Analysis of the Population-Based Ant Colony Optimization Algorithm for the TSP and the QAP

Sabrina Oliveira*§, Mohamed Saifullah Hussin†, Andrea Roli‡, Marco Dorigo* and Thomas Stützle*
*IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
Email: {smoreira,mdorigo,stuetzle}@ulb.ac.be
§Federal Center of Technological Education of Minas Gerais (CEFET-MG), Belo Horizonte, MG, Brazil
†PPIMG, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Malaysia
Email: saifullah@umt.edu.my
‡IDEIS, Universitá di Bologna, Cesena, Italy
Email: andrea.roli@unibo.it

***Abstract*—The population-based ant colony optimization algorithm (P-ACO) differs from other ACO algorithms because of its implementation of the pheromone update. P-ACO keeps track of a population of solutions, which serves as an archive of solutions generated by the ants' colony. Pheromone updates in P-ACO are only done based on solutions that enter or leave the solution archive. The population-based scheme reduces considerably the computation time needed for the pheromone update when compared to classical ACO algorithms such as Ant System. In this work, we study the behavior of P-ACO when solving the traveling salesman and the quadratic assignment problem. In particular, we investigate the impact of a local search on P-ACO parameters and performance. The results show that P-ACO reaches competitive performance but that the parameter settings and algorithm behavior are strongly problem-dependent.**

I. Introduction

Ant Colony Optimization (ACO) [4], [6], [7] is a metaheuristic that is mainly used for tackling combinatorial optimization problems. In ACO, the ants search is stochastically biased by heuristic and pheromone information. The way pheromone update is implemented differs across ACO variants, and the choice of an appropriate pheromone update mechanism is essential to obtain effective ACO algorithms. Pheromone update usually involves pheromone evaporation, which is implemented as the reduction of the pheromone trail strength, and the deposit of pheromone by one or several ants. A detailed description of pheromone update procedures of ACO algorithms can be found in [7].

In the population-based ACO (P-ACO) algorithm [11], the pheromone values change according to a population of solutions. This population of solutions has the role of an archive of solutions generated by the ants. Every time a new solution enters the archive, a fixed amount of pheromone is added to the entries of the pheromone matrix that correspond to the solution's components. The pheromone deposited by a solution on its corresponding entries in the pheromone matrix is removed when it leaves the solution archive.

The mechanism used to deposit and remove the pheromone in P-ACO results in a faster pheromone update procedure than in classic ACO algorithms such as Ant System or $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (MMAS). For example, when applied to the traveling salesman problem (TSP) and to the quadratic assignment problem (QAP) on an instance of size n, the pheromone update in the P-ACO algorithm can be computed in $O(n)$, while for Ant System or MMAS it takes $O(n^2)$, if no additional, often problem-specific speed-ups are used.

In this paper, we study in detail the behavior of P-ACO when applied to the TSP and the QAP, extending earlier studies that have been conducted by Guntsch [9] and a few other authors [1], [2], [14], [15], [19]. In particular, we investigate the algorithm behavior examining first different parameter settings and their effect on the algorithm performance as a function of the computation time used. We then study the impact of different strategies for the population management. These strategies decide which solutions enter the population or which are forced to leave the population; different strategies were proposed in [9]. In addition, we also study the effect that local search has on the best choice of parameter settings and of the population management strategy. We also improve P-ACO with a pheromone matrix restart procedure. To put the results by P-ACO in perspective, we compare it to MMAS, a state-of-the-art ACO algorithm for the TSP and the QAP [16].

The paper is organized as follows. In Section II, we describe the P-ACO pheromone update procedure. We present and discuss the experimental setup and results in Section III. In Section IV, we further compare P-ACO and MMAS. Finally, we conclude in Section V.

II. Population-based ACO

P-ACO introduces a solutions archive (also called population) and changes the pheromone update rule [9]–[11] but it keeps the same solution construction and objective function evaluation mechanisms as used in most ACO algorithms [7]. The pheromone update of P-ACO operates on two data structures: the *pheromone matrix* and the *solution archive*. The algorithm starts with an empty solution archive, denoted by P, and all the entries of the pheromone matrix have an initial value τ_0, which has the same effect as the minimum pheromone trail limit in MMAS [16]. The maximum number of solutions in the solution archive is K. At the end of each iteration, the best solution found π is added to the solution archive and is used to update the entry τ_{ij} of the pheromone matrix as follows:

$$\tau_{ij} = \tau_0 + \Delta \sum_{k=1}^{|P|} w_k \cdot I_{ij}^k, \tag{1}$$

where w_k is a weight,

$$\Delta = \frac{\tau_{max} - \tau_0}{K}, \tag{2}$$

τ_{max} is a parameter and I_{ij}^k is an indicator function that is equal to one if the solution component ij is present in the solution generated by ant k and zero otherwise. The pheromone matrix is updated incrementally. Each time a solution enters P, the entries τ_{ij} corresponding to solution components ij of the solution are increased by $\Delta \cdot w_k$.

When the solution archive reaches $|P|$=K solutions, the next solutions built may replace solutions in the archive depending on the mechanism used to define how and when a solution should enter or leave P. The mechanism used defines the pheromone update strategy, which strongly influences the algorithm performance. When this replacement happens, the pheromones corresponding to the components of the solution removed from P are decreased by $\Delta \cdot w_k$.

Several strategies were proposed to manipulate P-ACO's solutions archive. In this paper, we analyze three of the five strategies proposed in [9]: the age-based strategy, the quality-based strategy and the elitist-based strategy.

Age-based strategy. This strategy follows a FIFO *(first in first out)* queue behavior. When the archive is full, the new solution replaces the one that entered the archive longest ago. Pheromone deposit is done on the components of all solutions that enter the archive and the pheromone removal is done on the components of the solutions that leave the archive.

Quality-based strategy. This strategy allows the iteration-best solution π to enter the solution archive only if its solution quality is better than the worst in P. Thus, if π is worse than all solutions in P, the solution archive remains unchanged. When P is full, the new solution replaces the worst of P at that moment.

Elitist-based strategy. This strategy keeps track of the best solutions found during a run. Each time a new best solution e is found, the elitist update is applied. This means that, given a weight $w_e \in [0, 1]$, the best solution found so far receives a weight $w_e \cdot \tau_{max}$ and the other solutions of the population receive a weight $\Delta \cdot (1 - w_e)/(K - 1)$.

III. Experiments

To examine the behavior of P-ACO, several experiments were performed with and without 3-opt local search on the TSP instances and with and without 2-opt local search on the QAP instances. The constant parameters used in P-ACO for the TSP are the same as in [9]: m = 10 ants, β = 2 and α = 1. For MMAS without local search, we set m = $n/4$, where n is the instance size, β = 2, α = 1, ρ = 0.02 for small size instances, ρ = 0.05 for medium size instances and ρ = 0.2 for large size instances. When local search is applied, the following values are set in MMAS: m = 25 and ρ = 0.2.

The TSP instances used in these experiments were taken from TSPLIB: eil101, d198, kroA200, rd400, d657, u724, pcb1173, u1817, d2103, u2319 and are available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. We call the first three instances small, the following three medium-size, and the remaining ones large instances. All the results reported in this paper for the TSP were obtained by an implementation built on top of the ACOTSP software package available at http://www.aco-metaheuristic.org/aco-code/. Hence, the P-ACO algorithm tested in this paper shares the same data structures and speed-up techniques as MMAS.

Unless stated otherwise, the parameter settings used by the algorithms solving the QAP are the same as those used for the TSP, which are considered as the default configuration for both algorithms. The QAP instances used are kra30a, wil50, tai80b, tai100b, taken from the QAPLIB (available at http://www.opt.math.tu-graz.ac.at/qaplib/inst.html) and a large instance, es300 (available at http://iridia.ulb.ac.be/supp/IridiaSupp2010-009/).

Our experiments have been run on an Intel Xeon 2.4 Ghz quad-core CPUs with 6 MB of cache and 8 GB of RAM running under Cluster Rocks Linux. The algorithms studied are coded in `C` and compiled with `gcc` version 4.1.2. For the TSP, we stop the algorithms after 100, 1000, and 2000 CPU seconds for small, medium and large instances, respectively. For the QAP, we stop the algorithms using the time needed to run our re-implementation of Taillard's Robust Tabu Search [18] for $10000 \cdot n$ iterations.

The rest of this section is divided into three subsections: In Subsection III-A we use a profiler to study the impact of the pheromone update on the computation time spent by P-ACO, MMAS and Ant Colony System (ACS) [5]. (Parameter settings of ACS follow the defaults proposed in the literature.) In Subsection III-B we analyze P-ACO specific parameters, and in Subsection III-C we propose an improvement of P-ACO. For the results presented in Subsections III-B and III-C, each algorithm and their variants were executed 20 times on each instance. The qualitative analysis conducted with the aid of solution quality over time (SQT) plots depict the average performance of each algorithm. The plots we choose to present in this paper are just a sample set to represent our findings. Other plots can be found at http://iridia.ulb.ac.be/supp/IridiaSupp2011-010/index.html.

A. Impact of the Pheromone Update on Computation Time

The pheromone update mechanism of P-ACO is typically faster than that of other ACO algorithms. In order to measure the percentage of the overall time spent by the pheromone update procedure and the solution construction procedure, we ran P-ACO and other ACO algorithms using a *profiling* tool.

Each algorithm was executed only once. P-ACO, MMAS and ACS were executed for the TSP. The comparison with ACS for the TSP is interesting because its pheromone update procedure also takes only $O(n)$ steps as P-ACO, but with a

TABLE I

SUMMARY OF RESULTS WITH THE *gprof* PROFILER - PERCENTAGE OF COMPUTATION TIME USED FOR SOLUTION CONSTRUCTION (INCLUDING LOCAL SEARCH IF EXECUTED), $\%cst$, AND FOR PHEROMONE UPDATE, $\%ph$. THE UPPER TABLE REPORTS RESULTS FOR THE TSP INSTANCES; THE LOWER TABLE FOR THE QAP INSTANCES. TWO VARIANTS OF P-ACO, ACS, AND MMAS (WITHOUT LOCAL SEARCH, AND WITH LOCAL SEARCH,+*ls*) WERE USED. NOTE THAT IN MOST CASES THE TOTAL COMPUTATION TIME DOES NOT SUM UP TO 100% DUE TO OTHER FUNCTIONS USED BY THE CODE.

	P-ACO		**ACS**		**MMAS**		**P-ACO+ls**		**ACS+ls**		**MMAS+ls**	
Instance	$\%cst$	$\%ph$	$\%cst$	$\%ph$	$\%cst$	$\%ph$	$\%cst$	$\%ph$	$\%cst$	$\%ph$	$\%cst$	$\%ph$
d198	95.64	**4.09**	81.45	**18.02**	43.32	**55.81**	98.53	**0.61**	93.90	**6.05**	97.08	**1.89**
eil101	93.63	**5.87**	83.02	**16.86**	47.14	**52.56**	97.20	**0.85**	95.50	**4.17**	97.40	**1.24**
kroA200	94.77	**4.86**	77.77	**22.02**	42.70	**56.98**	97.91	**0.85**	95.43	**4.26**	97.44	**1.47**
rd400	94.06	**5.06**	79.67	**20.09**	28.71	**70.03**	98.41	**0.60**	94.17	**5.62**	95.76	**3.09**
pcb1173	94.53	**5.02**	77.49	**22.34**	11.11	**88.21**	97.41	**1.57**	93.15	**6.08**	95.75	**3.86**
u2319	92.45	**4.96**	76.22	**23.64**	9.04	**88.47**	96.01	**3.09**	89.76	**8.17**	90.93	**6.36**

	P-ACO		**MMAS**		**P-ACO+ls**		**MMAS+ls**	
Instance	$\%cst$	$\%ph$	$\%cst$	$\%ph$	$\%cst$	$\%ph$	$\%cst$	$\%ph$
kra30a	99.70	**0.30**	90.46	**9.09**	99.99	**0.01**	99.46	**0.54**
wil50	99.61	**0.38**	90.08	**9.92**	99.99	**0.01**	99.73	**0.27**
tai80b	99.74	**0.26**	90.51	**9.49**	99.99	**0.01**	99.80	**0.20**
tai100b	99.76	**0.24**	90.52	**9.46**	99.99	**0.01**	99.83	**0.16**
es300	99.69	**0.31**	88.92	**11.06**	99.99	**0.01**	99.91	**0.09**

larger constant hidden in the $O(n)$ notation. For the QAP, only P-ACO and MMAS were considered.

The percentages of the computation time required for all functions related either to pheromone update or to solution construction (including local search if executed) were computed and are presented in Table I for the TSP and the QAP. As expected, P-ACO spends much less time updating pheromones than MMAS. The amount of computation time saved by P-ACO is particularly impressive when compared to MMAS on the TSP without a local search. For large instances, up to ca. 90% of MMAS's computation time is spent for updating pheromones. This is due to the fact that the evaporation affects all $O(n^2)$ pheromone entries and that the solution construction is of almost linear complexity due to the exploitation of candidate lists. For P-ACO, the fraction of the computation time used for updating pheromones is always small (around 5% in the discussed case). P-ACO's pheromone update is also much faster than ACS's one.

Once local search is added, the relative importance of P-ACO's fast pheromone update is much reduced, due to two reasons. First, because the local search takes a significant amount of time; second, when used with local search, MMAS does not update the full pheromone matrix, but only the part corresponding to the candidate lists and is therefore much faster. P-ACO's advantage with respect to computation time is nevertheless still clear in the TSP case, where tour length computation is of linear complexity and the solution construction is of (almost) linear complexity. On the QAP, the situation is different. Without local search, the speed advantage of P-ACO with respect to MMAS is much reduced and with local search, the percentage of time spent for pheromone update by P-ACO and MMAS is almost negligible. This is due to the fact that for the QAP the solution construction and the objective function computation are of equal or higher complexity than the phermone update. Consequently, the relative impact of the pheromone update becomes less important.

In summary, the relative importance of the time spent updating pheromones depends on two factors: (1) the use of a local search, and (2) the complexity of the computation of the objective function.

B. Analysis of P-ACO Specific Parameter Settings

The P-ACO age-based algorithm has three specific parameters: K, τ_{max}, and τ_{min}. To understand how they influence the algorithm behavior, we tested the algorithm with different values of K and τ_{max}, while τ_{min} is always set to $1/(n-1)$, which is also the initial pheromone value τ_0. The values we examine are the same as those proposed by Guntsch [9], that is, K = 1, 5, and 25, and τ_{max} = 1, 3, and 10. Some additional values are studied in various places in the paper, in order to refine the observations taken from the results achieved by the parameter values studied in [9]. For this analysis, we consider only the age-based strategy, which is the simplest among the strategies to manage the solution archive. The impact of other strategies on performance is studied later.

1) Analysis of K: The plots in Figure 1 (left column) show that, for both problems without local search P-ACO performs clearly the worst when $K = 1$. A similar behavior was observed across all instance sizes for both problems. This behavior can be explained by the fact that P-ACO is an algorithm that focuses on exploitation, but a larger solution archive counterbalances this by a better exploration of the search space. P-ACO shows the opposite behavior when solving the TSP if local search is applied. The best value of K for solving the TSP now is one. This is probably due to the fact that the search intensification of a local search around one solution is important to guide the algorithm to high-quality solutions. However, for the QAP, P-ACO performs better with a larger value of K and apparently for the QAP a certain diversity of solutions in the solution archive is needed. Note that in Figure 1, τ_{max} is 3 for the TSP. For the QAP it is 3 when local search is applied but 100 when not. These parameter settings are justified in the following.

2) Analysis of τ_{max}*:* The ratio between τ_{max} and τ_{min} for P-ACO is approximately a constant times n. Thus, the amount of pheromone deposited in the pheromone matrix increases linearly with the instance size, the maximum pheromone trail being τ_{max}. In Figure 2, we have an improvement of the final solution quality on the TSP (without local search) when τ_{max}

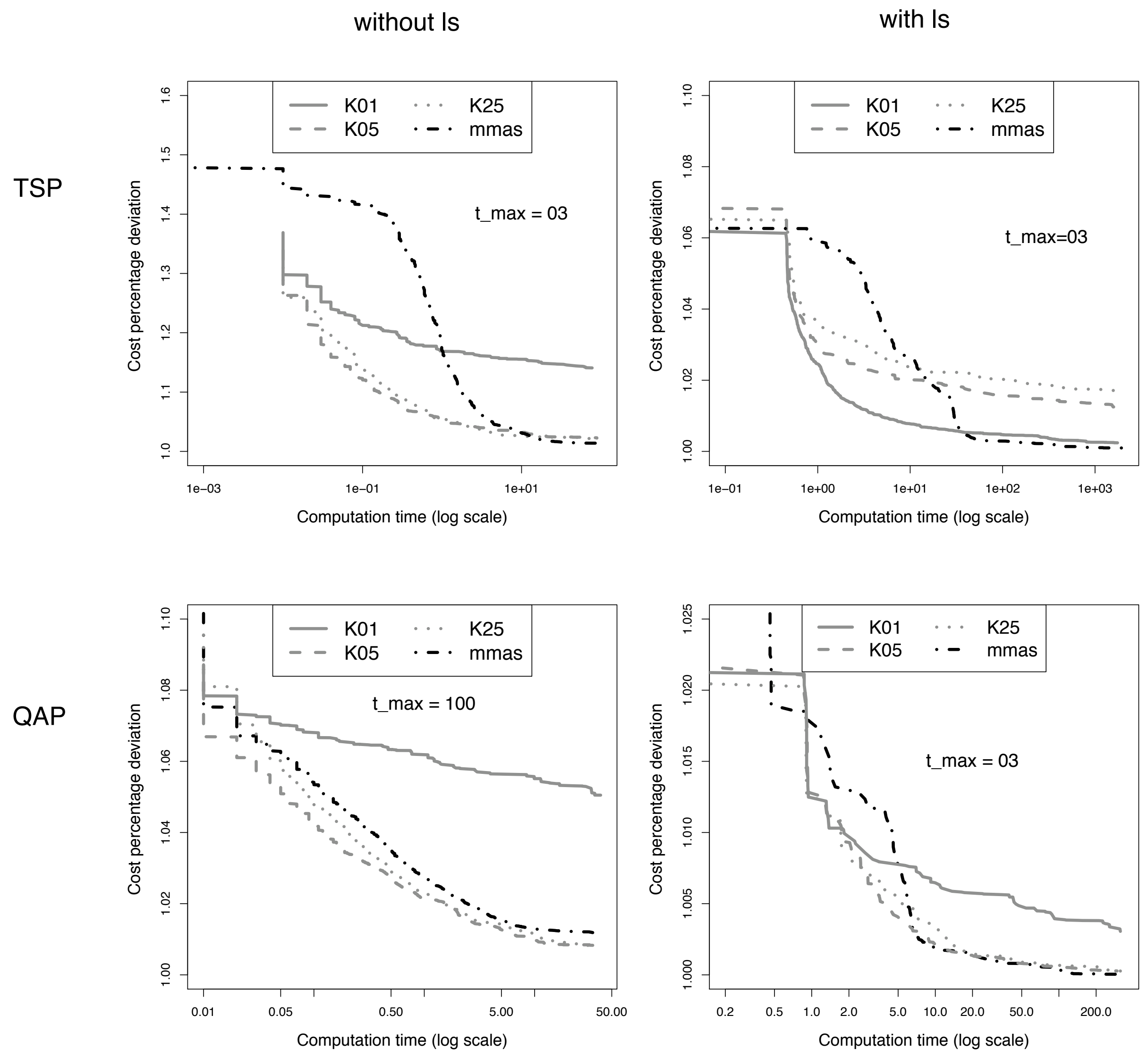

Fig. 1. Solution quality over time of MMAS and P-ACO using different values of K for the best setting of τ_{max}, without local search on instance d198 for the TSP and on instance wil50 for the QAP, with local search on instance u1817 for TSP, and on instance tai100b for QAP.

is equal to 3 or 10. The results show that generally τ_{max} = 3 is able to improve the algorithm's solution quality for the best values of K in comparison to τ_{max} = 1 and that no improvement can be seen in the final solution quality for larger values of τ_{max}, where we had tested also τ_{max} = 50. When local search is applied, no substantial differences can be observed.

For the QAP, we have tested a wider range of τ_{max} values: 1, 3, 10, 30 or 100. We found that increasing the value of τ_{max} considerably improves the performance of P-ACO and that τ_{max} = 100 appears to be the best value for QAP without the use of local search. In case of P-ACO with local search, the best value of τ_{max} is much lower, with a setting of τ_{max} = 3 usually resulting in good performance. We also found that the settings of τ_{max} and K depend on each other. Without local search, P-ACO always obtains good results when large τ_{max} is paired with large K, while with local search, small τ_{max} with large K always gives good results.

Overall, the ratio between τ_{max} and τ_{min} influences, together with the setting of K, how strong the search intensification is. Given the two factors we had mentioned in Subsection III-A about P-ACO speedup dependence, we can now state that (1) the speed advantage of P-ACO is much reduced when local search is applied and (2) P-ACO's parameters need to be adapted when applied to different problems and high performance is desired.

C. Strategies for Updating the Solution Archive

Different strategies to manage the solution archive have been proposed to improve the performance of P-ACO. In order to compare their influence on the algorithm's performance, we have tested three strategies from those presented in [9].

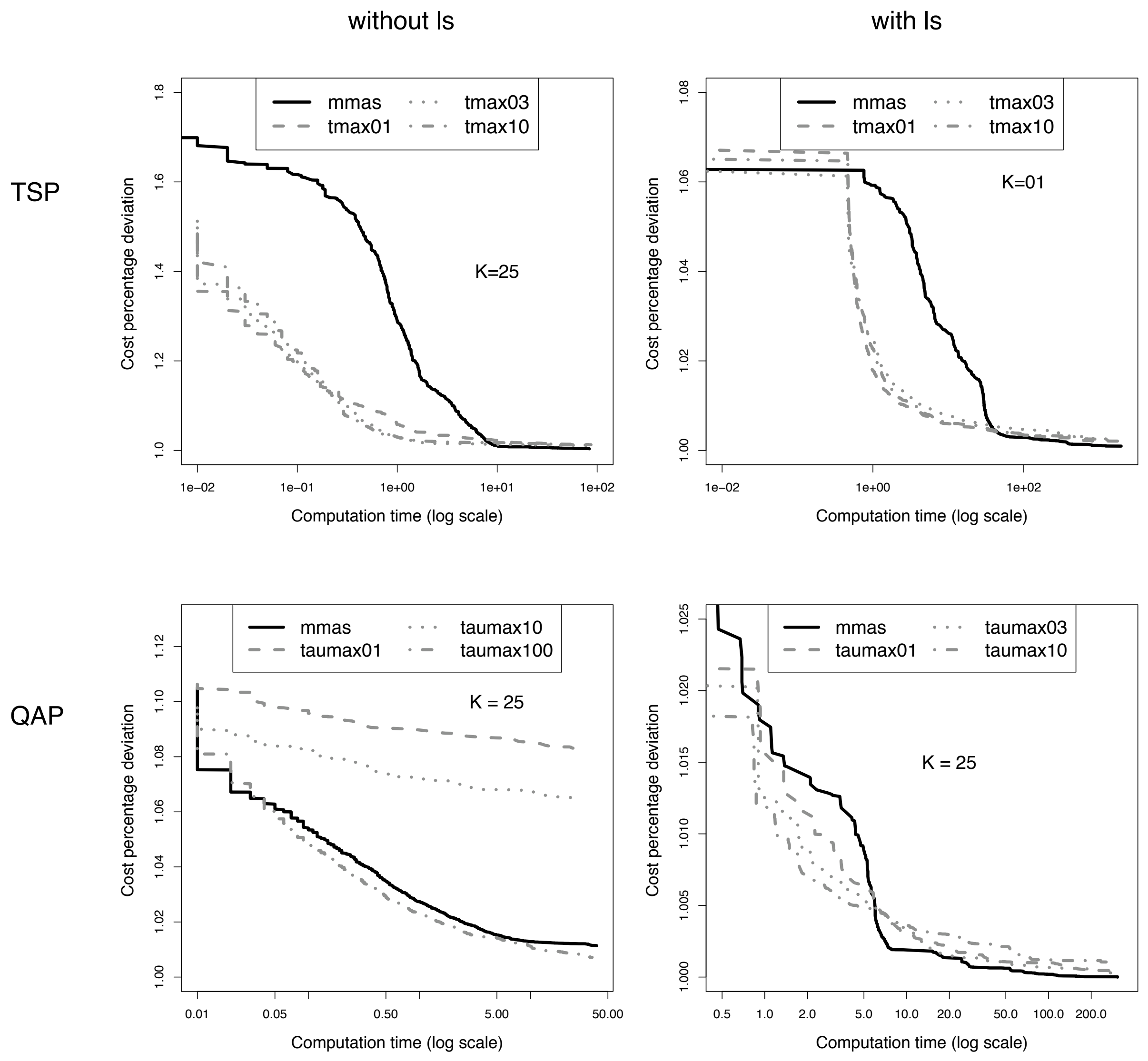

Fig. 2. Solution quality over time of MMAS and P-ACO using different values of τ_{max} for the best setting of K, without local search on instance kroA200 for TSP, and on instance wil50 for QAP, with local search on instance u1817 for TSP, and on instance tai100b for QAP.

All strategies tested here were explained in Section II. We have used for each strategy the best values of K and τ_{max} of Subsections III-B1 and III-B2.

For the TSP, τ_{max} is set to 3, and K is set to 25 when local search is not applied and to 1 when it is applied. For the QAP, τ_{max} is set to 30 when local search is not applied, and to 3 when local search is applied. The value of K is always set to 25. The values chosen for τ_{max} and K are the ones that provide the best tradeoff between the two values. To define the best weight for the elitist-based strategy, we tested three values (w_e = 0.25, 0.5 and 0.75) for both problems. These weights are the same as those proposed in [9].

In Figure 3, we present the plot of the solution quality over time of P-ACO with the best parameter settings among those tested for each population update strategy. For the TSP, the quality-based strategy shows the best performance. The elitist-based strategy with the weight set to either 0.50 or 0.75 obtains good results, too. P-ACO often performs better than MMAS for short computation times, which is not necessarily the case when we look for the final solution quality reached at the end of the run. Indicative results for the QAP are shown in the two plots at the bottom of Figure 3. Without local search, the performance of the elitist-based strategy is similar to that of MMAS, while the other two strategies are much worse. With the use of local search, all three strategies reach high-quality solutions, with the age-based strategy being the best, although on most instances still performing worse than MMAS with respect to the final solution quality reached.

Using other pheromone update strategies, we managed to obtain improved results under specific experimental conditions. Often, P-ACO shows a better performance than MMAS after a short computation time, but does not necessarily

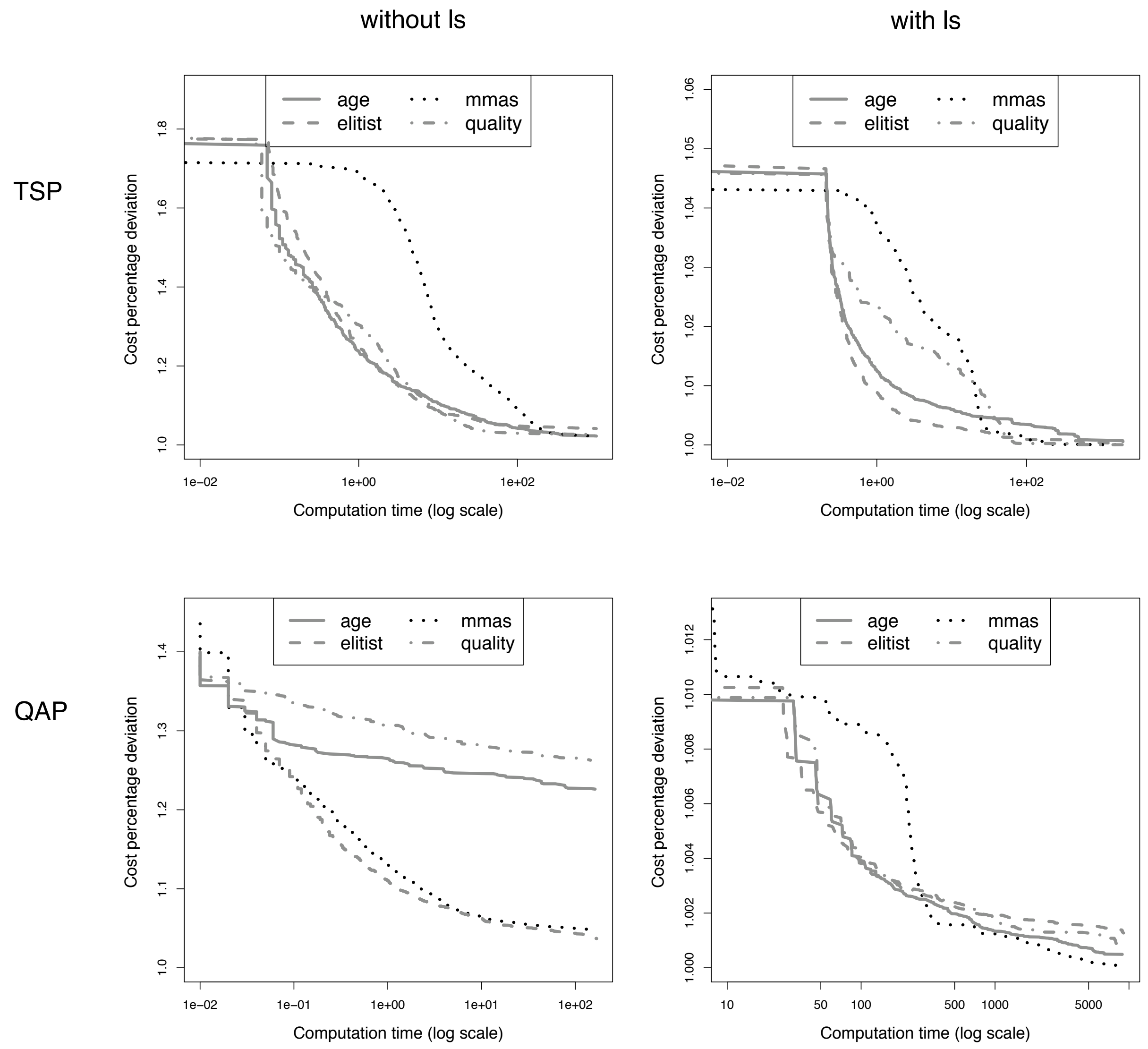

Fig. 3. Solution quality over time of MMAS and P-ACO considering all the strategies tested, without local search on instance d198 for TSP and on instance tai80b for QAP, with local search on instance pcb1173 for TSP, and on instance es300 for QAP.

improve on the final solution quality reached by it. A detailed comparison with the results presented in [9] cannot be made because of the lack of actual data.

D. P-ACO with Restart

P-ACO is an algorithm that presents a strong exploitation capability and, hence, a fast convergence to high quality solutions. On the other hand, its exploration during the search may be insufficient. Following other ACO algorithms such as MMAS, we implemented a restart procedure. The procedure re-initializes the pheromone matrix values to τ_0 after a number r of iterations without improvement. We tested this procedure over a range of values of r. For the TSP, we tested the pheromone re-initialization after n, $25n$, $50n$ and $100n$ iterations when local search was not applied and after 100, 250, 500 and 1000 iterations when it was applied. For the QAP, we examined values of $r \in \{n, 25n, 50n, 100n\}$ without local search, and of $r \in \{10, 25, 50, 100\}$ with local search.

Plots in Figure 4 show the results achieved when solving the TSP, using $r = 50n$ when local search is not applied and $r = 500$ when local search is applied. For the QAP, r is set to $100n$ if no local search is applied, and $r = 10$ if local search is applied. We chose these values as they were the ones for which the algorithm showed the best overall results. In Figure 4, an improvement in the performance of P-ACO can be seen when we compare the final solution quality achieved by the same algorithm with and without the restart procedure, especially when local search is applied. In summary, by adding occasional restarts, the algorithm exploitation capability remains but P-ACO's final solution quality improves.

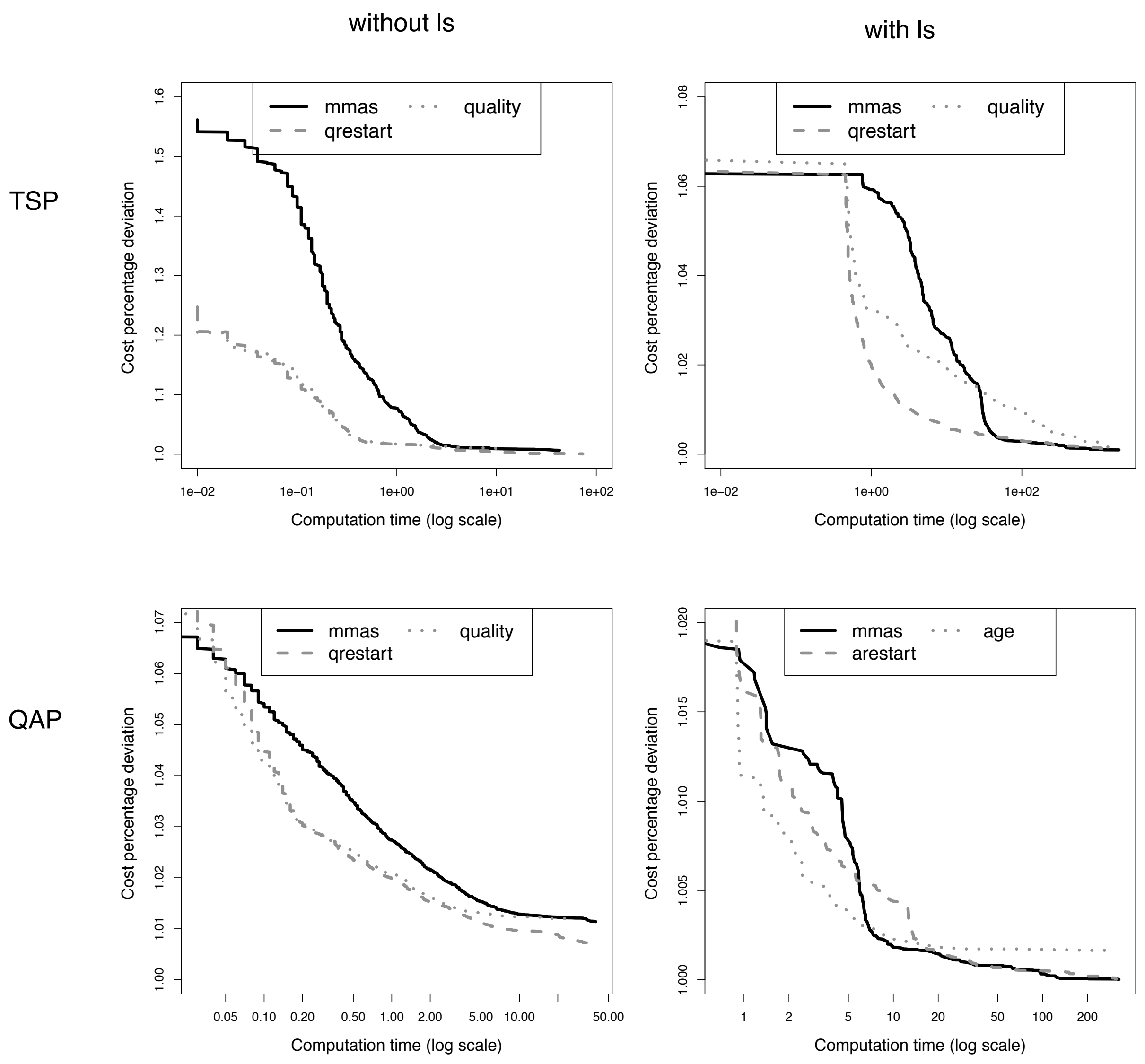

Fig. 4. Solution quality over time of MMAS and P-ACO considering the quality strategy with the restart procedure, without local search on instance d198 for TSP, and on instance wil50 for QAP, with local search on instance pcb1173 for TSP, and on instance tai100b for QAP.

IV. Detailed comparison to MMAS

In this section, we compare the performance of P-ACO and MMAS with and without local search when applied to the TSP and the QAP reporting for both the percentage deviations from the optimal or best-known solutions reached at the termination of an algorithm's run. To test for the statistical significance of the differences, we use the pairwise Wilcoxon signed rank test at a significance level of 0.05. To avoid issues from potential over-tuning, we use in this section other instances than before. For the TSP, all new instances are taken from TSPLIB. For the QAP, the following instances are used: sko42, sko72, tai100a, and wil100 from QAPLIB; bl100, ci100, bl144, and ci144 from the microarray instance set [3], and dre132 and tai175e from Drezner et al. [8]. The P-ACO and MMAS algorithm configurations remain the same as we presented in Section III-D.

The results presented in Table II show that for the TSP without local search, P-ACO performs better than MMAS except on instance kroA100. Differently, the performance of MMAS and P-ACO with local search are similar when applied to large instances. When solving the QAP, P-ACO without local search always obtains better results than MMAS, except on tai175e. In 9 out of 10 instances, the differences are significant. With the use of local search, in most cases, MMAS performs slightly better than P-ACO, except on instance dre132, where the difference is rather large. In 8 out of 10 instances, the differences are significant. Overall, these results indicate that P-ACO with restart appears to be competitive with MMAS.

V. Conclusions

In this article, we have discussed and analyzed the P-ACO algorithm for the TSP and the QAP. Extensive experiments

TABLE II

GIVEN IS THE AVERAGE PERCENTAGE DEVIATION FROM THE KNOWN-OPTIMAL OR THE BEST KNOWN SOLUTIONS (INDICATED BY "**BEST**") OBTAINED BY P-ACO AND MMAS ACROSS 30 TRIALS. INSTANCES WITH NO PUBLICLY AVAILABLE BEST KNOWN SOLUTION (DRE132 AND TAI175E) ARE COMPARED BASED ON THE BEST SOLUTION FOUND IN OUR EXPERIMENTS. THE LEFT TABLE GIVES THE RESULTS FOR THE TSP INSTANCES AND THE RIGHT TABLE THE RESULTS FOR THE QAP INSTANCES. ENTRIES IN BOLD INDICATE STATISTICALLY SIGNIFICANT DIFFERENCES.

Inst.	**best**	**P-ACO**	**MMAS**	**P-ACO+ls**	**MMAS+ls**
kroA100	21282	0.00	0.01	-	-
kroA150	26524	**0.52**	0.97	-	-
rat195	2323	**0.52**	0.59	-	-
pcb442	50778	**0.92**	2.05	-	-
d493	35002	**1.92**	2.40	-	-
rl1889	316536	-	-	**0.15**	0.26
u2152	64253	-	-	0.18	**0.13**
pr2392	378032	-	-	0.16	0.13
pcb3038	137694	-	-	0.24	0.25
fnl4461	182566	-	-	0.33	0.36

Inst.	**best**	**P-ACO**	**MMAS**	**P-ACO+ls**	**MMAS+ls**
sko42	15812	**1.71**	2.44	0.00	0.00
sko72	66256	**1.55**	1.88	0.09	**0.02**
tai100a	21125314	**2.94**	3.04	2.04	**0.99**
wil100	273038	**0.68**	0.91	0.02	0.02
bl100	9272	5.96	6.20	4.62	**2.11**
ci100	523146366	**2.36**	2.92	0.48	**0.27**
dre132	4380	**30.44**	32.03	24.56	**8.41**
bl144	13472	**7.11**	7.56	5.51	**2.35**
ci144	795009899	**3.01**	3.79	0.89	**0.53**
tai175e	57540	73.97	**66.75**	3.67	**1.49**

were conducted to investigate the behavior of P-ACO when different parameter settings and pheromone update strategies are used. A significant part of the comparison between P-ACO and MMAS revealed that for the TSP the pheromone update of the former is much faster than that of the latter. For the QAP, on the other hand, the speed improvement by P-ACO is negligible, since much of the computation time is utilized for the solution construction and the computation of the objective function, which is much more expensive than for the TSP. We have considered strategies for updating P-ACO's solution archive and found that the quality-based strategy is the best option for solving the TSP, while for the QAP the best strategy depends on whether local search is used or not. Without local search, the quality-based strategy appears to be preferable, while with local search, the age-based strategy is better. We also showed that the usage or not of local search has strong impact on parameter settings for P-ACO applied to the TSP. For example, for the case with local search, a population size of one is clearly the best for P-ACO.

Extensive analysis of the development of solution quality over time shows that in many cases P-ACO outperforms MMAS with standard parameter settings for short computation times. However, the situation would possibly be different if MMAS parameter settings were fine-tuned to improve its anytime behavior [13], [17]. When compared to MMAS, P-ACO shows an early stagnation behavior. We have shown that a restart mechanism may significantly improve its performance. Our results allow us to conclude that, with the restart procedure and the right configuration, P-ACO is competitive with state-of-the-art ACO algorithms. However, P-ACO's sensitivity to parameter settings suggests that, as often the case also for other algorithms, the usage of an automatic configuration through tools such as irace [12] is strongly advisable.

Acknowledgments.

We acknowledge support from the COMEX project (P7/36) within the IAP Programme of the BelSPO. Marco Dorigo and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a research director and a senior research associate, respectively.

REFERENCES

[1] D. Angus and C. Woodward. Multiple objective ant colony optimisation. *Swarm Intelligence*, 3(1):69–85, 2009.

[2] M. Clauss, L. Lotzmann, and M. Middendorf. A population-based ACO algorithm for the combined tours TSP problem. *EAI Endorsed Trans. Self-Adaptive Systems*, 2(7):e2, 2016.

[3] S. A. de Carvalho Jr. and S. Rahmann. Microarray layout as a quadratic assignment problem. In *Proc. German Conference on Bioinformatics*, volume P-83 of *Lecture Notes in Informatics*, pages 11–20. GI, 2006.

[4] M. Dorigo. Ant colony optimization. *Scholarpedia*, 2(3):1461, 2007.

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

[6] M. Dorigo, M. A. Montes de Oca, S. Oliveira, and T. Stützle. Ant colony optimization. In J. J. e. a. Cochran, editor, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2011.

[7] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[8] Z. Drezner, P. Hahn, and É. D. Taillard. A study of quadratic assignment problem instances that are difficult for meta-heuristic methods. *Annals of Operations Research*, 2003.

[9] M. Guntsch. Ant Algorithms in Stochastic and Multi-Criteria Environments. PhD thesis, Universität Fridericiana zu Karlsruhe, Germany, 2004.

[10] M. Guntsch and M. Middendorf. Applying population based ACO to dynamic optimization problems. In M. Dorigo et al., editors, *ANTS 2002*, volume 2463 of *LNCS*, pages 111–122. Springer, Heidelberg, 2002.

[11] M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni et al., editors, *EvoWorkshops 2002*, volume 2279 of *LNCS*, pages 71–80. Springer, Heidelberg, 2002.

[12] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[13] M. López-Ibáñez and T. Stützle. Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, 235(3):569–582, 2014.

[14] W. Shi, T. Weise, P. R. R. Chiong, and B. Çatay. Hybrid PACO with enhanced pheromone initialization for solving the vehicle routing problem with time windows. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1735–1742. IEEE Press, 2015.

[15] R. Skinderowicz. Population-based ant colony optimization for sequential ordering problem. In M. Núñez, N. T. Nguyen, D. Camacho, and B. Trawiński, editors, *ICCCI 2015*. Springer, 2015.

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

[17] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, and M. Dorigo. Parameter adaptation in ant colony optimization. In Y. H. et al., editor, *Autonomous Search*, pages 191–215. Springer, Berlin, Germany, 2012.

[18] É. D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[19] T. Thalheim, D. Merkle, and M. Middendorf. A hybrid population based ACO algorithm for protein folding. In S. I. A. et al., editor, *IMECS '08*, Lecture Notes in Engineering and Computer Science, pages 200–205. International Association of Engineers, Newswood Limited, 2008.