$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem

Thomas Stützle and Holger Hoos
Technical University of Darmstadt
Department of Computer Science – Intellectics Group
Alexanderstr. 10, D-64283 Darmstadt, Germany
{tom,hoos}@informatik.th-darmstadt.de

Abstract— **Ant System is a general purpose algorithm inspired by the study of the behavior of Ant Colonies. It is based on a cooperative search paradigm that is applicable to the solution of combinatorial optimization problems. In this paper we introduce $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, an improved version of basic Ant System, and report our results for its application to symmetric and asymmetric instances of the well known Traveling Salesman Problem. We show how $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System can be significantly improved extending it with local search heuristics. Our results clearly show that $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System has the property of effectively guiding the local search heuristics towards promising regions of the search space by generating good initial tours.**

I. Introduction

The Ant System algorithm, originally introduced in [3], [4], is a new cooperative search algorithm inspired by the behavior of real ants. Ants are able to find good solutions to shortest path problems between a food source and their home colony. They communicate via pheromones (aromatic substances) that they use in variable quantities to mark their trails. An ant's tendency to choose a specific path is positively correlated to the intensity of a found trail. The pheromone trail evaporates over time, i.e, it looses intensity if no more pheromone is laid down by other ants. If many ants choose a certain path and lay down pheromones, the intensity of the trails increases and thus this trail attracts more and more ants.

The behavior of ant colonies is imitated to some extent by Ant System by using simple agents, called *ants*, that communicate via a mechanism inspired by the pheromone trails. This search metaphor can be applied to the solution of combinatorial optimization problems [6]. We tested our improved version of Ant System, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS), using symmetric and asymmetric instances of the Traveling Salesman Problem (TSP) taken from TSPLIB [15].[1]

The rest of this paper is organized as follows: To make the paper self-contained we first introduce the TSP and Ant System (Section II). In Section III, we introduce our improved version of Ant System, $\mathcal{MM}$AS, and present some computational results. After showing how $\mathcal{MM}$AS can be significantly improved by adding local search (Section IV) we discuss the positive effect of cooperation among ants in Section IV-C. Finally, in Section V we sum up our experimental results and give an outlook on further work.

Corresponding Author

[1] Accessible via `ftp://ftp.iwr.uni-heidelberg.de/pub/tsplib`.

II. Ant System for TSPs

A TSP can be represented by a complete weighted directed graph $G = (\mathcal{V}, \mathcal{A}, d)$ where $\mathcal{V} = \{1, 2, \ldots, n\}$ is set of nodes (cities), $\mathcal{A} = \{(i,j) \mid (i,j) \in \mathcal{V} \times \mathcal{V}\}$ the set of arcs, and $d : \mathcal{A} \mapsto \mathbb{N}$ a weight function associating a positive integer weight d_{ij} with every arc (i,j).[2] The aim is to find a route of minimal length visiting every city exactly once. For symmetric TSPs, the distances between nodes are independent of the direction, i.e. $d_{ij} = d_{ji}$ for every pair of nodes. In the more general asymmetric TSP (ATSP) at least for one pair of nodes we have $d_{ij} \neq d_{ji}$. The TSP is a $\mathcal{NP}$-hard optimization problem which has many applications and is extensively studied in literature [12], [16]. It also has become a standard testbed for algorithms that try to find near optimal solutions to $\mathcal{NP}$-hard combinatorial optimization problems. This is one more reason to apply our extensions of Ant System to this problem class.

To solve TSPs, Ant System uses pheromone trails τ_{ij} associated with each arc (i,j). Initially, each ant is set on some randomly selected city and begins constructing a valid tour (i.e. a tour visiting every city exactly once) starting from the initial city.[3] A tour is successively built by choosing the next node probabilistically according to a probability distribution proportional to:[4]

$$p_{ij} \sim \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta} \quad \text{if } j \text{ not yet visited, else } 0 \tag{1}$$

where η_{ij} is a local heuristic function which in Ant System is defined as $\eta_{ij} = 1/d_{ij}$. This value is high if the distance between nodes i and j is small. The probability distribution for the selection of the next city is biased by the parameters α and β which determine the relative influence of the trail strength and the heuristic information. To keep track of the cities already visited, every ant maintains a tabu list, in which its actual partial tour is stored. After all ants have constructed a complete tour and have calculated the corresponding tour length L_k, the trail is updated. Every ant is allowed to lay down a total constant quantity Q of pheromone. Thus, the trail intensities are updated according to the formula

[2] $d_{i,j}$ can be interpreted as the distance between nodes i and j.

[3] For a more detailed description of the Ant System algorithm the reader is referred to [6].

[4] To induce a probability distribution, we have to divide the numbers by a common denominator. But this only complicates the formula and distracts the attention from the essential part.

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (2)$$

where ρ is the persistence of the trail (thus, $(1-\rho)$ corresponds to the evaporation) and m is the number of ants. The amount $\Delta\tau_{ij}^{k}$ is equal to Q/L_k if arc (i,j) is used by ant k in its tour, otherwise zero. Thus, frequently used arcs and arcs contained in short tours receive a high amount of pheromone and therefore will be selected more often in future cycles of the algorithm. This can be interpreted as a learning of *good* arcs. Here *good* means that an arc participates often in short tours. The two basic steps *tour construction* according to (1) and *trail update* according to (2) are then simply repeated for a given number of iterations (cycles).[5]

The most important part of Ant System is the treatment of the trail intensities. If some trail intensities are very high, the ants are very likely to choose the next arc among those with a very high trail intensity. In practice, the long term effect of the trail intensities is to successively reduce the size of the effective search space by concentrating the search on a relatively small number of arcs. To characterize the amount of exploration Ant System performs, the λ-branching factor was introduced in [10]. For the value of $\lambda = 0.05$ we found out that if the if the mean 0.05-branching factor is very close to 1, only very few (often only one) arc exiting from a node have a very high selection probability and practically no new solutions are explored. In the following we will refer to such a situation as stagnation of the search.

III. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

The performance of Ant System can be enhanced by allowing only the best ant to update the trails in every cycle. Yet a disadvantage of this strategy is the early stagnation of the search that makes further tour improvements impossible. When stagnation occurs, the trails on few arcs grow so high that the ants will always construct the corresponding tour again and again.

In $\mathcal{MM}$AS we only allow the best ant to update the trails. To alleviate the problems concerning early stagnation, we introduced explicit maximum and minimum trail strengths on the arcs, hence the name $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. The maximum and minimum trail limits are chosen in a problem-dependent way depending on the average arc length (for details we refer to [17]). This way the influence of the trail intensities is limited. As we use as a lower limit $\tau_{\min}$, the probability that a specific arc is chosen may become very small, but will be still greater than zero. The trail limits alleviate the problem associated with the early stagnation of search especially for long runs, thus leading to a higher degree of exploration. The trail strengths in $\mathcal{MM}$AS are initialized to $\tau_{\max}$ for all arcs. After each iteration the evaporation will reduce the trail strength by a factor ρ. Only the trail on arcs participating in the best tours are allowed to increase their intensities or maintain them at a high level. Thus, the trail strength on *bad* arcs decreases slowly and only *good* arcs can maintain a high level of trail strength and will therefore be selected more often by the ants.

[5] We refer to the complete cycle of tour construction and trail update as one iteration, similar to generation in genetic algorithms.

TABLE I

Results on problems of the First International Contest on Evolutionary Computation, 25 runs for each problem. 20.000 iterations for ATSPs, 10000 for symmetric TSPs. `quality` is the percentage deviation form the known optimal solution. `best` the best result obtained and `avg.best` the average tour length over 25 runs.

Problem	`best` (`quality`)	`avg.best`	σ
eil51.tsp	426 (0.0%)	427.2	1.13
kroA100.tsp	21282 (0.0%)	21352.05	50.30
d198.tsp	15960 (1.14%)	16065.95	73.82
ry48p.atsp	14422 (0.0%)	14461.64	39.27
ft70.atsp	38690 (0.04%)	38903.44	149.85
kro124.atsp	36416 (0.50%)	36594.36	156.03
ftv170.atsp	2826 (1.74%)	2836.40	14.91

The performance of $\mathcal{MM}$AS improves considerably over Ant System. Despite of using maximum and minimum trail limits, long runs of the $\mathcal{MM}$AS still can show stagnation behavior. If the mean 0.05-branching factor approaches very low values only few new tours are built, leading to very limited exploration of possibly better tours. To avoid this, we added the *trail-smoothing mechanism*: In case of stagnation of the search as indicated by the mean branching factor, we adjust the trail intensities according to a *proportional update:* The trail intensity is increased proportionally to the difference between $\tau_{\max}$ and the current trail intensity $\tau_{ij}(t)$ on the arc (i,j), i.e.

$$\texttt{increase} \sim \tau_{\max} - \tau_{ij}(t)$$

As an advantage of the proportional update, we do not completely forget the trails learned so far. Its overall effect is that by increasing the trail intensities, the probability distribution for the selection of the next node is influenced in such a way that the exploration of new tours is higher. We call this approach *smoothing* of the trails as the differences between high and low trail intensities become less pronounced, i.e. smoother. With this approach the solution quality for longer runs increased significantly.

To give an indication of the performance with respect to solution quality of $\mathcal{MM}$AS, we present the results of $\mathcal{MM}$AS for some TSP in Table I. The TSPs were taken from the First International Contest on Evolutionary Optimization [1]. For an easier comparison to other existing improved variants of Ant System we also give the results obtained with Ant Colony System (ACS, on of the best existing extension of Ant System [9]), for these problems, see Table II. Problems with extension `atsp` are asymmetric TSP, problems with extension `tsp` are symmetric TSP. The number associated with each instance is the number of cities, except of problem `kro124.atsp` that comprises 100 cities. The parameter settings are $\rho = 0.99$, $\alpha, \eta = 1.0$, and $m = n$ except for `ftv170.atsp`, where we chose $m = n/2$. In all tables we detail the best result obtained for the runs, the aver-

TABLE II

Solution quality for Ant Colony System (ACS) based on 15 independent runs. Table is reproduced from [9].

Problem	best	avg.best	σ
eil51.tsp	426	428.06	2.48
kroA100.tsp	21282	21420	141.72
d198.tsp	15888	16054	71.15
ry48p.atsp	14422	14565.45	115,23
ft70.atsp	38781	39099.05	170,32
kro124p.atsp	36241	36857.00	521,19
ftv170.atsp	2774	2826.47	33,84

TABLE III

Performance of $\mathcal{MM}$AS with additional candidate sets. 20.000 iterations for ATSPs, 10000 for symmetric TSPs. Averages over 25 independent runs.

Problem	best (quality)	avg.best	σ
eil51.tsp	426 (0.0%)	426.7 (0.16%)	0.73
kroA100.tsp	21282 (0.0%)	21302.80 (0.01%)	16.39
d198.tsp	15963 (1.14%)	16048.60 (1.70%)	79.72
att532.tsp	28000 (1.13%)	28194.80 (1.83%)	144.11
ry48p.atsp	14422 (0.0%)	14465.30 (0.30%)	39.27
ft70.atsp	38690 (0.04%)	38913.50 (0.52%)	206.33
kro124.atsp	36416 (0.50%)	36572.85 (0.97%)	137.67
ftv170.atsp	2787 (1.16%)	2807.75 (1.91%)	12.67

age solution and its standard deviation σ. The percentages are the deviation form the known optimal solution. Except for the problems `ftv170.atsp` and `d198.tsp` the average performance of the $\mathcal{MM}$AS achieved over 25 independent runs is better and the standard deviation of the solution quality is generally smaller than the corresponding results for ACS.

One disadvantage of all the above algorithms is the high run time. Basically the algorithm constructs m tours in each iteration and the construction of each tour itself has complexity $O(n^2)$, leading to a total complexity of $O(m \cdot n^2)$ for each iteration of $\mathcal{MM}$AS. As we usually choose m equal to the number of nodes this results in an overall complexity of $O(n^3)$. Therefore, the run time grows rather fast with increasing problem size.

To lower the run time for $\mathcal{MM}$AS, we considered the possibility of adding candidate sets [16], [5]. Besides of reducing computation times by far this modification also has a positive influence on the performance of $\mathcal{MM}$AS. To some extent this can be observed for the larger problems (Table III).[6] The effect of adding candidate sets is still more notable if less iterations are allowed and when time limits are given for the run time of $\mathcal{MM}$AS, see Section IV-B. One should remind, that for many problems the optimal tour can be found within a surprisingly low number of nearest neighbors. So for the problem instance `pcb442.tsp` with 442 cities an optimal solution can be found within a subgraph of the 6 and for `pr2392.tsp` with 2392 cities within a subgraph of the 8 nearest neighbors [16]. Thus, it is plausible, that the addition of candidate sets has a positive influence on the performance of $\mathcal{MM}$AS.

[6]Note, that the average results for $\mathcal{MM}$AS in Table III are all better than the ones obtained for ACS, see Table II. Yet for ACS the best solutions are most often better than for $\mathcal{MM}$AS.

IV. A local search $\mathcal{MM}$AS

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System can be interpreted as a randomized restart procedures that iteratively tries to construct good solution to combinatorial optimization problems. The difference to simple restart algorithms is that several solutions are constructed in parallel and that trail intensities are used as a communication mechanism between the iterations of the restart mechanism biasing the following iterations towards promising regions of the search space. An often used extension to such a kind of algorithm algorithm is the addition of a local search phase to improve the constructed solutions [7]. Local search is also often used to improve the performance of Genetic Algorithms [18], [8] and also has been used in one application for Ant System [13]. The reason for adding local search algorithms to $\mathcal{MM}$AS is twofold: On the one hand, we want to enhance the performance by adding local search, yielding an earlier detection of high quality solutions and to guide the learning mechanism mroe directly. On the other hand, we hope that $\mathcal{MM}$AS is able to construct good initial tours for the following local search phase, such that near optimal tours can be found.

We apply local search to $\mathcal{MM}$AS after every iteration. In this section we especially investigate the issue of how local search should be added to $\mathcal{MM}$AS, especially which ants should be allowed to perform local search. Here we consider basically two possibilities. One in which all ants are allowed to perform local search after each iteration. The other possibility is to allow only the iteration-best ant to improve its current tour by a local search. Additionally one also has to choose the number of ants in $\mathcal{MM}$AS when adding local search. In the experimental investigation we use one version with a constant number of 10 ants and one in which the number of ants is increased proportionally to problem size.[7] When we allow all ants to perform local search, we only use a constant number of 10 ants as for larger problem instances the number of local searches would become too high. We will refer to the different versions as follows: `10 + all-ls` is the version in which 10 ants are used and all ants are allowed to perform a local search, `10 + best-ls` is the version with a constant number of 10 ants and only the best is allowed to perform local search, and $\mathcal{MM}$`AS + ls` is the version in which the number of ants is proportional to problem size and the best ant performs local search.

For symmetric TSPs we implemented the so called 2-opt heuristic which involves the exchange of two edges. For ATSPs, 2-opt is not directly applicable since the direction in which the arcs are traversed has to be considered.[8] Thus, for ATSPs we used one specific 3-opt move that allows the insertion of arcs without reversing the direction of any partial tour. We will refer to this form of 3-opt for ATSP as *reduced 3-opt*. The local search may be implemented using

[7]Note, that in $\mathcal{MM}$AS without local search the number of ants has to be increased proportionally to problem size to yield reasonable performance.

[8]In a 2-opt move one partial tour has to be traversed in the opposite direction. Hence, for ATSPs the length of a partial tours has to be calculated again, leading to high run times.

TABLE IV

$\mathcal{MM}$AS with additional 2-opt for symmetric TSP, with *reduced 3-opt* for ATSP. Three different versions of $\mathcal{MM}$AS and local search are considered. The runs were stopped after $k \cdot n \cdot 100$ steps of $O(n^2)$. The average solution quality is calculated over 10 runs. The best version for a specific problem instance is indicated by bold face numbers.

Problem	`10 + all-ls`	`10 + best-ls`	$\mathcal{MM}$`AS + ls`
kroA100.tsp	21502 (1.03%)	**21427** (0.68%)	21481 (0.94%)
d198.tsp	16197 (2.64%)	**15856** (0.48%)	16056 (1.75%)
lin318.tsp	43677 (3.92%)	**42426** (0.94%)	42934 (2.15%)
pcb442.tsp	53993 (6.33%)	**51794** (2.00%)	52357 (3.11%)
att532.tsp	29235 (5.59%)	**28233** (1.98%)	28571 (3.20%)
rat783.tsp	9576 (8.74%)	**9142** (3.81%)	9171 (4.14%)
p43.atsp	5626 (0.11%)	5626 (0.11%)	**5625** (0.10%)
ry48p.atsp	16318 (13.1%)	**14970** (3.80%)	16308 (13.1%)
ft70.atsp	40278 (4.15%)	**39193** (1.34%)	40127 (3.76%)
kro124p.atsp	44167 (21.9%)	**39201** (8.20%)	43232 (10.3%)
ftv170.atsp	3909 (41.9%)	**2923** (6.10%)	2939 (6.68%)

different pivoting rules. In a best-improvement heuristic the best possible neighborhood move is selected. As especially for larger problems this involves very high computation times, we considered also a first-improvement pivoting rule, in which the first improving move is performed. The results presented for ATSPs are based on the first-improvement version, for symmetric TSPs on the best-improvement version. For symmetric TSPs we additionally restricted the set of possible 2-opt moves to a 35 nearest neighbor subgraph [16].

A. Experimental results

In this section we aim to assess the relative value of the different ways in which local search may be added to $\mathcal{MM}$AS. The results are obtained after a fixed number of steps of complexity $O(n^2)$ as proposed for the Second International Contest on Evolutionary Optimization. The number of steps is limited by $k \cdot n \cdot \text{const}$, where $k = 1$ for symmetric TSP and $k = 2$ for ATSPs, n is the number of cities of the TSP instance and const $\in \{1, 50, 100, 500, 1000, 2500\}$. We report on our results obtained for some values of const, see Table IV to VI. The reason for stopping the runs after several values of const is that according to the limit on the number of steps the relative order of the variants according to their performance may change.[9] To investigate the performance of the different versions of $\mathcal{MM}$AS with local search, we used the problem instances of Section III and additionally also some larger instances of symmetric TSPs. The parameter settings were determined in preliminary runs of the different kinds of algorithm and are chosen considering performance on longer runs of the algorithms. If before the run of an algorithm it is known that the allowed number of steps is rather low, other parameter settings might be better as the ones used here.

For a rather small number of steps allowed, see Table IV, except for problem `p43.atsp`, the version `10 + best-ls` performs best on all problems.[10] If we allow for more

[9] See also Section IV-B when CPU-time limits are given.

[10] Compared to version $\mathcal{MM}$`AS + ls` the rather large differences are also due to the different values for ρ that are best for the two versions for longer runs. For $\mathcal{MM}$`AS + ls` the values for ρ are higher than for `10 + best-ls`, thus learning is slower. We verified that for $\mathcal{MM}$`AS + ls` with smaller values for ρ still performance would be slightly worse.

TABLE V

$\mathcal{MM}$AS with additional 2-opt for symmetric TSP, with *reduced 3-opt* for ATSP. The runs were stopped after $k \cdot n \cdot 500$ steps of $O(n^2)$. We give the average solution quality for the different versions of $\mathcal{MM}$AS with local search.

Problem	`10 + all-ls`	`10 + best-ls`	$\mathcal{MM}$`AS + ls`
kroA100.tsp	**21282** (0.0%)	**21282** (0.0%)	**21282** (0.0%)
d198.tsp	15851 (0.45%)	15829 (0.31%)	**15817** (0.23%)
lin318.tsp	**42159** (0.31%)	42239 (0.50%)	42376 (0.83%)
pcb442.tsp	51274 (0.98%)	51553 (1.53%)	**51265** (0.96%)
att532.tsp	28027 (1.23%)	28103 (1.51%)	**27973** (1.04%)
rat783.tsp	9082 (3.13%)	9096 (3.29%)	**9056** (2.84%)
p43.atsp	**5622.0** (0.04%)	5625.4 (0.10%)	5625.4 (0.10%)
ry48p.atsp	14983 (3.89%)	**14694** (1.89%)	14780 (2.48%)
ft70.atsp	39229 (1.44%)	38919 (0.64%)	**38775** (0.26%)
kro124p.atsp	39881 (10.1%)	37452 (3.37%)	**37391** (3.20%)
ftv170.atsp	2969 (7.77%)	**2828** (2.65%)	2862 (3.88%)

TABLE VI

$\mathcal{MM}$AS with additional 2-opt for symmetric TSP, with *reduced 3-opt* for ATSP. The runs were stopped after $k \cdot n \cdot 2500$ steps of $O(n^2)$ for all ATSPs and the smaller symmetric TSPs. For problem instances `pcb442`, `att532`, and `rat783` the runs were stopped after $n \cdot 1000$ steps. We give the average solution quality for the different versions of $\mathcal{MM}$AS with local search over 10 independent runs.

Problem	`10 + all-ls`	`10 + best-ls`	$\mathcal{MM}$`AS + ls`
kroA100.tsp	**21282** (0.0%)	**21282** (0.0%)	**21282** (0.0%)
d198.tsp	15821 (0.26%)	15816 (0.23%)	**15786** (0.04%)
lin318.tsp	**42070** (0.09%)	42135 (0.25%)	42195 (0.39%)
pcb442.tsp	**51131** (0.69%)	51505 (1.43%)	51212 (0.85%)
att532.tsp	**27871** (0.67%)	28063 (1.36%)	27911 (0.81%)
rat783.tsp	9047 (2.74%)	9085 (3.17%)	**8976** (1.93%)
p43.atsp	**5620.6** (0.01%)	5623.1 (0.06%)	5623.8 (0.07%)
ry48p.atsp	14566 (1.00%)	14559 (0.95%)	**14494** (0.50%)
ft70.atsp	38855 (0.47%)	38830 (0.41%)	**38707** (0.09%)
kro124p.atsp	37415 (3.27%)	36901 (1.85%)	**36655** (1.17%)
ftv170.atsp	2812 (2.07%)	**2790** (1.27%)	2807 (1.89%)

steps, see Table V and Table VI, this changes. In case of ATSPs in 3 out of 5 problem instances the version $\mathcal{MM}$`AS + ls` gives the best average solution quality, see TableVI. For the smallest problem `p43.atsp` the version `all + ls` and for the largest instance `ftv170.tsp` the version `10 + ls` is best. The results for symmetric TSPs are slightly different. Except for the very *easy* to solve problem instance `kroA100.tsp` the version `10 + ls` is never best. Interestingly the version `all + ls` performs best for three problem instances, instances `lin318.tsp,` `pcb442.tsp` and `att532.tsp`. Yet for the largest instances again $\mathcal{MM}$`AS + ls` is best.

It is interesting to have a closer look at possible reasons for the behavior of the algorithms. In case of `10 + all-ls` one specific problem seems to be that many local searches are performed without having any effect on the trails, as only the best local minimum is used to update the trails. In general better initial tours also lead to better local minima. Thus, it seems reasonable to restrict the application of local search to the best initial tours as done for example in version `10 + best-ls`. For short runs of the algo-

TABLE VII

Greedy random tour construction with subsequent local search. Results obtained with $k \cdot n \cdot 2500$ steps of $O(n^2)$ except for `pcb442`, `att532` with $n \cdot 1000$ steps. `quality` refers to the percentage deviation of average solution quality from the known optimal solution over 10 runs. Above for symmetric TSPs, below for ATSPs.

	kroA100	d198	lin318	pcb442	att532
quality	0.0%	1.0%	2.3%	3.5%	3.5%

	p43	ry48p	ft70	kro124p	ftv170
quality	0.01%	6.1%	2.8%	10.9%	15.1%

rithm actually the version `10 + best-ls` is the best possible choice. Yet if longer runs are allowed, this version again in most cases is outperformed by $\mathcal{MM}$`AS + ls` and again `10 + all-ls`. The reason for this might be twofold. In case of $\mathcal{MM}$`AS + ls` it is simply better to choose the best initial tour among a higher number of possible initial tours.[11] In case of `10 + all-ls` for symmetric TSPs it seems that a run needs enough time so that most of the initial tours are rather good initial tours for the subsequent local search. If this is the case, it seems sensible to allow more ants to perform local search. Thus, one idea to improve on the versions proposed here might be to start with few ants and to allow only the iteration-best ant to improve its tour. As high quality solutions are found, one could increase the number of ants and allow more ants to perform local search.

In Table VII we indicate the results obtained with a greedy tour construction heuristic. A greedy tour construction heuristic can be obtained in a straight forward way by setting in Equation 1 $\alpha = 0$ and $\eta > 0$. Note that for $\eta = 0.0$ we obtain a heuristic that constructs random initial tours. We found that a value of $\eta = 5$ produced the best results, by far better then random tour construction or nearest neighbor initial tours. Note that it can be proved that for $\eta \to \infty$ the nearest neighbor tour construction heuristic is obtained. When comparing Table VI to Table VII one can observe that by using variants of $\mathcal{MM}$AS clearly much better results are obtained with the improvement for ATSPs being relatively higher than for symmetric TSPs. Note, that the results presented in Table VII are by far better than the usually used method of comparing one's algorithm to random tour construction with subsequent local search.

B. Performance under time constraints

To give an indication of run times for the algorithm and to further assess the relative value of the different strategies to improve the $\mathcal{MM}$AS, we performed some additional experiments under time constraints, see Table VIII.[12] We compared the solution quality of $\mathcal{MM}$AS without candidate sets ($\mathcal{MM}$AS), $\mathcal{MM}$AS with candidate sets ($\mathcal{MM}$AS + C), and the versions proposed for local search.

As we have seen, $\mathcal{MM}$AS can obtain high quality solutions if enough time is given. For the severe time constraints

[11] 10 tours are possible for `10 + best-ls`, $n/3$ for $\mathcal{MM}$`AS + ls`.

[12] The experiments were performed on Sparc20 Workstations SS20/50.

TABLE VIII

Results for some TSP under time constraints. `time` gives the CPU-time allowed. Here average solution quality is given of 10 independent runs. Allowed CPU times are 30 sec. for `eil51.tsp`, 100 sec. for `kroA100.tsp`, 200 sec. for `d198.tsp` and 500 sec. for `lin318.tsp`.

Variant	eil51	kroA100	d198	lin318
$\mathcal{MM}$AS	502	63070	67910	314182
$\mathcal{MM}$AS + C	446	26127	24703	55170
10 - all+ls	426.2	21284	16225	44670
10 - best+ls	427.3	21283	15992	42994
$\mathcal{MM}$AS + ls	427.5	21290	15945	43158

given here, it has no time to perform enough iterations to learn the trails. This problem is alleviated to some extent by the use of candidate sets as the tours can be constructed faster and more iterations can be made and the candidate sets may be interpreted as a learning aid for $\mathcal{MM}$AS by choosing possibly *good* arcs. A faster detection of good tours for $\mathcal{MM}$AS would be also possible by lower values for ρ or by using higher values for β during the start of the algorithm. Additional local search improved the solution quality for restricted time by far. Yet, as before among the strategies on how to add local search to the $\mathcal{MM}$AS there is no clear winner.

C. Cooperation improves local search

Cooperation among the ants helps the basic Ant System and its variations to significantly improve the performance [6], [17]. Now we will show that this is still true if we add a local search phase to improve the solution of the ants. We consider only the case in which all ants perform local search and a fixed number of 10,000 tour constructions and local searches is allowed. We choose the number of ants m and the number of iterations *iterat* such that $iterat \times m = 10,000$, i.e. we vary the number of ants. Thus, if only one ant is used, it performs local search and updates the trails. If more ants are used, only the ant with the best local minimum is allowed to update the trails. The results in Table IV-C for problem `ry48p.atsp` indicate that also when using local search the cooperation among ants remains helpful. Around 5 ants were already enough to provide very good performance, giving considerably better performance than using only one or two ants. It is interesting to note that the solution quality decreases again if too many ants are used. The reason is that not enough iterations are performed to learn the trails.

If we compare the performance of $\mathcal{MM}$AS with local search to greedy tour construction with a local search phase like in Section IV-A, $\mathcal{MM}$AS with local search performs significantly better. For 10,000 applications of local search we obtained for 10 runs an average tour length of 14870.60, with the best local minimum being 14782. Interestingly, if only one ant is used better results are obtained, see Table IV-C.

TABLE IX

Parameter Settings for the Ant System on ry48p.atsp, Averages over 10 tries, $\rho = 0.9$, $\eta = 1.0$, $\alpha = 1.0$, $\mathcal{MM}$AS with additional 3-opt, 10000 local minima

m	Iterations	best	avg.best	σ
1	10000	14603	14718 (2.05%)	89.52
2	5000	14541	14625 (1.41%)	63.49
5	2000	14422	14514 (0.64%)	55.92
10	1000	14422	14500 (0.54%)	73.88
20	500	14422	14508 (0.60%)	58.41
30	333	14422	14509 (0.60%)	60.24
40	250	14451	14508 (0.60%)	46.60
80	125	14422	14511 (0.62%)	62.57
160	62	14621	14743 (2.23%)	117.48

V. Conclusion and Further Research

In this paper we presented $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, an extension of Ant System introduced in [3], [4]. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System yields significant improvements in performance over Ant System and performs at least at the same level of performance as ACS, in most cases giving a better average performance. A feature shared by $\mathcal{MM}$AS, and ACS is the strategy that only the best ant updates the trails in each iteration of the algorithm. The differences between $\mathcal{MM}$AS and ACS are mainly in the way how a premature stagnation of the search is prevented. $\mathcal{MM}$AS uses bounds on the allowed values for the trail intensities and provides an additional mechanism, smoothing of the trails, to prevent early stagnation. Furthermore, we extended $\mathcal{MM}$AS by adding a local search phase and demonstrated that thereby the solution quality of $\mathcal{MM}$AS can be significantly improved. The combined system outperforms local search with restart by far, which shows that $\mathcal{MM}$AS is very effective in guiding the local search heuristic. Additionally we investigated several possibilities of how to add local search to $\mathcal{MM}$AS. We are only aware of one other work [13], in which Ant System was combined with a local search phase. In other experiments, not reported here, we also could show that $\mathcal{MM}$AS with local search outperforms Ant System with additional local search [17].

Apart from Ant System, many different heuristics have been applied to the TSP. We do not claim that our approach can yet compete with specialized TSP-algorithms like iterated Lin-Kernighan (see [11], [14]). Nevertheless, our results clearly indicate that by using a general purpose heuristic approach like $\mathcal{MM}$AS high quality solutions for the TSP can be obtained. Furthermore $\mathcal{MM}$AS proved to be very useful in guiding the local search algorithms. Still there is a large potential to improve the performance of $\mathcal{MM}$AS plus local search by using more sophisticated local search heuristics for TSP like Lin-Kernighan, see [18], [11], [8], or exploiting more directly specific problem characteristics of Euclidean TSPs [2], [8].

Another interesting issue is the development and study of techniques for automatic adjustment of the most important parameters of $\mathcal{MM}$AS. In this context, evolutionary approaches might be a promising approach. The application of the Ant System is not limited to TSPs (as can be seen e.g. in [6]); therefore we also plan to investigate whether our encouraging results for $\mathcal{MM}$AS on TSP carry over to other problem classes.

All in all, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System— especially when combined with local search heuristics — seems to be a very promising tool providing an adaptive framework for the solution of hard combinatorial optimization problems.

Acknowledgments

We would like to thank Marco Dorigo and the anonymous reviewers for their constructive comments that helped to improve this paper. We also like to thank Thomas Rath for discussions about the topic of this paper and his help.

References

[1] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. Gambardella. Results of the First International Contest on Evolutionary Optimisation. Technical Report TR/IRIDIA/96-18, IRIDIA, Université Libre de Bruxelles, 1996.

[2] K. D. Boese, A. B. Kahng, and S. Muddu. A New Adaptive Multi-Start Technique for Combinatorial Global Optimization. *Operations Research Letters*, 16:101–113, 1994.

[3] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In *Proceedings of ECAL91 - European Conference on Artificial Life*, pages 134–142. Elsevier Publishing, 1991.

[4] M. Dorigo. *Optimization, Learning, and Natural Algorithms.* PhD thesis, Politecnico di Milano, 1992.

[5] M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approachto the Traveling Salesman Problem. Technical Report 96-05, IRIDIA, Université Libre de Bruxelles, 1996.

[6] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[7] T. Feo and M. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[8] B. Freisleben and P. Merz. A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems. In *IEEE Conference on Evolutionary Computation (ICEC'96)*, pages 616–621. IEEE Press, 1996.

[9] L. Gambardella and M. Dorigo. Solving Symmetric and Asymmetric TSPs by Ant Colonies. In *IEEE Conference on Evolutionary Computation (ICEC'96).* IEEE Press, 1996.

[10] L. M. Gambardella and M. Dorigo. Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In *Proceedings of the Twelfth Iternational Conference on Machine Learning*, pages 252–260. Morgan Kaufmann, 1995.

[11] D. S. Johnson. Local Optimization and the Traveling Salesman Problem. In *Proc 17th Colloquium on Automata, Languages, and Programming*, volume 443 of *LNCS*, pages 446–461. Springer Verlag, 1990.

[12] E. Lawler, J. Lenstra, A. R. Kan, and D. Shmoys. *The Traveling Salesman Problem.* John Wiley & Sons, 1985.

[13] V. Maniezzo, M. Dorigo, and A. Colorni. The Ant System Applied to the Quadratic Assignment Problem. Technical Report IRIDIA/94-28, Universit Libre de Bruxelles, Belgium, 1994.

[14] O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov Chains for the TSP Incorporating Local Search Heuristics. *Operations Research Letters*, 11:219–224, 1992.

[15] G. Reinelt. TSPLIB — A Traveling Salesman Problem Library. *ORSA Journal On Computing*, 3:376–384, 1991.

[16] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*, volume 840 of *LNCS.* Springer Verlag, 1994.

[17] T. Stützle and H. Hoos. Improving the Ant-System: A detailed report on the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Technical Report AIDA–96–12, FG Intellektik, TH Darmstadt, Aug. 1996.

[18] N. L. Ulder, E. H. Aarts, H.-J. Bandelt, P. J. van Laarhoven, and E. Pesch. Genetic Local Search Algorithms for the Traveling Salesman Problem. In H.-P. Schwefel and R. Männer, editors, *Proceedings 1st International Workshop on Parallel Problem Solving from Nature*, number 496 in lncs, pages 109–116. sv, 1991.