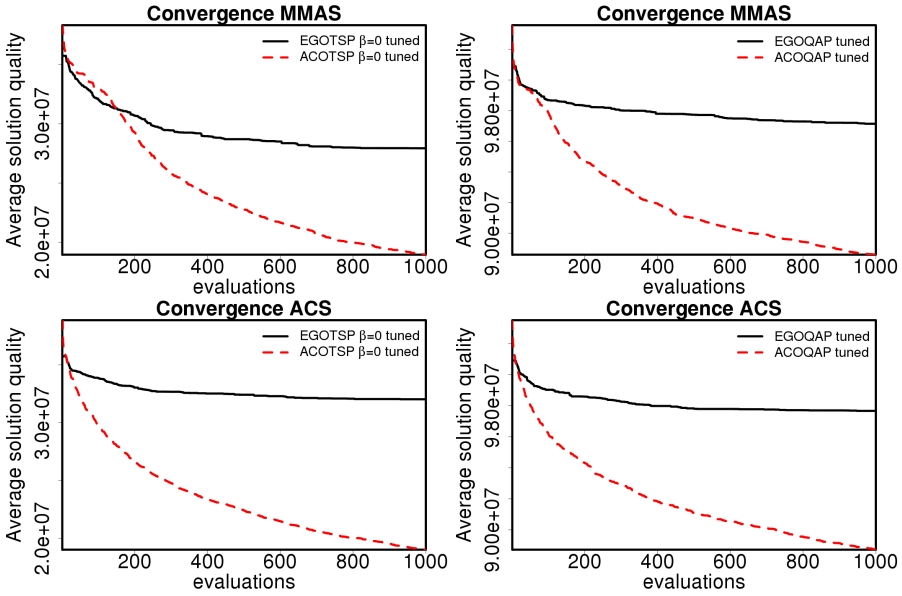

Fig. 9 Average solution quality as a function of the number of solution evaluations. Compared are the EGO-ACO algorithms versus the ACO algorithms for MMAS (top) and ACS (bottom) using tuned ACO parameter settings. The average is computed across all test instances.

material. Figure 10 gives the resulting percentage deviations of solution quality obtained by the EGO-ACO algorithms w.r.t. the results obtained by the ACO algorithms, both using parameter settings tuned for 100 evaluations. Except for AS and EGOTSP-RAS, applying ACO algorithms directly on the problems instead of searching the surrogate model improves performance in a statistically significant way. Hence, even in situations with very low overall number of evaluations, the ACO algorithms are preferred over the EGO-ACO ones. The best performance among the ACO algorithms is reached by EAS and ACS for the TSP and by RAS and EAS for the QAP, although the difference with respect to the other algorithms (except AS) is not large in absolute terms. [For details, see the supplementary material (Pérez Cáceres et al, 2015)].

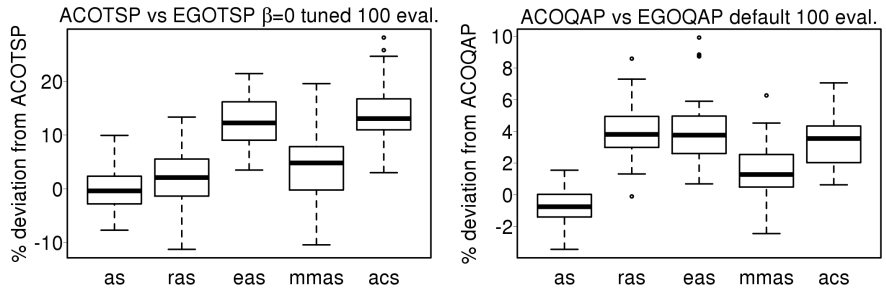

Fig. 10 Boxplots of the percentage deviation of the solution quality obtained after 100 evaluations by the EGO-ACO algorithms with parameter settings tuned for 100 evaluations taking as reference the results obtained by the respective ACO algorithms with parameter settings tuned for 100 evaluations. Positive values indicate that the ACO algorithm obtained a better solution. The results are given across all test instances.

3.4 Heuristic Information

Heuristic information may help to guide the search process and be exploited during the solution construction by the ants. Even in the case of expensive optimization problems, the algorithm designer may have knowledge about the problem that may allow her to derive such heuristic information; often, this will also be possible if one is in real-time scenarios that allow only for very few evaluations. In this section, we explore the relative performance of the ACO and EGO-ACO algorithms if such heuristic information is available. In particular, we use the example of the application to the TSP, where it is known that the heuristic information provides reliable guidance towards good-quality solutions.

3.4.1 ACO algorithms

As a first step, we compare the performance of default parameter settings of the ACO algorithms to that of AS. The only exception is that we set $\beta = 2$ for all ACO algorithms, since we know in advance that different settings of this parameter will strongly affect the results. Figure 11 (left plot) shows the percentage deviation of the solutions generated by the ACO algorithms from that of AS. Maybe surprisingly, RAS and MMAS exhibit much worse results than AS. This may be explained by the fact that both algorithms use a relatively low evaporation rate setting as default that does not allow them to converge towards high-quality solutions within the available computation budget. Differently, EAS and ACS improve over the AS results. This might be explained by the fact that their search is more exploitative already after the first evaluations. For EAS, this is due to the higher evaporation rate than in RAS and MMAS and to the strong pheromone deposit by the global-best solution when compared to AS. For ACS, this is mainly due to the large setting of the q_0 parameter that directly exploits the global-best solution. After tuning (Fig. 11, right plot) all parameters including β, RAS, EAS, MMAS and ACS all show better performance than AS. The differences between the algorithms are not statistically significant except for the difference between MMAS and EAS and the differences between all other algorithms and AS. The main reason that the conclusions change after tuning is, as in Section 3.1.2, that the various ACO

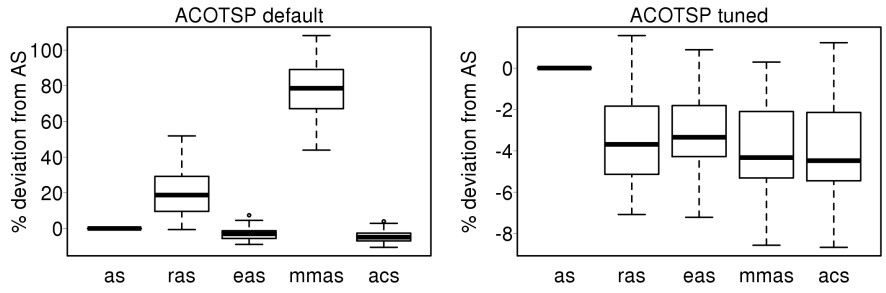

Fig. 11 Boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms taking AS as reference, using their default (left plot) and tuned (right plot) parameter settings. The line for AS is by definition the zero-line. All algorithms use heuristic information. Positive values indicate that AS obtained a better solution. The results are given across all test instances.

algorithms do not benefit equally from the tuning [see supplementary material for details (Pérez Cáceres et al, 2015)].

3.4.2 Efficient global optimization

Also in EGO-ACO the ants may use heuristic information during the solution construction. We therefore did the same analysis as in Section 3.2 but this time with heuristic information enabled. Figure 12 shows on the left plot the percentage deviation reached by the EGO-ACO algorithms using default parameter settings relative to EGO-AS; on the right side, it shows the percentage deviation from EGO-AS after using tuned parameter settings for the ACO algorithms. For default parameter settings, EGO-ACS is the only algorithm that gives a statistically significant improvement over EGO-AS, while EGO-EAS obtains slightly worse results than EGO-AS. Using tuned parameter settings for the ACO algorithms has a major impact on the performance of the EGO-ACO algorithms, and except for EGO-ACS, all other EGO-ACO algorithms profit strongly from it (see right plot in Fig. 12).

Finally, it is interesting to compare the results obtained with EGO-ACO to those obtained with the ACO algorithms applied directly to the problem. Figure 13 shows that with the exception of EGO-MMAS and MMAS, in all other cases, it is beneficial to apply the respective ACO algorithms directly to the combinatorial problem that is to be solved. Hence, given that among RAS, EAS, MMAS and ACS no clear winner arises when using tuned parameter settings and heuristic information, the experimental results would suggest to use any of these instead of EGO.

3.5 Parameter analysis

It is interesting to examine how the parameter settings differ between different tuning runs. To do so, we tuned each ACO algorithm 20 times with irace. The resulting distributions of the parameter settings that were found are given in Figure 14 for the case in which ACO algorithms are used without heuristic information

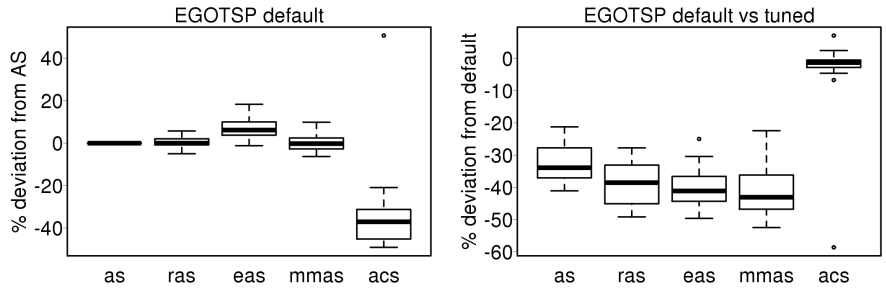

Fig. 12 Left: boxplots of the percentage deviation of solution quality obtained by the EGO-ACO algorithms taking EGO-AS as reference, using their default parameter settings. Positive values indicate that EGO-AS obtained a better solution. Right: boxplots of the percentage deviation of solution quality obtained by the EGO-ACO algorithms using tuned parameter settings from the respective results using default parameter settings. All algorithms use heuristic information. Positive values indicate that the default version obtained a better solution. The results are given across all test instances.

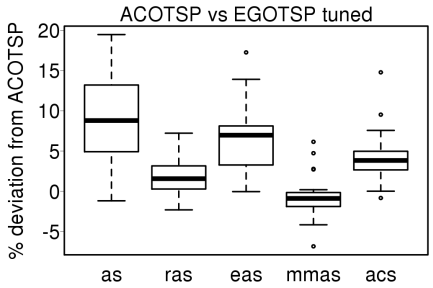

Fig. 13 Boxplots of the percentage deviation of the solution quality obtained by the EGO-ACO algorithms with tuned parameter settings taking as reference the results obtained by the respective ACO algorithms with tuned parameter settings. All algorithms use heuristic information. Positive values indicate that the ACO algorithm obtained a better solution. The results are given across all test instances.

and in Figure 15 for the case in which heuristic information is used. The dotted (red) line in each of the plots indicates the default parameter setting.

We consider first the case without heuristic information, that is, the results in Fig. 14. Although the parameter settings selected by irace vary from tuning to tuning, some clear trends can be observed. The smallest difference between default and tuned parameter settings is observed for AS, while for other algorithms, the differences can be much larger. For example, EAS and MMAS use much less ants (parameter m) than in the default parameter settings. This allows them to do also a larger number of iterations than is possible with the default setting of m. In addition, both algorithms and also RAS have a much larger settings for the pheromone evaporation rate ρ than by default. A high pheromone evaporation helps to focus the search quickly towards the best solutions that have been identified. Clearly, an explanation for these differences is the need to exploit aggressively the best solutions found so far due to the possibility of only evaluating very few solutions. This interpretation is supported by a setting of q_0 that is larger than zero for all ACO algorithms. Only in the case of ACS the distribution of the tuned values for

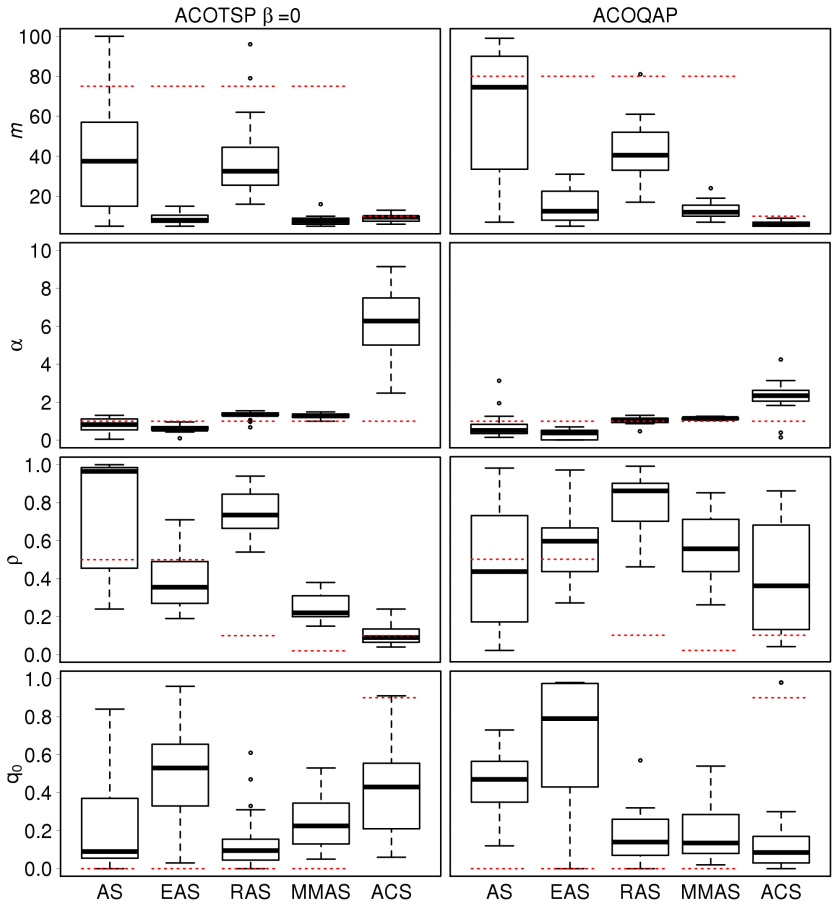

Fig. 14 Distribution of the parameter settings found in 20 runs of irace when ACO algorithms do not use heuristic information. The dotted (red) lines indicate the default parameter setting for each algorithm. The default parameter setting for the number of ants is the instance size; as in our test instances the size varies, we assumed an "average" instance size of 80 to indicate the default setting.

q_0 is maybe surprising as the tuned settings of q_0 are mostly much smaller than the default setting of $q_0 = 0.9$. However, in the case of ACS, this is made up by the rather high value for α that is obtained in the tuning. In a sense, the way ACS exploits the best solutions found so far is shifted from high settings to parameter q_0 to exploitation by high settings to parameter α.

The overall trends that arise for the tuned parameter settings in the ACOTSP and the ACOQAP scenarios are similar. Nevertheless, differences also arise in the best parameter settings for specific algorithms. In fact, in various cases, the boxplots for the parameter settings in the two scenarios for two same ACO algorithms do not overlap (see Fig. 15).

If we consider the tuned parameter settings in the case where heuristic information is available (see Fig. 15), the trends are mainly the same, though typically less pronounced. The main reason here is probably the fact that the heuristic in-

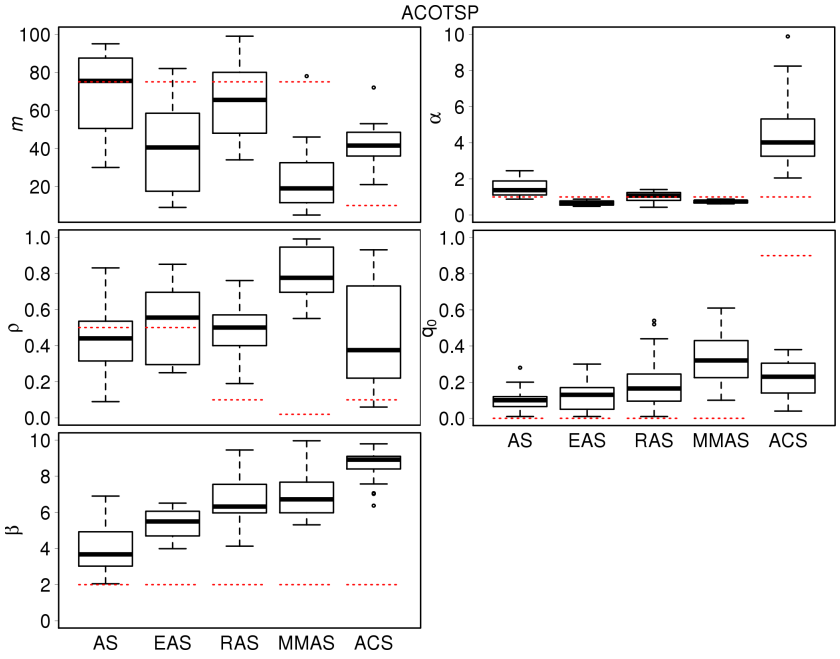

Fig. 15 Distribution of the parameter settings found in 20 runs of irace when ACO algorithms use heuristic information. The dotted (red) lines indicate the default parameter setting for each algorithm. The default parameter setting for the number of ants is the instance size; as in our test instances the size varies, we assumed an "average" instance size of 80 to indicate the default setting.

formation in the TSP case is relatively reliable in focusing the attention of the search on the most promising parts of the search space and, thus, requiring a less strong focus on exploitation through other parameters.

3.6 Cross-benchmark comparison

The main trend arising in the tuned parameter settings is a shift towards settings that imply a stronger exploitation of the search history and, in particular, of the best-so-far solution identified during the search. In our case here, tuning is feasible because we consider benchmark problems such as the TSP and the QAP for which solution evaluation is very quick in practice. If one would like to apply ACO algorithms to a problem where tuning becomes infeasible due to the high computational cost, one may still adopt parameter settings of ACO algorithms tuned for low-budget scenarios on less expensive benchmark problems. While one could argue that parameter settings tuned for low-budget scenarios in the TSP or QAP might not be useful for low-budget scenarios in other expensive problems, we conjecture that the importance of the low budget might be more crucial than the actual problem to be tackled. We examine this question by performing a cross-benchmark comparison of the tuned parameter settings. In other words, we compare default versus tuned-for-QAP parameter settings on the TSP and default vs. tuned-for-TSP parameter settings on the QAP. Figure 16 gives the correspond-

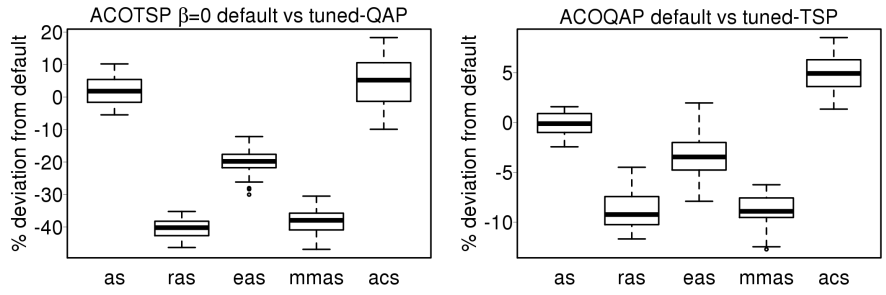

Fig. 16 Left: boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms with tuned-for-QAP parameter settings taking as reference the results obtained by the respective ACO algorithms with default parameter settings on the TSP. Right: boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms with tuned-for-TSP parameter settings taking as reference the results obtained by the respective ACO algorithms with default ACO parameter settings on the QAP. Positive values indicate that the default version obtained a better solution. The results are given across all test instances.

ing results. The plots indicate that RAS, EAS, and MMAS profit strongly from such a tuning even if done on a different problem. Differently, ACS results worsen slightly when compared to its default settings, while AS results are almost unaffected. This indicates that the former algorithms suffered in their default setting from a strong lack of exploitation in the low-budget case, while ACS in its default version already shows significant exploitation of the search history and does not benefit from tuning on a single other problem.

When we do the same comparison to explore how much may be gained by tuning on the right problem, we can observe in Fig. 17 that, as expected, some loss of performance arises. In the case of RAS, EAS, and MMAS this loss is relatively small when compared to the gains that may arise if replacing default parameter settings with more exploitative ones. Differently, the loss for ACS seems to be already quite significant and even be more than when compared to ACS default settings. In practice, one would consider re-tuning the various ACO algorithms across different problems with the goal of obtaining more robust settings that possibly generalize better than the ones tuned for a single combinatorial problem. This is, however, left for future research.

4 Conclusions and Future Work

In this paper, we have considered situations where the number of candidate solutions that can be evaluated is very limited. This can be due to either tight real-time constraints or due to very costly evaluation of candidate solutions as they arise, for example, in simulation optimization. We have analyzed the performance of five well-known ACO algorithms for such scenarios using as test problems the TSP and the QAP. In addition, we have used the ACO algorithms as the search algorithms within surrogate modeling approaches, in particular, the EGO approach.

Our findings can be summarized as follows. First, the tuning of the ACO algorithms is crucial: Default parameter settings have often been proposed using different application contexts where a much larger number of evaluations of candidate solutions is considered. In particular, RAS and MMAS have strongly profited

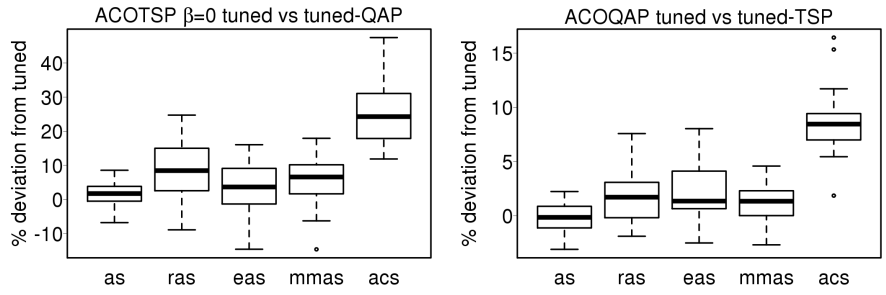

Fig. 17 Left: boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms with tuned-for-QAP parameter settings taking as reference the results obtained by the respective ACO algorithms with tuned-for-TSP parameter settings on the TSP. Positive values indicate that the version using tuned-for-TSP parameter settings obtained a better solution. Right: boxplots of the percentage deviation of the solution quality obtained by the ACO algorithms with tuned-for-TSP parameter settings taking as reference the results obtained by the respective ACO algorithms with tuned-for-QAP parameter settings on the QAP. Positive values indicate that the version using tuned-for-QAP parameter settings obtained a better solution. The results are given across all test instances.

from this tuning; other algorithms such as ACS exploit strongly the best found solutions at the start of the search process already in their default setting and profit to a less extent from tuning. Second, when considering tuned parameter settings, the improved ACO algorithms remain preferable over the basic AS; this is not necessarily true for the default parameter settings from the literature, highlighting the importance of proper tuning. Third, the use of surrogate modeling approaches does not provide advantages over the known ACO algorithms, at least for the settings considered here. Fourth, the more exploitative search behavior implied by the tuned parameter settings seems to be responsible for the improved performance: Our experiments show that tuned parameter settings for one problem under low-budget cases for EAS, RAS, and MMAS can be used to obtain improved performance also on the other problem.

This work can be extended in a number of directions. Even though we have some indication that the tuned parameter settings transfer from one problem to some other, the current tuning setting is not the most realistic one. In an expensive function evaluation setting, tuning would be very time-consuming making it difficult to afford a time-intensive fine-tuning for each different problem being tackled. One possibility could be to obtain more general settings, for example, by tuning ACO algorithms across many different combinatorial problems for low-budget scenarios and in this way derive robust parameter settings. Alternatively, one may obtain a candidate set of different parameter settings that were found to be promising for different problems and select in pre-experiments the most promising ones. Another direction could be to instantiate new ACO algorithms for low-budget situations from a framework of ACO algorithms. One may assemble such algorithms from algorithmic components and instantiate possibly new ones that are superior to the known, pre-defined ACO algorithms. Finally, the poor performance shown by EGO in the problems tackled here suggests to us that there is still room for improvement when applying surrogate modeling approaches (Jones et al, 1998; Knowles et al, 2009) to combinatorial search spaces and, in particular, permutation-based problems.

Acknowledgements The research leading to the results presented in this paper received support from the COMEX project within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office and from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC Grant Agreement No. 246939. Manuel López-Ibáñez and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a postdoctoral researcher and a senior research associate, respectively. Leslie Pérez Cáceres acknowledges support of CONICYT Becas Chile.

References

April J, Glover F, Kelly JP, Laguna M (2003) Simulation-based optimization: Practical introduction to simulation optimization. In: Chick SE, Sanchez PJ, Ferrin DM, Morrice DJ (eds) Proceedings of the 35th Winter Simulation Conference: Driving Innovation, ACM Press, New York, NY, vol 1, pp 71–78

Balaprakash P, Birattari M, Stützle T (2007) Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In: Bartz-Beielstein T, Blesa MJ, Blum C, Naujoks B, Roli A, Rudolph G, Sampels M (eds) Hybrid Metaheuristics, Lecture Notes in Computer Science, vol 4771, Springer, Heidelberg, Germany, pp 108–122

Bersini H, Dorigo M, Langerman S, Seront G, Gambardella LM (1996) Results of the first international contest on evolutionary optimisation. In: Bäck T, Fukuda T, Michalewicz Z (eds) Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), IEEE Press, Piscataway, NJ, pp 611–615

Bullnheimer B, Hartl R, Strauss C (1999) A new rank-based version of the Ant System: A computational study. Central European Journal for Operations Research and Economics 7(1):25–38

Dorigo M (1992) Optimization, learning and natural algorithms. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, in Italian

Dorigo M, Gambardella LM (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. MIT Press, Cambridge, MA

Dorigo M, Maniezzo V, Colorni A (1991) The Ant System: An autocatalytic optimizing process. Tech. Rep. 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy

Dorigo M, Maniezzo V, Colorni A (1996) Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B 26(1):29–41

Fernandez S, Alvarez S, Díaz D, Iglesias M, Ena B (2014) Scheduling a galvanizing line by ant colony optimization. In: Dorigo M, et al (eds) Swarm Intelligence, 8th International Conference, ANTS 2014, Lecture Notes in Computer Science, vol 8667, Springer, Heidelberg, Germany, pp 146–157

Gambardella LM, Montemanni R, Weyland D (2012) Coupling ant colony systems with strong local searches. European Journal of Operational Research 220(3):831–843

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13(4):455–492

Knowles JD, Corne D, Reynolds AP (2009) Noisy multiobjective optimization on a budget of 250 evaluations. In: Ehrgott M, Fonseca CM, Gandibleux X, Hao JK, Sevaux M (eds) Evolutionary Multi-criterion Optimization, EMO 2009, Lecture Notes in Computer Science, vol 5467, Springer, Heidelberg, Germany, pp 36–50

López-Ibáñez M, Prasad TD, Paechter B (2008) Ant colony optimisation for the optimal control of pumps in water distribution networks. Journal of Water Resources Planning and Management, ASCE 134(4):337–346

López-Ibáñez M, Dubois-Lacoste J, Stützle T, Birattari M (2011) The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf`

Moraglio A, Kattan A (2011) Geometric generalisation of surrogate model based optimization to combinatorial spaces. In: Merz P, Hao JK (eds) Proceedings of EvoCOP 2011 – 11th European Conference on Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol 6622, Springer, Heidelberg, Germany, pp 142–154

Moraglio A, Kim Y, Yoon Y (2011) Geometric surrogate-based optimisation for permutation-based problems. In: Krasnogor N, Lanzi PL (eds) GECCO (Companion), ACM Press, New York, NY, pp 133–134

Pellegrini P, Favaretto D, Moretti E (2006) On $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System's parameters. In: Dorigo M, et al (eds) Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Lecture Notes in Computer Science, vol 4150, Springer, Heidelberg, Germany, pp 203–214

Pellegrini P, Mascia F, Stützle T, Birattari M (2014) On the sensitivity of reactive tabu search to its meta-parameters. Soft Computing 18(11):2177–2190

Pérez Cáceres L, López-Ibáñez M, Stützle T (2014) Ant colony optimization on a budget of 1000. In: Dorigo M, et al (eds) Swarm Intelligence, 8th International Conference, ANTS 2014, Lecture Notes in Computer Science, vol 8667, Springer, Heidelberg, Germany, pp 50–61

Pérez Cáceres L, López-Ibáñez M, Stützle T (2015) Ant colony optimization on a budget of 1000: Supplementary material. URL `http://iridia.ulb.ac.be/supp/IridiaSupp2015-004`

Schiavinotto T, Stützle T (2007) A review of metrics on permutations for search space analysis. Computers & Operations Research 34(10):3143–3153

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

Stützle T, Hoos HH (1997) The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and local search for the traveling salesman problem. In: Bäck T, Michalewicz Z, Yao X (eds) Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), IEEE Press, Piscataway, NJ, pp 309–314

Stützle T, Hoos HH (2000) $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Future Generation Computer Systems 16(8):889–914

Teixeira C, Covas J, Stützle T, Gaspar-Cunha A (2012) Multi-objective ant colony optimization for solving the twin-screw extrusion configuration problem. Engineering Optimization 44(3):351–371

Zaefferer M, Stork J, Bartz-Beielstein T (2014a) Distance measures for permutations in combinatorial efficient global optimization. In: Bartz-Beielstein T, Branke J, Filipič B, Smith J (eds) PPSN 2014, Lecture Notes in Computer

Science, vol 8672, Springer, Heidelberg, Germany, pp 373–383
Zaefferer M, Stork J, Friese M, Fischbach A, Naujoks B, Bartz-Beielstein T (2014b) Efficient global optimization for combinatorial problems. In: Igel C, Arnold DV (eds) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2014, ACM Press, New York, NY, pp 871–878
Zeng Q, Yang Z (2009) Integrating simulation and optimization to schedule loading operations in container terminals. Computers & Operations Research 36(6):1935–1944