Pre-scheduled and adaptive parameter variation in MAX-MIN Ant System

Michael Maur, Manuel López-Ibáñez,* and Thomas Stützle

Abstract— **MAX-MIN Ant System (MMAS) is an ant colony optimization (ACO) algorithm that was originally designed to start with a very explorative search phase and then to make a slow transition to an intensive exploitation of the best solutions found during the search. This design leads to a rather slow initial convergence of the algorithm, and, hence, to poor results if the algorithm does not run for sufficient time. This article illustrates that varying the parameter settings of MMAS while solving an instance may significantly improve the anytime search behavior of MMAS. Even rather simple pre-scheduled variations of the parameter settings that only depend on the number of iterations or the computation time show improvement over fixed parameter settings. The degree of improvement, however, is not uniform across problems. In particular, the improvement is very strong in the traveling salesman problem (TSP) but small, if at all noticeable, in the quadratic assignment problem (QAP). This paper also presents an adaptive parameter variation specifically designed for the TSP. The experimental results show that the pre-scheduled variations are comparable to the proposed adaptive variation in terms of the anytime behavior of MMAS.**

I. Introduction

Ant colony optimization (ACO) is a metaheuristic inspired by the swarm behavior of some species of ants [1]. Among the most performing ACO algorithms are variants such as ant colony system (ACS) [2], MAX-MIN ant system (MMAS) [3], [4], and rank-based ant system [5].

Although it is now known that effective ACO algorithms typically include local search to improve the solutions generated by the ants [1], [2], [3], [6], early comparisons of ACO algorithms applied to the TSP focused on variants without local search. Moreover, comparisons were done for rather long runs, typically up to $10\,000 \cdot n$ solution constructions. The reason for such long runs is probably the influence of the first international contest on evolutionary optimization [7], which used this stopping criterion for symmetric TSP instances. When comparing ACO algorithms using this stopping criterion, MMAS emerges as a top performing contender [4]. This is maybe not surprising because MMAS was targeted to be an effective ACO algorithm for rather high computation times (or, equivalently, a rather large number of solution constructions). MMAS was designed to have a relatively long initial exploration phase with a subsequent transition to a strong exploitation phase. With these default settings [4], MMAS reaches very good results (when compared to other ACO algorithms) on TSP instances of few hundred cities. A main drawback of these design choices is, however, that MMAS has a slow convergence towards high quality solutions. Convergence is faster in MMAS variants that exploit local search, mainly because such variants use a smaller number of ants and a higher pheromone evaporation rate. Nevertheless, even with local search, MMAS still has a slower convergence to high-quality solutions than more aggressive ACO algorithms such as ACS.

In this article, we examine possibilities for improving the convergence speed of MMAS towards high quality solutions (or said in other words, the *anytime behavior* [8] of MMAS) by using schemes for the adaptation of parameter values at computation time. While the study of online parameter variation has mostly focused on evolutionary algorithms (EAs) [9] and reactive search approaches [10], there are only few works studying online parameter variations in ACO algorithms. A recent comprehensive review of such approaches [11] concludes that most of these works propose rather complex adaptive strategies, without a clear understanding of the effect of different settings during a single run of an ACO algorithm. In contrast, in our research efforts we have first studied the impact of various non-default parameter settings on the convergence speed. Interestingly, modified parameter settings can strongly improve upon the default ones for short run times, but they often have a detrimental effect for longer runs. Based on these insights, we first investigate simple pre-scheduled variations of parameter settings, where the value of a parameter depends only on the current time or iteration. Our experimental results show considerable improvements to the anytime behavior of MMAS with and without local search when applied to the TSP. We further test whether these performance improvements carry over to other application problems, where local search plays a very dominant role, using the quadratic assignment problem (QAP) as a case study. Pre-scheduled parameter variation can improve over the default settings for the MMAS application to the QAP [12], [4]; however, this is the case mainly because these parameter settings left room for improvement and fixed parameter settings with a better anytime performance exist. Finally, we compare the pre-scheduled parameter variations with an adaptive parameter strategy specifically designed for the TSP. This adaptive strategy modifies parameter settings depending on the distance between solutions. Our experiments show that the adaptive strategy effectively adapts parameter values towards very good settings at each stage of the search. Nonetheless, the comparison between the adaptive

* Manuel López-Ibáñez (corresponding author) and Thomas Stützle are with the IRIDIA laboratory, Université Libre de Bruxelles, CP194/6, Av. F. Roosevelt 50, 1050 Brussels, Belgium (email: {manuel.lopez-ibanez,stuetzle}@ulb.ac.be).

Michael Maur is with Fachbereich Rechts- und Wirtschaftswissenschaften, TU Darmstadt, Darmstadt, Germany.

978-1-4244-8126-2/10/$26.00

and the fine-tuned, pre-scheduled strategy shows that the latter is able to match the performance of the former.

The paper is structured as follows. In the next section, we give a short overview of MMAS. In Section III, we examine the performance of parameter schedules. Section IV discusses the results for MMAS applied to the QAP. Section V introduces a parameter adaptation strategy designed specifically for the TSP, and compares it with one of the previous pre-scheduled variation strategies. We conclude in Section VI.

II. MAX-MIN Ant System

MMAS is an ACO algorithm that builds directly upon AS, incorporating a much more aggressive pheromone update procedure and mechanisms to avoid search stagnation. When applying MMAS to the TSP, each ant starts at a randomly chosen initial city, and constructs a tour by randomly choosing at each step the city to visit next according to a probability defined by pheromone trails and heuristic information. In particular, the probability of ant k choosing a successor city j when being at city i is given by

$$\mathbf{p}_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in N^k} [\tau_{ih}]^{\alpha} \cdot [\eta_{ih}]^{\beta}}, \qquad (1)$$

where τ_{ij} is the pheromone trail strength associated to edge (i, j), η_{ij} is the corresponding heuristic information; α and β are two parameters that influence the weight given to pheromone and heuristic information, respectively; and N^k is the feasible neighborhood, that is, a candidate list of cities not yet visited in the partial tour of ant k.

Departing from the traditional MMAS but following previous work [12], [13], we also incorporate the *pseudo-random action choice rule* of ACS [2], which allows for a greedier solution construction. With a probability q_0 an ant chooses next a city j such that

$$j = \arg \max_{h \in N^k} \{[\tau_{ih}]^{\alpha} \cdot [\eta_{ih}]^{\beta}\}; \qquad (2)$$

otherwise, the ant performs the probabilistic selection based on Eq. 1. A value of $q_0 = 0$ disables the pseudo-random action choice rule and reverts back to the traditional MMAS.

The pheromone update of MMAS updates all pheromone trails as

$$\tau_{ij} \leftarrow \max\{\tau_{\text{min}}, \min\{\tau_{\text{max}}, (1-\rho) \cdot \tau_{ij} + \Delta\tau_{ij}^{\text{best}}\}\}, \qquad (3)$$

where ρ, $0 < \rho \leq 1$, is a parameter called evaporation rate and

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} F(s_{\text{best}}) & \text{if edge } (i,j) \in s^{\text{best}}, \\ 0 & \text{otherwise.} \end{cases} \qquad (4)$$

The best solution (s^{best}) is either the iteration-best solution, the best-so-far solution or the best solution since a re-initialization of the pheromone trails (restart-best). In MMAS, these solutions are chosen alternately [4].

There are essentially three mechanisms for search diversification in MMAS. First, the explicit pheromone trail limits (τ_{max} and τ_{min}) play the main role by guaranteeing that there is a minimum probability of selecting any feasible choice. Second, the initialization of the pheromone trails to τ_{max}, together with a relatively small evaporation rate, produces a small variability among the pheromone levels at the start of the search, and, hence, the sampling of the space is rather uniform. Later in the search, once the pheromone trails converge to reflect the best found solutions, MMAS goes into an exploitation phase. Finally, pheromone trails are occasionally re-initialized to τ_{max} if the algorithm has not improved the best-so-far solution in a number of iterations.

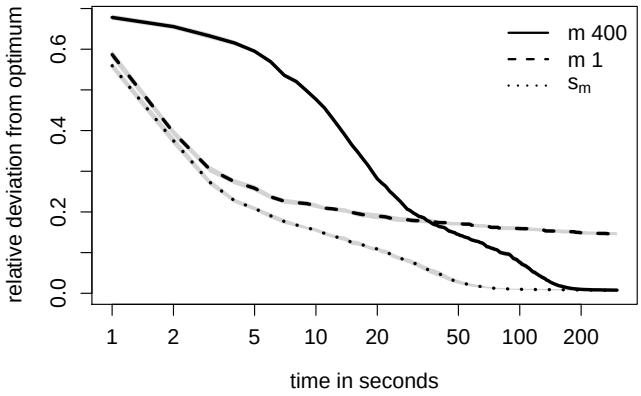
The application of MMAS to the quadratic assignment problem (QAP) [4] is similar to the application to the TSP. The main differences are that MMAS for the QAP does not use any heuristic information and that the pheromone information represents the assignment of an object to a location. More details about the application of MMAS to the QAP can be found in the original publication [4].

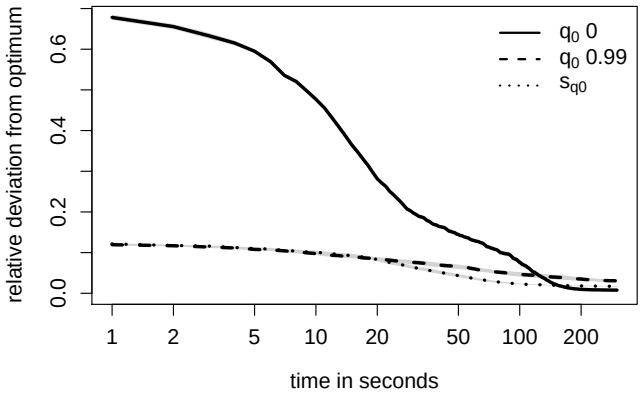
In the next sections, we report experimental results of MMAS for both the TSP and QAP with different fixed settings and various pre-scheduled parameter variations. The experiments are run on a cluster of Intel Xeon™ E5410 quad-core processors with 2.33 GHz CPUs with 6 MB L2-Cache and 8 GB RAM under Rocks Cluster GNU/Linux. Due to the sequential implementation of the code, only one core is used for running the executable. The algorithms are compiled with `gcc`, version 3.4. Results for the TSP are generated with the publicly available ACOTSP software [14]. The implementation of MMAS for the QAP follows the original proposal [4].

III. Pre-scheduled parameter variations for the TSP

We report in this section the result of pre-scheduled variations of the number of ants (m) and the greediness factor (q_0, Eq. 2) for the application of MMAS to the TSP. For reasons of limited space, we do not report here results of pre-scheduled variations of the influence of heuristic information (β), although initial experiments suggest that pre-scheduled variation of β further improves the anytime behavior of MMAS [11]. On the other hand, we could not find any pre-scheduled variation of the evaporation factor (ρ) that would significantly enhance the anytime behavior of MMAS over fixed parameter settings.

We confirmed our results in four random Euclidean instances for each size of 100, 200, 300 and 400 cities (MMAS without local search), and 1 500, 3 000 and 6 000 cities (MMAS+ls). Here, we only show results for an instance of size 400 and another one of size 3 000, but the results are representative of those obtained on other instances. Each plot shows the results on a single instance. The x-axis shows the computation time in seconds in logarithmic scale, whereas the y-axis gives the relative deviation from the optimum. Each line is the mean quality of the best-so-far solution over 25 independent runs with different random seeds. To assess the variability among different runs, we plot a 95% confidence interval around each mean value as a gray shadow





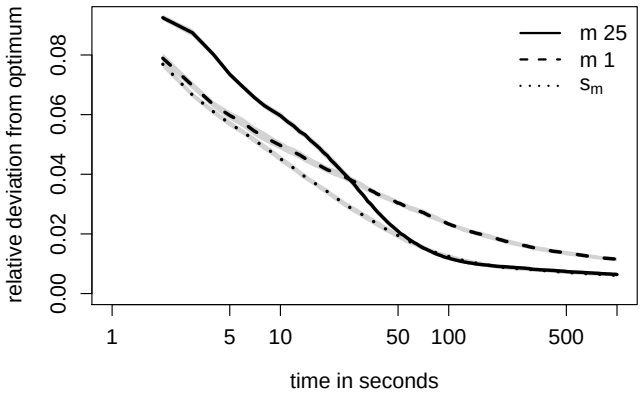


Fig. 1. Comparison of fixed and pre-scheduled variation of parameter m for MMAS (top) and MMAS+ls (bottom). The pre-scheduled parameter variation (s_m) starts at $m = 1$, adding one ant every ten iterations.

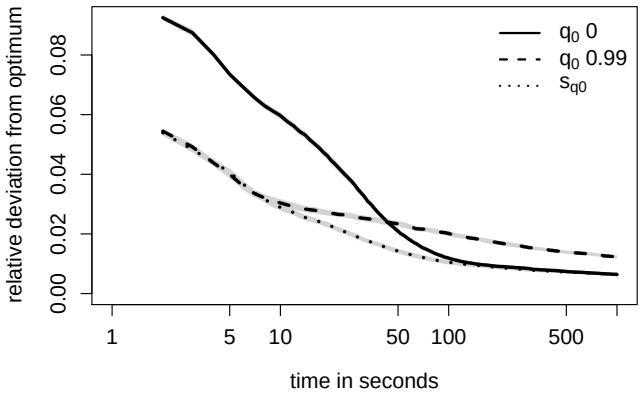


Fig. 2. Comparison of fixed and pre-scheduled variation of parameter q_0 in MMAS (top) and MMAS+ls (bottom). The pre-scheduled parameter variation (s_{q0}) starts at $q_0 = 0.99$ and decreases by 0.001 every 2 iterations until $q_0 = 0$.

behind each line. The variability is very small in the case of the TSP, and therefore the confidence intervals are not visible in many plots (c.f. Fig. 1). In the case of the QAP, the variability is much higher and the confidence intervals are clearly visible (c.f. Fig. 4).

We first examine pre-scheduled variations of the number of ants (m). We propose a simple schedule that starts with a single ant ($m = 1$) and slowly increases the number of ants at a rate of one ant every 10 iterations. We compare in Fig. 1, for both, MMAS and MMAS+ls, this parameter schedule (s_m) and several fixed settings: a single ant ($m = 1$), and the default setting ($m = n$ for MMAS, and $m = 25$ for MMAS+ls) [1]. In both cases, the pre-scheduled variation (s_m) obtains very good results for short runtimes, similar to those obtained with a single ant. In addition, it matches the final performance of the default settings for longer runtimes. We observed the same results across all the instances tested in this study. A single ant quickly converges to good solutions by performing many more iterations in a same time. On the other hand, a larger number of ants sample more effectively around the best solution without incurring the overhead of updating the pheromone information at every iteration.

Next, we consider pre-scheduled variations of the parameter q_0. In this case, we propose a schedule (s_{q0}) that starts with a high value of $q_0 = 0.99$ and decreases q_0 at a rate of 0.001 every two iterations until reaching $q_0 = 0$, disabling the greedy choice and reverting to the traditional MMAS. Figure 2 illustrates, for both MMAS and MMAS+ls, the comparison between the pre-scheduled variation (s_{q0}) and fixed settings of $q_0 = 0$ (the default for MMAS) and $q_0 = 0.99$. The conclusions from this and similar plots for other instances is that the pre-scheduled variation s_{q0} is able to match the best performance of both fixed settings, providing a superior result if the runtime is not known in advance, and, therefore, improving the anytime behavior of MMAS. Our explanation of the success of the pre-scheduled variation is that higher settings of q_0 focus the search around the best-so-far solution, speeding up the convergence of the algorithm and quickly obtaining good results. However, if there is enough computation time available, lower settings of q_0 are necessary to increase exploration.

Finally, we also tested the combination of pre-scheduled variations of both parameters m and q_0. However, there is a strong interaction between these two parameters, and varying both at the same time does not lead to a strong improvement. We use Fig. 3 to illustrate the general observations. First, we notice that varying only q_0 produces much better results than varying only m, especially in MMAS+ls and for short runtimes. In the case without local search, varying both parameters (s_m, s_{q0}) almost matches the performance of only s_{q0} for short runtimes, whereas it matches the performance of

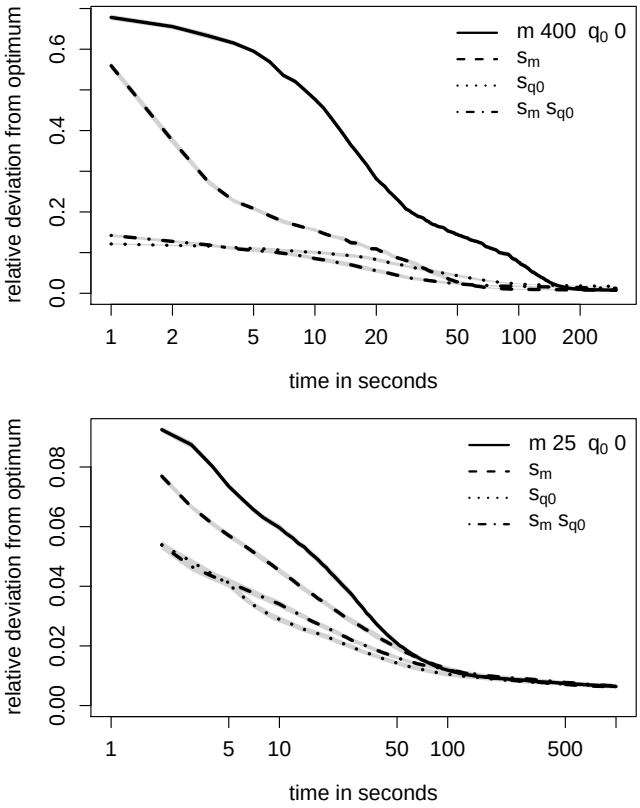


Fig. 3. Pre-scheduled variation of both parameters m and q_0 in MMAS (top) and MMAS+ls (bottom). The variation schemes are the same used in Figs. 1 and 2.

only s_m for the remainder of the run, effectively combining the best behaviors of both approaches. The overall conclusion is that it is more important to vary q_0 than m when using local search, whereas without local search there are some advantages to the combined pre-scheduled variation of both parameters.

IV. Pre-scheduled parameter variations for the QAP

As the next step, we analyzed the behavior of pre-scheduled parameter variations on the effectiveness of MMAS for the QAP. As above, we consider schedules varying the number of ants m and the parameter q_0. Given that the performance of ACO algorithms for the QAP is very poor without local search, we directly focus on the MMAS algorithm with local search (henceforth, denoted simply by MMASQAP). The local search algorithm is a best-improvement algorithm that stops upon hitting a local optimum with respect to the 2-exchange neighborhood, where the 2-exchange neighborhood exchanges the location of two objects. In the QAP case, we have compared two pre-scheduled parameter variations, a fast and a slow one, of each parameter. In the fast schedule of q_0 (fast s_{q0}), the initial value of 0.8 is decreased by 0.05 every 4 iterations until a final value of $q_0 = 0$, whereas the fast schedule of m (fast s_m) starts with one ant and adds an additional ant every 4 iterations until a maximum of 15 ants. In the slow parameter schedules (slow s_{q0} and slow s_m), the initial parameter values ($q_0 = 0.8$ and $m = 1$) are kept constant during 15 iterations; after 15 iterations, slow s_{q0} decreases the value of q_0 by 0.05 every 5 iterations until reaching a value of 0.3, whereas slow s_m adds one ant every 5 iterations until reaching seven ants.

We test these pre-scheduled variations and various fixed parameter settings on several benchmark instances from QAPLIB. Plots in this section are similar than those for the TSP, and in particular, each line is the mean quality of the best-so-far solution over 20 independent runs, and we plot behind each line a 95% confidence interval around the mean value. We give exemplary plots comparing the slow and fast schedules to fixed parameter settings in Figures 4 and 5, for m and q_0, respectively. The general observation is that for the number of ants the slow schedule is preferable, whereas for q_0 the differences between the fast and the slow schedules are not fully clear. In a few instances, the pre-scheduled variations improved performance over the default fixed parameter settings for short running times. However, we found in most instances better fixed settings than the default ones, and, in many cases, the confidence intervals of the various settings overlap.

We check also the results if both parameters are varied concurrently, using at the same time a fast schedule for q_0 and a slow one for m. We illustrate the results in Figure 6. Although the combined schedule reaches better solution quality for very short computation time, we cannot claim that the combined schedule provides any clear improvement.

V. An Adaptive MAX-MIN Ant System Variant: MMASdde

In this section, we compare the pre-scheduled parameter variation of q_0 to an adaptive scheme for modifying the same parameter in the application of MMAS+ls to the TSP. Adaptive strategies try to use the information that is gained at computation time during the search to adapt the values of parameters, and they are therefore very different from pre-scheduled parameter variations. The results presented in this section were obtained without the usual pheromone re-initialization of MMAS. However, some preliminary experiments suggest that the method is quite robust and future extensions of this work should contrast these results with the standard pheromone reinitialization of MMAS.

We propose a new adaptive strategy that we call *MAX-MIN Ant System with distance dependent elitism* (MMASdde). The idea of this adaptive strategy is based on the observation that the best solutions during a run of MMAS+ls are typically generated when the average distance among the tours is a small constant. In other words, the algorithm seems to perform best when there is an optimal degree of heterogeneity among the solutions generated. Here we measure the heterogeneity between two tours by their distance in the solutions space; in particular, we use the bond distance, which is given by the number of cities n minus the number of common edges.

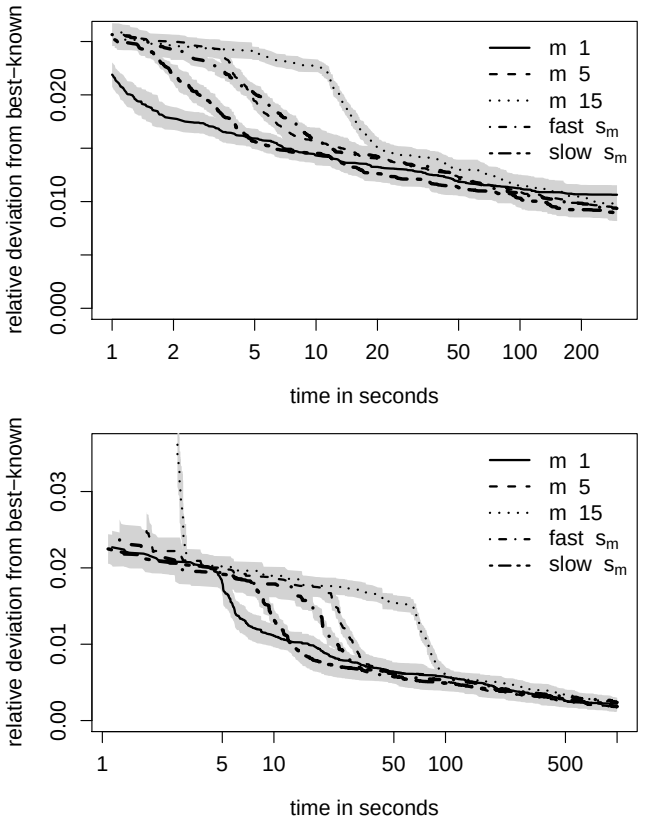


Fig. 4. Comparison of fixed and pre-scheduled variation of parameter m in MMASQAP. Results are for instance `tai100a` (top) and instance `tai150b` (bottom) from QAPLIB.

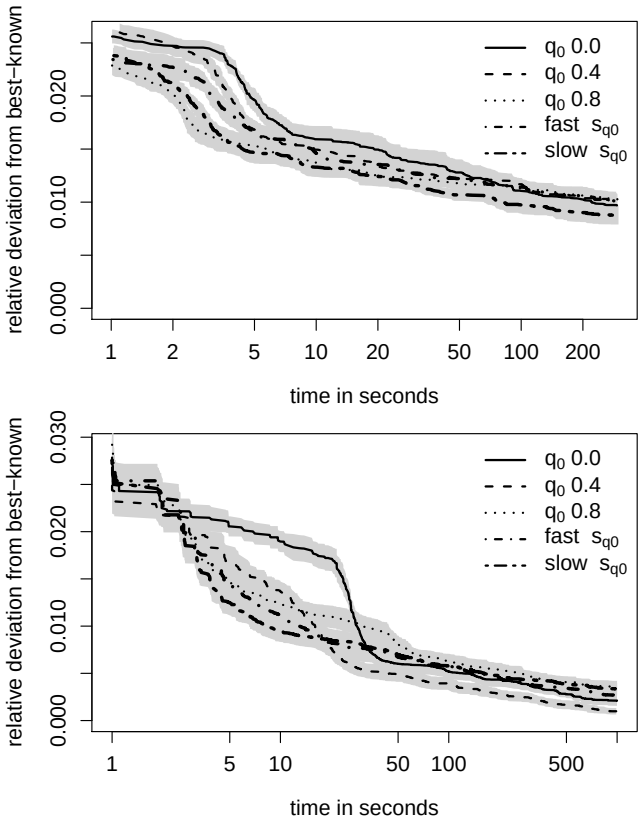


Fig. 5. Comparison of fixed and pre-scheduled variation of parameter q_0 in MMASQAP. Results are for instance `tai100a` (top) and instance `tai150b` (bottom) from QAPLIB.

As a first step, we determine appropriate distance values. In particular, we run the default variant of MMAS+ls and compute the average distance among tours when the algorithm converges, which is also the point when the best solutions are found. We measure this value for various instance sizes and compute a linear regression. The results of the linear regression are given in Table I. According to the linear regression, the average distance of the reference configuration, thereafter called the *target distance* $d_{\text{avg}}^{\text{target}}$, can be derived as:

$$d_{\text{avg}}^{\text{target}} = 12.8243575 + 0.00390003 \cdot n \qquad (5)$$

Preliminary results indicate that MMASdde is not extremely sensitive with respect to the actual target distance. Here, we use the value predicted by the linear regression as the target distance for any of the instances we test.

Taking into account the information above, we propose an adaptive strategy (a_{q0}), where the value of q_0 is increased or decreased depending on the average distance among tours in the current iteration. The rule of thumb we use is based on the intuition that increasing q_0 should lead to a decrease of the average distance, while the decrease of q_0 should lead to an increase of the average distances. This effect is because q_0 determines the greediness of solution construction: high values of q_0 favor the selection of a small group of the best-performing edges, leading to the construction of relatively similar solutions, while low values of q_0 increase the chances of less visited edges becoming part of a tour, and, hence, lead to higher average distance. This rule of thumb is used to modify the value of q_0 throughout a run, trying to keep the average distance among solutions close to a target distance.

TABLE I

AVERAGE SOLUTION DISTANCE AT THE END OF THE RUN, USING THE MMAS REFERENCE CONFIGURATION ON DIFFERENT PROBLEM SIZES: EXPERIMENTAL RESULTS AND LINEAR REGRESSION

instance	average distance	considered iterations	distance predicted by linear regression	regression residual
1000-1	15.62	45,001-50,000	16.72	-1.10
1500-1	17.10	23,001-28,000	18.67	-1.57
2000-1	22.89	13,001-18,000	20.62	2.27
2500-1	22.14	8,001-12,000	22.57	-0.43
3000-1	26.34	7,501-10,000	24.52	1.82
3500-1	25.74	5,501-8,000	26.47	-0.73
4000-1	27.67	4,751-6,000	28.42	-0.75
4500-1	32.35	4,501-5,500	30.37	1.98
6000-1	34.77	35,001-40,000	36.22	-1.45

For a precise definition of the adaptive parameter scheme, we need to consider three aspects: (1) the degree of modification of the parameter value, (2) the initial parameter value, and (3) the frequency of modification. Standard possibilities to modify the parameter value are either to add/remove a constant amount or to multiply/divide by a

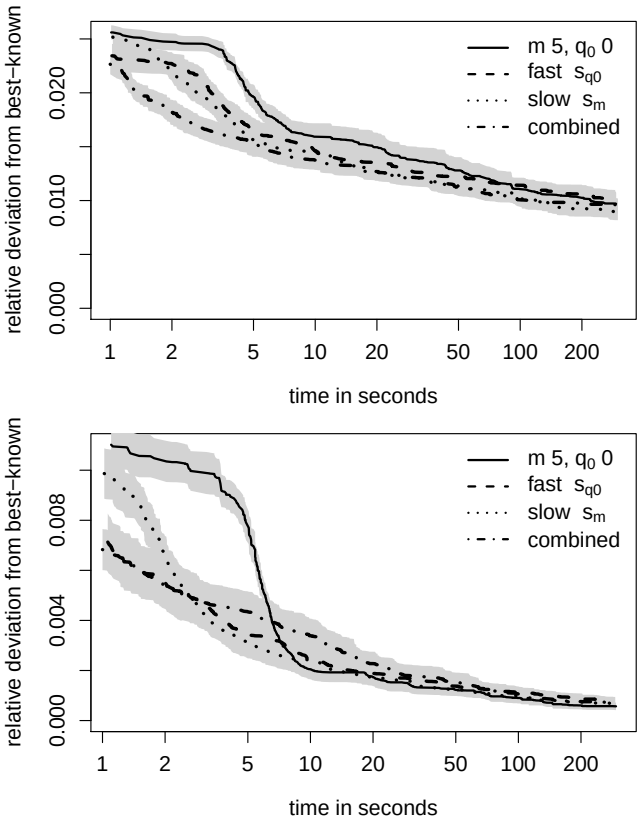


Fig. 6. Pre-scheduled variation of both parameters m and q_0 in MMASQAP. Results are for instance `tai100a` (top) and instance `sko100a` (bottom) from QAPLIB.

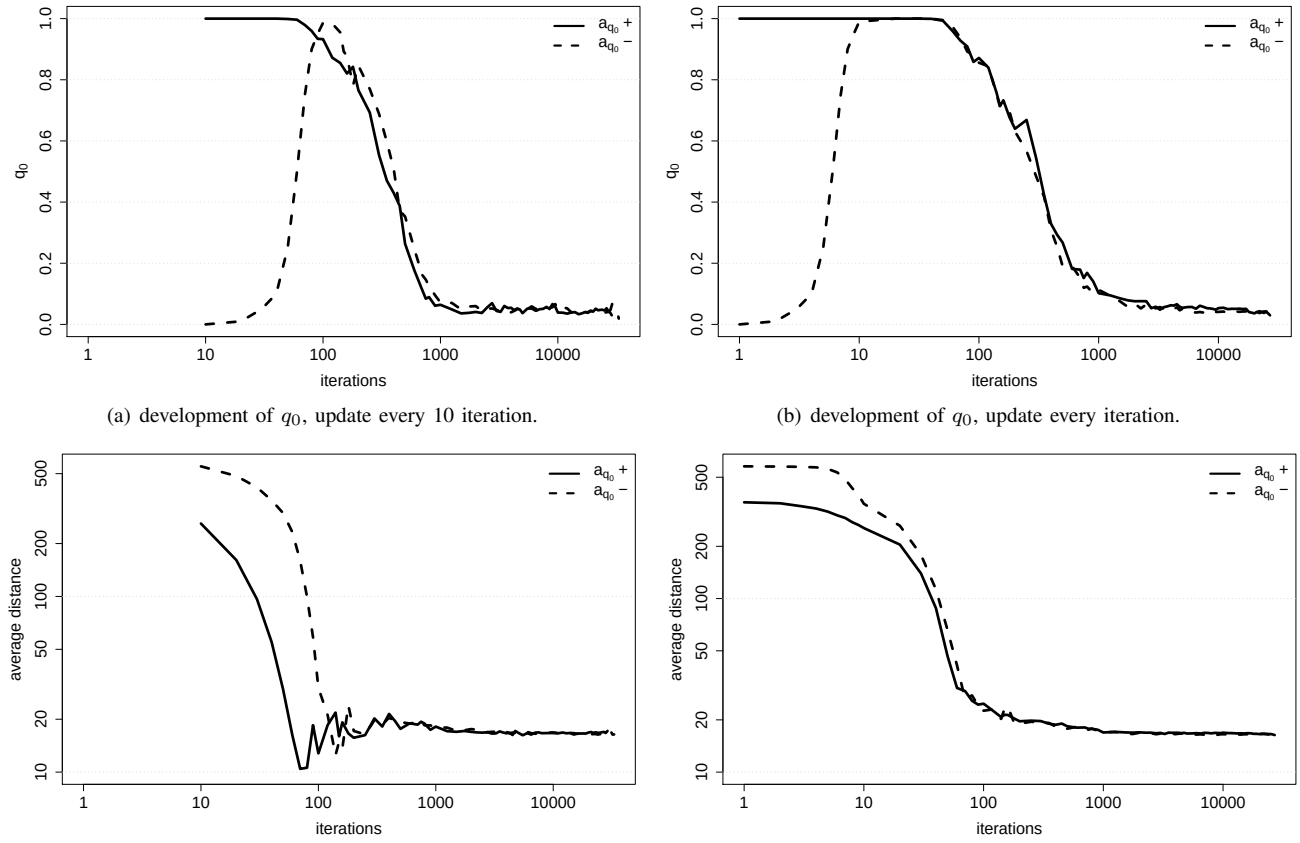
constant factor. We consider here a different approach that uses a finite ordered set of possible values of q_0, and modifications of q_0 change the current value by replacing it with the next larger (or smaller) value in the set. We examined several possible sets of values in some preliminary experiments, however, we did not find strong differences among them, so we arbitrarily choose the following: $q_0 \in \{0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 1.0\}$. For the initialization of q_0, we explored two possibilities. Either we start at the highest possible value if the set, which is identified as $\mathrm{a}_{q0}+$, or at the lowest possible value, identified as $\mathrm{a}_{q0}-$. While $\mathrm{a}_{q0}+$ is more appropriate, considering the results in Section III, the use of $\mathrm{a}_{q0}-$ demonstrates the ability of the adaptive strategy to cope with a suboptimal initialization. Finally, we have to establish the frequency of adaptation of q_0. As the computation of the average solution distance is somewhat computationally expensive ($O(m^2n)$), there is a trade-off between frequent changes, which are expensive but keep the average distance close to the target level, and slower changes, which are faster but may deviate from the target level. We examine two possible values for the frequency of adaptation: every iteration and every 10 iterations.

Figure 7 illustrates the effect of each combination between initialization with four different adaptive approaches $\mathrm{a}_{q0}+$ and $\mathrm{a}_{q0}-$, and a frequency of adaptation of 1 and 10 iterations. The two upper plots (Figs. 7(a) and 7(b)) show the evolution of the value of q_0 with respect to the number of iterations. Both plots show that the adaptive strategy that starts with a low value of q_0 ($\mathrm{a}_{q0}-$) quickly converges to a similar curve as the adaptive variant that starts with a high value of q_0. As for the frequency of adaptation, a high frequency (every iteration) allows $\mathrm{a}_{q0}-$ to match faster the settings of $\mathrm{a}_{q0}+$. However, there are not strong difference between the two frequencies of adaptation in the case of $\mathrm{a}_{q0}+$. The two bottom plots (Figs. 7(c) and 7(d)) show the evolution of the average distance. Here, the faster frequency of adaptation produces a smoother curve, however, both settings quickly reach the target distance. Therefore, we focus on the results obtained when the adaptation frequency is 10 iterations for a comparison of the performance of the two adaptive strategies, $\mathrm{a}_{q0}+$ and $\mathrm{a}_{q0}-$, the pre-scheduled variation s_{q0} (Section III), and two fixed values $q_0 = 0$ and $q_0 = 0.99$. Figure 8 illustrates this comparison for two TSP instances of size 1 500 and 6 000. The adaptive parameter variation that starts with a low value of q_0 ($\mathrm{a}_{q0}-$) performs worse than the other parameter variations at the start of the algorithm. However, its performance is never worse, and it is at some times better, than that of the default value ($q_0 = 0$). In both instances, the best anytime behavior is obtained by the adaptive $\mathrm{a}_{q0}+$ approach and pre-scheduled parameter variations (s_{q0}). These two strategies have similar anytime behavior, both of them being clearly superior to fixed settings of q_0. The fact that the pre-scheduled variation of q_0 matches the performance of the adaptive variant, suggests that the evolution of the algorithm is smooth in the sense that the optimal adaptation of q_0 can be approximated fairly well with a pre-scheduled variation. This result is also indicated by the smoothness of the curves in the top plots of Fig. 7.

VI. Conclusions

In this article, we have examined the impact pre-scheduled parameter variation can have on the anytime behavior of MMAS. In the application of MMAS to the TSP, pre-scheduled variations of some parameters allow us to obtain a much improved anytime performance when compared to the default MMAS settings. The improvements are particularly impressive when not using local search and for short computation time. Nonetheless, the performance improvement was also quite strong in the case of MMAS+ls. Most interestingly, by rather simple schedules, one could interpolate smoothly between parameter settings of MMAS that result in good solutions quality for short computation times and those that reach high solution quality for high computation times. Hence, for comparisons to MMAS in TSP applications, we strongly recommend in the future the usage of such pre-scheduled parameter variations.

The positive impact of the parameter schedules was less clear in the case of MMAS with local search for the QAP. For many large QAPLIB instances, the pre-scheduled parameter variations did not improve upon optimal fixed parameter settings. One reason may be that local search plays a very



(a) development of q_0, update every 10 iteration.

(b) development of q_0, update every iteration.

(c) distance plot, update every 10 iteration.

(d) distance plot, update every iteration.

Fig. 7. Effect of q_0 starting value when using a distance-based adaptive method to set q_0 in MMAS, instance 1500-1, 2-opt

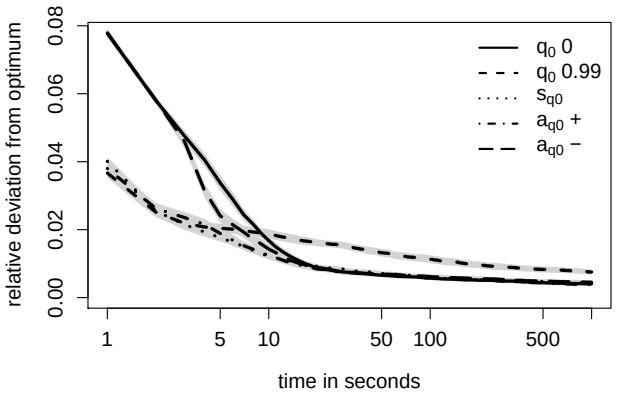
dominant role in the QAP, hiding to a certain extent the effect of parameter variations of the MMAS algorithm.

In the case of the TSP, we also compared the pre-scheduled parameter variations with an informed, adaptive strategy. The adaptive strategy proved to be very effective at finding very good parameter settings during the run. However, such settings may also be reproduced with an appropriate pre-scheduled parameter variation. Hence, the comparison of both strategies does not show a clear advantage of either strategy. Of course, this might not be the case in problems where good parameter settings do not vary smoothly during the run, or when such fine-tuned pre-scheduled variations cannot be found.

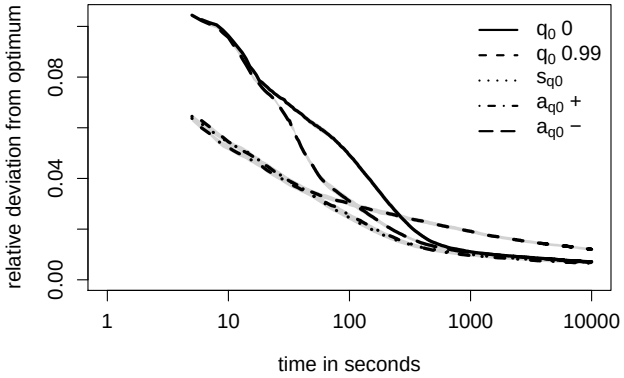
Our study has focused on the parameters m, the number of ants in the ant colony, and q_0, a parameter that determines the greediness of the pseudo-random proportional action choice rule. Pre-scheduling these parameters was shown to be very effective. Combining the pre-scheduled variations of m and q_0 did not result in strong further improvements. Initial results indicate that schedules for other parameter such as β or α can also result in improved anytime behavior [11]. Whether combinations of pre-scheduled parameter variations of these parameters or more than two parameters may result in further performance improvements, is still an open question. A number of other research directions are also worth exploring. First, the parameter variation strategies proposed here have their own parameters, and the robustness of the strategies with respect to these parameters needs to be further assessed. Another interesting aspect is to explore the impact of the heterogeneity of the instances to be tackled on the relative performance of pre-scheduled parameter variations versus adaptive ones. Intuitively, adaptive parameter variations should be more advantageous than pre-scheduled parameter variations for heterogeneous and dynamic problem instances; however, for homogeneous and static instances, and when the main goal is to improve an algorithm's anytime behavior, pre-scheduled parameter variation appears to be sufficiently good.

ACKNOWLEDGMENT

This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Associate. The authors also acknowledge support from the FRFC project "*Méthodes de recherche hybrides pour la résolution de problèmes complexes*".



(a) instance 1500-1: varying q_0, 2-opt



(b) instance 6000-1: varying q_0, 2-opt

Fig. 8. Comparison of fixed, pre-scheduled and adaptive parameter settings of MMAS+ls.

REFERENCES

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

[2] M. Dorigo and L. M. Gambardella, "Ant Colony System: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[3] T. Stützle and H. H. Hoos, "The MAX-MIN Ant System and local search for the traveling salesman problem," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, T. Bäck, Z. Michalewicz, and X. Yao, Eds. Piscataway, NJ: IEEE Press, 1997, pp. 309–314.

[4] ——, "MAX-MIN Ant System," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[5] B. Bullnheimer, R. Hartl, and C. Strauss, "A new rank-based version of the Ant System: A computational study," *Central European Journal for Operations Research and Economics*, vol. 7, no. 1, pp. 25–38, 1999.

[6] T. Stützle and H. H. Hoos, "Improving the Ant System: A detailed report on the MAX-MIN Ant System," FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA–96–12, Aug. 1996.

[7] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. M. Gambardella, "Results of the first international contest on evolutionary optimisation," in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, T. Bäck, T. Fukuda, and Z. Michalewicz, Eds. Piscataway, NJ: IEEE Press, 1996, pp. 611–615.

[8] S. Zilberstein, "Using anytime algorithms in intelligent systems," *AI Magazine*, vol. 17, no. 3, pp. 73–83, 1996.

[9] F. Lobo, C. F. Lima, and Z. Michalewicz, Eds., *Parameter Setting in Evolutionary Algorithms*. Berlin, Germany: Springer, 2007.

[10] R. Battiti, M. Brunato, and F. Mascia, *Reactive Search and Intelligent Optimization*, ser. Operations research/Computer Science Interfaces. Springer, 2008, vol. 45.

[11] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, and M. Dorigo, "Parameter adaptation in ant colony optimization," IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2010-002, Jan. 2010.

[12] T. Stützle, "MAX-MIN Ant System for the quadratic assignment problem," FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA–97–4, Jul. 1997.

[13] T. Stützle and H. H. Hoos, "MAX-MIN Ant System and local search for combinatorial optimization problems," in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I. Osman, and C. Roucairol, Eds. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 137–154.

[14] T. Stützle, "ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem," 2002. [Online]. Available: http://www.aco-metaheuristic.org/aco-code/