Improving the Ant System: A Detailed Report on the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

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

— revised version[1] —

Abstract

Ant System is a general purpose heuristic algorithm inspired by the foraging behavior of real ant colonies. Here we introduce an improved version of Ant System, that we called $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. We describe the new features present in $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, make a detailed experimental investigation on the contribution of the design choices to the improved performance and give computational results for the application to symmetric and asymmetric Traveling Salesman Problems. The performance of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System can be further improved by adding a local search phase in which some ants are allowed to improve their solution.

1 Introduction

Ant System (AS) [6, 7, 9] is a new kind of cooperative search algorithm inspired by the foraging behavior of colonies of real ants. The originally blind ants are able to find astonishing good solutions to shortest path problems between food sources and their home colony. The communication between the ants is based on pheromone trails that may be laid down by individual ants. An ant's tendency to choose a specific path is positively correlated to the intensity of a found trail, i.e., the stronger a trail, the higher the probability that an ant will follow that particular path. Yet, the pheromone trail evaporates over time, it looses intensity if no more pheromone is laid down by other ants. If a large number of ants chooses specific ways, the intensity of this trail increases and more ants are likely to choose that specific trail. Ant System imitates the foraging behavior of ants and allows the application of this new search metaphor to the solution of hard combinatorial optimization problems like the Traveling Salesman Problem (TSP), the Quadratic Assignment Problem (QAP) [22] and the job-shop scheduling problem [9].

In this paper we present $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) and investigate its behavior on TSPs. The development of $\mathcal{MM}$AS is motivated by the fact that thew initial form of Ant System gives reasonable good results only on small problem instances and rather poor solution quality on

[1]This revised version of the original Technical Report differs in two main aspects from the earlier version. One is that the upper and lower trail limits are investigated more thoroughly, the other that the results obtained for $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for asymmetric Traveling Salesman Problems with local search are updated. The main reason is that the initial experiments were made with a less powerful implementation of 3-opt leading to significantly worse solution quality.

larger instances compared to other more problem specific methods. Thus, $\mathcal{MM}$AS is designed to improve the performance of Ant System.

The paper is organized as follows. To make the paper self-contained, in Section 2 we introduce the symmetric and asymmetric TSPs, discussing also some simple local search methods that we used to improve the constructed tours. In Section 3 we present basic Ant System applying it to the TSP. Then, we introduce to modifications leading to $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and show how it compares to basic Ant System in terms of solution quality. We also investigate the influence of the parameter settings on the performance of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. In Section 6 we consider the addition of local search procedures to $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. We present different schemes of how local search can be added. The computational results indicate that high quality solutions may be obtained. In general, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System is able to guide the local search efficiently by the construction of good initial tours for the subsequent local search phase. We additionally show that cooperation among the ants helps to improve the performance and finally resume the work and outline some possibilities of further research in 8.

2 The Traveling Salesman Problem

In the Traveling Salesman Problem a given set of n cities has to be traversed so that every city is visited exactly once and the tour ends in the initial city. The optimization goal is to find a shortest possible tour. More formally, a TSP can be represented by a complete weighted directed graph $G = (\mathcal{V}, \mathcal{A}, d)$ with the set of nodes (cities) $\mathcal{V} = \{1, 2, \ldots, n\}$, the set of arcs $\mathcal{A}$, $(i, j) \in \mathcal{V} \times \mathcal{V}$, and the weight function $d : \mathcal{A} \mapsto \mathbb{N}$, associating a positive integer weight d_{ij} with every arc (i, j), giving the inter-city distances. The goal then is to find a shortest Hamiltonian Cycle in the graph. The cost function f is given by the sum of the weights of the arcs that participate in a tour s, i.e. $f(s) = \sum_{(i,j) \in s} d_{ij}$. In the symmetric TSP the distances between nodes are independent of the direction, i.e. for every pair of nodes $d_{ij} = d_{ji}$. In the more general asymmetric TSP (ATSP) at least for one pair of nodes $d_{ij} \neq d_{ji}$. The TSP is a $\mathcal{NP}$-hard optimization problem [14] which has many applications and is extensively studied in literature [27, 20].

Among the algorithmic approaches for the solution of TSPs are elaborate Branch and Cut algorithms that are able to solve instances of hundreds up to few thousand cities to optimality.[2] Despite of this success of complete optimization algorithms still much interest is put on local search heuristics for the the approximate solution of TSPs. One reason is that complete search methods need tight upper bounds for the optimal value. On the other side local search method are still necessary to obtain near optimal solutions for large problems. Another important factor is that the TSP has become a standard test bed for algorithmic ideas. Apart from basic local search procedures, most metaheuristics like Simulated Annealing [17, 16], Tabu Search [19, 21], and Genetic Algorithms [15, 10, 11, 25] have been applied to the TSP. New algorithmic approaches should therefore be tested on this standard benchmark problem and a good performance on TSPs is taken as an indicator of the promise of a new approach. This is one of the main reasons why we apply $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System to this problem. Another reason is that Ant System has been proposed with the example application to the TSP. Furthermore, we want to investigate the behavior of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and we hope that some findings for the application to TSPs also carry over to other combinatorial optimization problems.

[2] The largest instance solved to optimality comprises 7397 cities [1], yet at the cost of several month of CPU-time on modern workstations.

3 Ant System

We briefly resume the application of Ant System to TSPs, for a more detailed description and motivation of Ant System we refer to [9]. When applying Ant System to TSPs, with every arc (i, j) a trail strength, imitating the pheromone, is associated. In Ant System, ants are used to construct tours. Ants are set initially on some starting node and construct a tour through all nodes. In each node the ants are aware of the distance to the neighboring nodes, the trail strength on the arcs, and of the cities they have already visited. Based on this knowledge the ants choose the next city probabilistically. After having constructed a complete tour, each ant is allowed to leave some amount of trail on the arcs it has passed in its tour. The trails in Ant System are used to influence the decisions of the ants in such a way that hopefully good tours may be build. In Figure 1, we resume the algorithm schematically. In the following, we will refer to one iteration of the algorithm as the whole cycle including tour construction and update of trails.

The Ant System Algorithm

Initialization: initial trail intensity = τ_0, place ants on initial cities

Repeat

Construction: Construct for all ants complete tours choosing the next city according to Equation 1. Compute the tour lengths for every ant.

trail-update Update the trails according to Equation 2

Until Termination criterion is met (E.g.,maximal number of iterations)

Figure 1: Schematic presentation of the Ant System algorithm

3.1 Construction of tours

In Ant System, each of m ants constructs a solution to the TSP. As information for the tour construction with each arc (i, j) a heuristic function η_{ij} and a trail strength τ_{ij} are associated. In Ant System the heuristic function $\eta_{ij} = 1/d_{ij}$ is used, it can be interpreted as the desire of choosing node j as the next node. From the current city i the next city j is selected according the following probability distribution

$$p_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \ not \ visited} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}} & if \text{ city } j \text{ is not yet visited} \\ 0 & otherwise \end{cases} \tag{1}$$

The larger the distance d_{ij}, the smaller is the probability to choose next city j. On the other side, the probability to choose city j is positively correlated to the trail strength τ_{ij}. The probability distribution for the selection of the next city is further influenced by the parameters α and β that determine the relative influence of the trail strength and the heuristic information. To keep track of the cities an ant has already visited every ant maintains a tabu list, where its partial tour is stored. Initially, each ant k, $k = 1, 2, \ldots, m$, is set on some randomly chosen node and then constructs a complete tour choosing in each step a city randomly according to Equation 1 until all cites are visited. Finally, the tour length L_k is calculated by ant k.

3.2 Update of pheromone trails

After the tour construction, the pheromone trails are updated. In AS all ants are allowed to lay down a constant quantity Q of pheromone to influence the trail strength. The analogy to the evaporation of the trail is kept by reducing the trails by a fixed factor after every iteration. In Ant System the trail strengths are updated according to the formula

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

where $0 < \rho < 1$ is the persistence of the trail, so $1 - \rho$ represents the evaporation. The amount $\Delta\tau_{ij}^{k} = Q/L_k$ is added if arc (i, j) is visited by ant k.

The trail strength τ_{ij} can be interpreted as an indirect form of a long term memory. Ant System can be interpreted as learning *good* arcs. Often used arcs found in shorter tours receive a higher amount of pheromone. Arcs that are not used very often and are contained in worse tours, only receive a small amount of pheromone and are less likely to be chosen in subsequent iterations.

3.3 Branching factor

The major concern in Ant System is the treatment of the trail intensities. If the trail intensities are very high on some arcs, the ants are very likely to choose the next arc among those, because by Equation 1 the influence of the trails biases the probability distribution towards these arcs. The long term effect of the trails 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, the λ-branching factor, $0 < \lambda < 1$, was introduced in [13]. Its definition is based on the following notion: If for a given node i the concentration of trail on almost all of its leaving arcs becomes very small, the freedom of choice for extending partial solutions from node i is rather limited. Consequently, if this situation simultaneously arises for all nodes of the graph, the part of the search space that is effectively searched by the ants becomes very small.

The λ-branching factor for one node is calculated as follows: If $\tau_{ij}^{\max}$ is the maximal and $\tau_{il}^{\min}$ the minimal trail intensity on arcs exiting from node i, the λ-branching factor is the number of exiting arcs having a trail intensity $\tau > \tau_{il}^{\min} + \lambda \cdot (\tau_{ij}^{\max} - \tau_{ij}^{\min})$. The average λ-branching factor $\bar{\lambda}$ is simply the average of the λ-branching factors of all nodes and gives an indication of the size of the search space explored by the Ant System. If this factor (we usually choose $\lambda = 0.05$) is very close to 1, on average only one arc exiting from a node has a high selection probability. Yet, note that the actual λ-branching factor depends on λ.

3.4 Some observations on the run time of Ant System and $\mathcal{MM}$AS

One disadvantage of Ant System is its high run time. 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 AS. As usually m is chosen 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. The high run time is mainly due to the sequential simulation of an inherently parallel algorithm. The tour construction steps for the individual ants can be done independently and therefore provide an obvious way how to parallelize the algorithm with a high expected speed-up. A large part of the run-times is due to the complexity of the iterations and therefore some effort should be invested in reducing the running times of the construction steps. Here one possibility is to compute for each city a *candidate set* [27] and then to choose the next city only among cities of the candidate set if possible. Only if all the members of the candidate set for a city are already included in the partial tour, other cities not in this set are considered. Using candidate sets drastically reduces the run-time of AS, as the size of the candidate set comprises usually between 10 to 30 cities. In this way a tour can be constructed in $O(n)$, reducing the complexity of one iteration of the algorithm to $O(n^2)$, although with a high constant hidden in the $O(\cdot)$–notation. One possibility for choosing a candidate set is to select a fixed number of nearest neighbors and the cities of the delauney graph, see [27]. Other possibilities to choose a candidate set would be to begin with all cities in the candidate set and then to dynamically restrict the set of cities depending on the trail strength on the arcs. Such a dynamic determination of the candidate set also has a motivation by Ant System itself. Arcs that are chosen very rarely will not receive a high reinforcement and therefore the trail strength will be very low. Thus, the probability that this arc is chosen anyway is very low. This means that we could use $\mathcal{MM}$AS itself to *learn* a candidate set.

3.5 Related Work on Ant System

Ant System was presented first in [6, 7] with the exemple application to TSPs and it was shown that satisfactory solutions to some rather small TSP instances can be obtained. Also several improvements on AS were proposed. In [9] an *elitist* strategy was proposed that is essentially based on additionally reinforcing the arcs belonging to the best tour found in the run of the algorithm. This is realized by adding a quantity $e \cdot Q / L^{best}$ to all arcs belonging to the best-so-far tour whenever the trails are updated. e is the number of elitist ants and L^{best} is the length of the best found tour.

In another work the relation to Reinforcement Learning algorithms [18] is exploited, resulting in the Ant-Q algorithm [13]. Based on this algorithm the same authors proposed Ant Colony System (ACS), presented first in [12]. The main difference of ACS to Ant System are the choice rule for the next city, a local update scheme and that only the global-best ant is allowed to update the trails. Equation 1 is modified to yield the so called *pseudo-random-proportional* rule. In this selection rule, with probability p the arc with maximal value for the product of trail strength times heuristic information is chosen, with probability $1 - p$ the selection is made according to Equation 1. Most notably is the local update rule used in ACS. Every ant modifies the trail on the arc it has chosen at the current stage of the tour construction according to $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$, where τ_0 is a parameter that has to be chosen as a very small constant. The effect of this local updating rule is that a balance between exploration and exploitation can be kept. As an ant chooses an arc, the trail strength on this arc is decreased and therefore is chosen less probably by subsequent ants. The modified update rule in ACS is the same as presented later in Section 4.1 using the global-best tour to update the trails.

Since the first version of this Technical Report another variant of Ant System was proposed, the rank-based version of Ant System [5]. Here, the ω best ants in each iteration and the elitist ants are allowed to modify the trails. The quantity of trail an ant may lay down depends on its rank in the current iteration and the quantity of trail the elitist ant may lay on some arc, see [5] for more details.

4 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

In initial experiments with Ant System we noted that more greediness improves performance finding good solutions much more quickly. Especially for larger problem instances Ant System without some greedy component gives rather poor results.[3] Initially we studied an extension of AS that we called *Generalized elitist strategy.* In this extension only the ants within a factor ϕ of the best solution were allowed to update the trails. This variation was further extended by an adaptive choice of ϕ depending on the λ-branching factor. In another extension ants were only allowed probabilistically to reinforce their tours. The probability that a specific ant is chosen to update the trails depended on its tour length, the better the tour the higher the probability for an ant to be chosen. These variations led to some improvements over AS, but also a major problem was the early stagnation [9] of the search. In such a situation practically no exploration of the search space takes place. In case stagnation of the search occurs, this is also indicated by $\bar{\lambda}$, in case of stagnation of the search search the branching factor is very close to 1. Yet, interestingly, the best solutions are found when search was *nearly* stagnated, i.e., when the branching factor is already rather close to 1.

Thus, it is clear that a better performance of Ant System on TSPs has to include in some form more exploitation of the best solutions found during search and a mechanism to avoid a too early stagnation of the search. This led us to the development of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, which differs in three aspects from Ant System. One aspect is that only one ant is allowed to update the trails in every iteration. To limit the stagnation of the search, a more direct influence on the trail limits is exerted by limiting the allowed range of the possible trail strength by some maximum and minimum trail values τ_{max} and τ_{min}. Additionally we initialize the trails in a specific way, giving the search progress and the trail intensities a specific interpretation. At the first glance, adding limits on the trail strength may seem quite unnatural. But also Ant System is only very loosely coupled to the original behavior of the ants. To us, the main idea for the application of Ant approaches to optimization problems is the use of the trails to have an indirect form of memory on past solutions and to use a reinforcement style of learning of solution attributes. The pheromone trails are used to extract good solution attributes of previously found solutions, solution attributes are, e.g, arcs in case of the TSP. In the next sections we discuss more thoroughly the differences between $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and AS.

4.1 Modified update rule

As detailed above, in $\mathcal{MM}$AS only one ant is used to update the trails in each iteration. Two possibilities to choose this ant may be considered. One, in which after each iteration of the algorithm, the iteration-best ant is allowed to update the trail levels, another in which the best found solution, (global-best tour) is chosen sometimes (or always) to update the trails. These two possibilities differ in the *greediness*, see also Section 5.2.3 for a more explicit discussion of this issue. Thus, update rule 2 is modified to:

$$\tau_{ij}(t) = \rho \cdot \tau_{ij}(t-1) + \Delta\tau_{ij}^{best} \tag{3}$$

where τ_{ij}^{best} is $1/L_{best}$. The tour that is chosen to update the trails may be the tour of the iteration-best ant, referred to as *iteration-best* with the corresponding tour length L_{best}^{iterat}, or the tour corresponding to the best found solution during the run of the algorithm, referred to as a

[3]Note, that initial experiments on Ant System were performed on TSP instances having no more than 75 cities [9], see also experimental results in Section 5.4.

global-best with corresponding tour length L^{global}_{best}. Also intermediate solutions may be applied like by default choosing the iteration-best ant to update the trails and only allow every fixed number of iterations the global-best tour to modify the trails. Using the iteration-best tour favors a higher degree of exploration of new tours, whereas choosing the global-best tour may lead to an early stagnation of the search.

4.2 Trail limits

If an elitist strategy is used in combination with AS, early stagnation of search may result. In this case no better tours are found by the ants and usually the best found tour is constructed by most of the ants. Due to selection rule 1 this is the case if the trail strengths on only one arc exiting from a city are so high that nearly always this arc will be chosen. To limit the influence of the trail strength and to have a more direct influence on the trail strength, we introduce explicit limits on the allowable trail strength by the two parameters τ_{max} and τ_{min}. Thus, for all arcs the trail strength has to be $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$. By limiting the range of values for the trail strength, the extreme values for the selection probability of arcs can be influenced. On the other side, the best solutions during the search are found when $\bar{\lambda}$ is already very low or, in other words, shortly before stagnation occurs. Thus the values for the upper and lower trail limit should take into account both issues, avoidance of stagnation and not inhibiting a concentration on a small subset of the arcs, where good solutions may be preferably found.

In preliminary experiments we found that especially the lower trail limits are more important. We propose to choose specific values for τ_{max} and τ_{min} according to the problem at hand. For the maximal trail strength we propose to set it to an estimation of its analytically maximal value. Due to the trail update as given by Equation 3 the analytically maximal trail strength can be calculated by a geometric series as

$$\tau^{theo}_{max} = \frac{1}{1-\rho} \cdot \frac{1}{L_{opt}} \tag{4}$$

where L_{opt} is the optimal tour length for the TSP instance. If the best found solution is L^{global}_{best}, then L_{opt} is substituted by L^{global}_{best} in Equation 4. Thus, the upper trail limit is adapted during the run of the algorithm to the problem instance at hand. Before the first iteration the trail strength on all arcs is set to some high value to ensure that after the first iteration the trails correspond to τ_{max}, see also the next section.

Still reasonable values for τ_{min} have to be determined. We choose these values according to the following considerations:

- The best tours are found shortly before stagnation of the search occurs. In such a case the probability that the best found tour is constructed in one iteration is significantly higher than 0. Better tours may be found near the best found tour.
- The main influence on the tour construction is determined by the relation between upper and lower trail limits.

For the following considerations we neglect the heuristic information. Based on some further simplifications we propose an analytical way of determining appropriate lower trail limits, instead of using hand-crafted values. Analytically good values for the trail limits can be found by relating convergence (and stagnation) of the algorithm to the minimum trail limit. We say that a run of

$\mathcal{MM}$AS is converged, if the best found tour is constructed with a probability significantly higher than 0. Say, we choose this probability to a specific value p_{best}. When stagnation occurs, usually the trails on arcs leading to the best solution will correspond to their maximal trail limit τ_{max}, whereas, ideally, on the other arcs the trail strength will correspond to τ_{min} or at least be very close to τ_{min}. So, for the further considerations we assume that on arcs participating in the best found tour have associated the maximally possible trail strength, whereas on all the other arcs the trail strength is equal to the minimum trail strength.

An ant constructs the best found solution, if it makes at each decision point the "right" decision and chooses an arc with maximal τ. With decision point we refer to the city at which the ant is currently in the construction process. This probability depends on the distribution of the trails and its values are indirectly determined by τ_{max} and τ_{min}. We make a further simplification and assume that this probability is constant throughout all decisions an ant has to make, let this probability be p_{dec}. Then, the probability that an ant constructs the best solution again is $p_{dec}^{(n-1)}$, as an ant has to make $n-1$ decisions to construct a tour. By setting

$$p_{dec}^{(n-1)} = p^{best} \tag{5}$$

we can determine p_{dec} as

$$p_{dec} = \sqrt[n-1]{p_{best}} \tag{6}$$

As we have calculated p_{dec}, we can determine appropriate settings for τ_{min}. On average an ant has to choose among $n/2$ cities in case no candidate lists are used or among $cand/2$ cities if candidate lists of size $cand$ are used. On average an ant has to choose among avg cities.[4] Now we adjust the lower trail limit $\tau_{\min}$ in such a way that[5]

$$\frac{\tau_{\max}}{\tau_{\max} + avg \cdot \tau_{\min}} = p_{dec} \tag{7}$$

Solving this equation for τ_{min} yields

$$\tau_{\min} = \frac{\tau_{\max} \cdot (1 - p^{dec})}{avg \cdot p^{dec}} \tag{8}$$

Thus, by the above made simplifications, analytically limits for the lower trail strength can be obtained. An additional advantage is that by varying p^{best} a systematic way of investigating the effect of the lower trail limits is provided.

Note, that there is a strong assumption on the problem domain behind thisderivation. It says that around good solutions other good or even better solutions are located. This is definitely the case for the TSP. There is a strong correlation between the tour length of neighboring tours that differ only in few arcs. In general, it has been shown that local optimal tours, i.e., reasonably good tours, are located in a small region of the search space [4] and the average distance among them is rather small. Thus, near the best found tour, there is a reasonably high chance that close to it an even better tour may be found. Yet, it has to be avoided that only the best tour is constructed again and again, thus the probability p_{best} was introduced to describe this "near convergence"

[4]Note, that for the determination of p_{dec} we also made the assumption that it remains constant. This justifies the use of avg here.

[5]Equation 7 is obtained from Equation 1 with the assumption made in this section.

behavior. Thus we conjecture that the above mentioned way of determining the lower trail limits is reasonable for domains in which the search space exhibits a similar structure to the one found in the TSP.

In Section 5.2.2 we will investigate the proposed settings of τ_{min} more thoroughly.

4.3 Initialization of trails

In $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System we initialize the trail strengths as the maximum possible trail strength for all edges. In this way, the search progress in $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System may be interpreted in the following way: After each iteration of the algorithm the trail strength will be reduced due to evaporation to $\tau_{ij}(t) = \rho \cdot \tau_{ij}(t-1)$. As only the best ant is allowed to update its tour, only the trails of arcs participating in the best tours are allowed to increase their intensities or maintain them at the upper trail level. Thus, arcs that do not receive any or very rare reinforcement will continuously lower their trail strength and be selected more rarely by the ants. In this sense, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System tries to avoid the errors made in the past. An error is associated with choosing arcs that lead to rather bad tours, call them *bad* arcs. So, the trail strength on *bad* arcs sinks down slowly and only *good* arcs can maintain a higher level of trail strength. An additional advantage is that the parameter ρ is really interpretable as a learning rate. Higher values for ρ indicate a slower learning of the "good" arcs, as the trail level on the other arcs sinks down more slowly. For higher values of ρ it is easier possible to correct early mistakes for giving reinforcement to certain arcs, as the difference in the trail strength is still rather low. Our understanding of ρ as a learning rate parameter will be underlined in Section 5.2.1.

5 Experimental results for $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

In this section we study the influence of the parameter settings on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. We especially stress the role played by the design choices of $\mathcal{MM}$AS. In section 5.3 we introduce a new mechanism that we called *smoothing of trails* to increase exploration. The effect of this mechanism is to make $\mathcal{MM}$AS and other versions of AS less sensible to specific parameter settings. Finally we compare the performance of $\mathcal{MM}$AS to results obtained by elitist AS and to Ant Colony System based on some longer runs.

5.1 Parameter Settings

As far as not more details are given, we choose the following parameter settings. Q is always chosen as $Q = 1$, the value of Q does not have a significant influence. The influence of the trail is chosen as $\alpha = 1$ and the influence of the heuristic information as $\beta = 1$ for ATSPs and $\beta = 2$ for symmetric TSPs. ρ is chosen $\rho = 0.98$, corresponding to a rather slow "learning". The trail limits were chosen as proposed in Section 4.2 setting $p_{best} = 0.05$. The number of ants is $m = n$, ants are randomly set on some initial city, and the trails are initialized to 1. The candidate list size is chosen as $cand = 20$.

5.2 Experiments with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System on ...

First, experiments on the parameter ρ and its influence on the final solution quality vs. number of allowed tour constructions is investigated. Next, the influence of the lower trail limits and the influence of which ant to choose for update is examined. Finally some more issues on the choice of appropriate values for α and β, and on the need for elitism are performed. For all the following experiments we usually allow a fixed number of tour constructions.

The benchmark instances we use in our experiments were all taken from TSPLIB [26]. We applied $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System to symmetric TSPs as well as to the more general ATSPs. For all the benchmark problems we used, the optimal solution value is known and the results are presented most often as excess over the known optimal solution. We will refer to the benchmark instances by their denominator, the number in the instance denominator indicates the number of cities. The only exception is problem instance **kro124p.atsp** that has 100 cities.

5.2.1 ... parameter values for ρ

To examine the influence of the "speed of learning", as indirectly determined by the parameter ρ, we present results obtained on two problems, **ft70.atsp** and **kroA100.tsp** for different settings of the allowed number of tour constructions and different settings of ρ. The number of tour constructions were chosen as $250 \cdot n$ and $2500 \cdot n$, respectively. The possible settings of ρ were chosen from the set $\{0.8,\ 0.9,\ 0.925,\ 0.95,\ 0.97,\ 0.98,\ 0.99\}$.

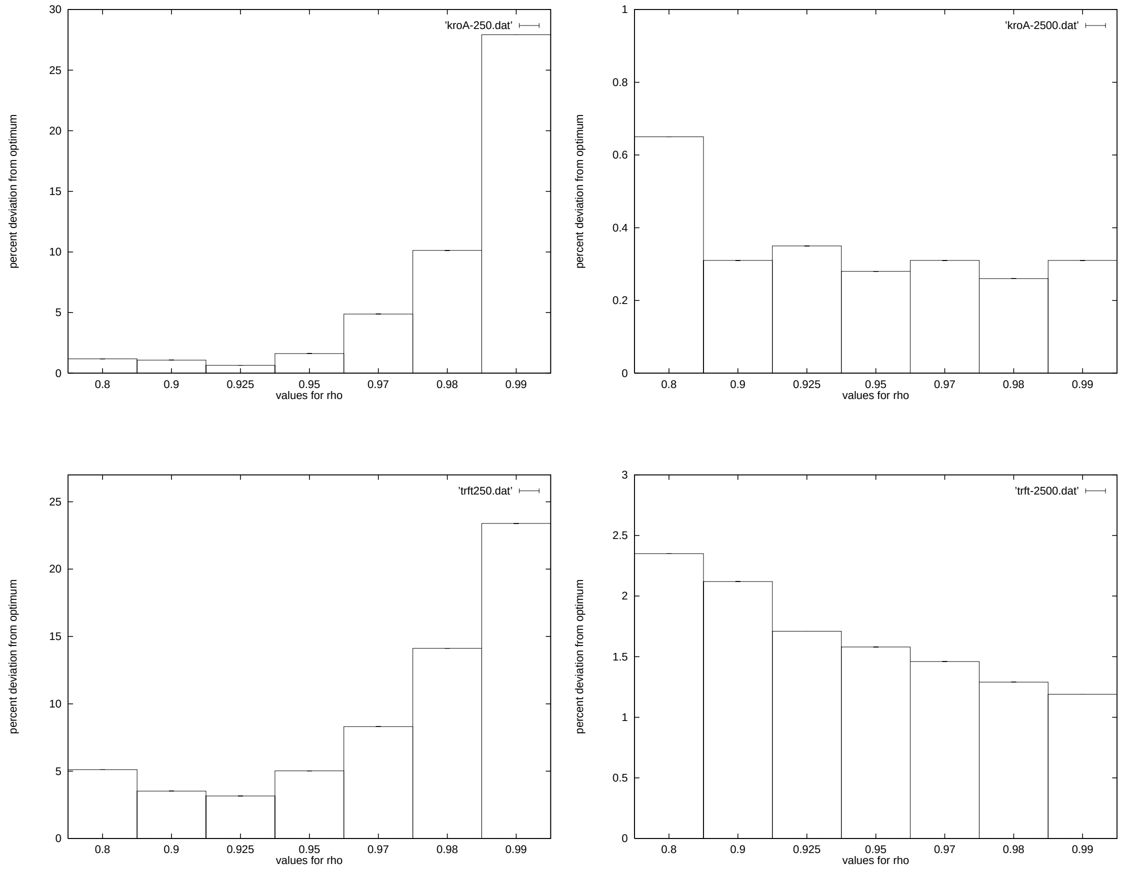


Figure 2: Influence of parameter ρ depending on the maximally allowed number of tour constructions in problem instances kroA100.tsp (upper part) and ft70.atsp (lower part). The number of tour constructions is $250 \cdot n$ (left side) and $2500 \cdot n$ (right side), respectively. Note the different scales on the y-axis.

From the results presented in Figure 2 one can see that with a higher number of tour constructions generally the average performance increases for all values of ρ. This is mainly due to the positive effect of the lower trail limits, see also next section. If only few tour constructions are allowed, lower values for ρ lead to a faster convergence of the algorithm and therefore also to better results. For high values of ρ too few iterations are performed so that not enough time is given to learn which arcs should preferably be used. On the contrary, for a higher number of tour constructions, it pays off to use a higher value for ρ, making learning slower and leading to a higher exploration of the search space especially in the beginning of the run of the algorithm. This can be noted also at hand the development of the average branching factor during a run. Usually the best solutions in a run of the algorithm are found at a rather low branching factor. If only few iterations are performed and ρ is rather high, the branching factor typically is still much higher than 1, so the quality of the tours found is rather poor. More iterations have to be allowed for the algorithm to find good tours. Thus, $\bar{\lambda}$ can be taken as an indicator of the convergence of the algorithm. Yet, one should remember that the specific average branching factor depends strongly on λ and the average branching factor at which the best tours are found, for a fixed λ, depends on the problem size. This observation can be explained as follows: The number of nodes at which still an exploration of tours is performed, may be estimated by $(\bar{\lambda} - 1) \cdot n$. To find better tours near convergence of the algorithm, the number of nodes were still exploration is taking place seems to be roughly a constant number, independent of problem size. Therefore, the larger n the less, on average, is $\bar{\lambda}$ at which good solutions are found.

5.2.2 ... lower trail limits

In this section we investigate the usefulness of the lower trail limits proposed in Section 4.2. We compare computational results using the proposed value for τ_{min} vs. $\mathcal{MM}$AS without using lower trail limits. The number of tour constructions was chosen for each problem instance in such a way that the runs converged. The maximally allowed number of tour constructions was set to $2500 \cdot n$, yet for most problem instances the runs were stopped earlier. The parameter ρ was set to $\rho = 0.98$, resulting in a rather slow learning.

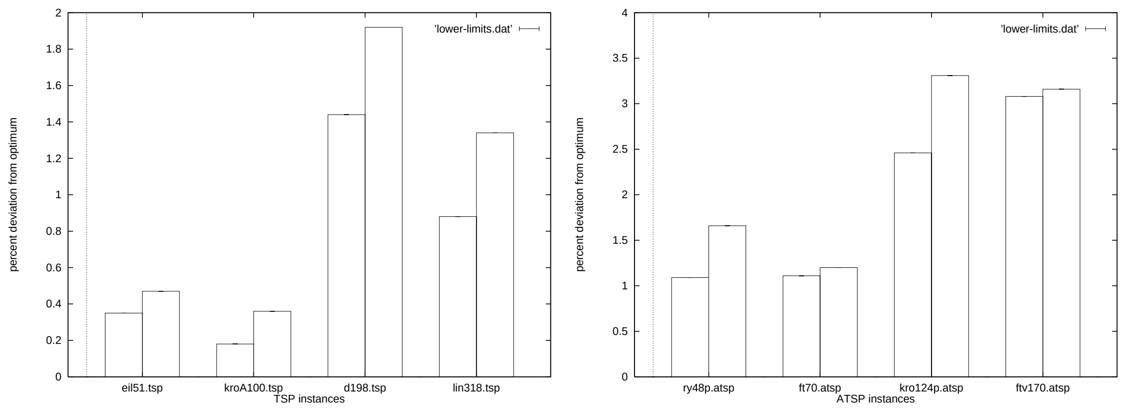


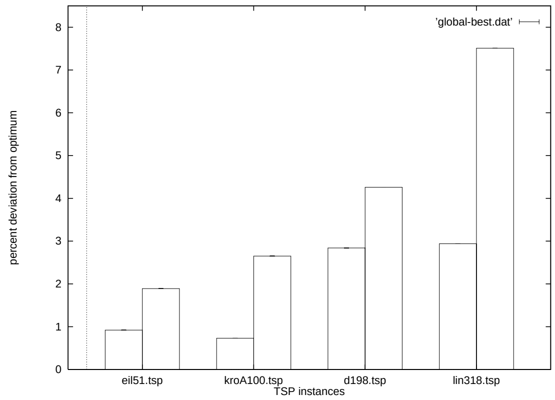
Figure 3: Influence of lower trail limits on tour length. On the left side symmetric TSPs, ATSPs on the right. For each problem instance the left column presents average results over 25 runs for $\mathcal{MM}$AS with lower trail limits, the right column for $\mathcal{MM}$AS without lower trail limits.

As can be seen in Figure 3, for all problems it is advantageous to use the lower trail limits, although for some problem instances the differences are not statistically significant. Also the proposed values seem to provide a good performance. On the other side, if no lower trail limits are used, still reasonably good results can be obtained. Thus, also good part of the performance of $\mathcal{MM}$AS is due to the idea of a slow learning that is made possible by using a rather high value for ρ. Recall from the previous section, that good solutions can be found faster by lower values for ρ and still the lower trail limits have a positive effect on the average results, see Figure 2. Thus, the lower trail limits are definitely helpful to provide a higher amount of exploration of the search space.

5.2.3 ... global versus iteration best

Another interesting question is whether after every iteration the iteration-best tour or the global-best tour should be chosen to reinforce the trails. We performed the same experiments as in the previous section, again exploring the usefulness of the trail limits.

The results presented in Figure 4 have to be compared with those of Figure 3 on the left side, in which the iteration-best ants were chosen to update the trails. Note, that the scales on both figures are different. The results in Section 5.2.2 are significantly better than the ones for the global-best update. Again it can be noted that the lower trail limits help to improve performance significantly. Thus, in general only using the global-best ant to update the trail limits does not seem to be a very good idea for $\mathcal{MM}$AS.[6] Nevertheless, the global-best ant may be used sometimes to reinforce the trails to direct more strongly the search. A main advantage of doing so is that a faster convergence of the algorithm may be achieved.



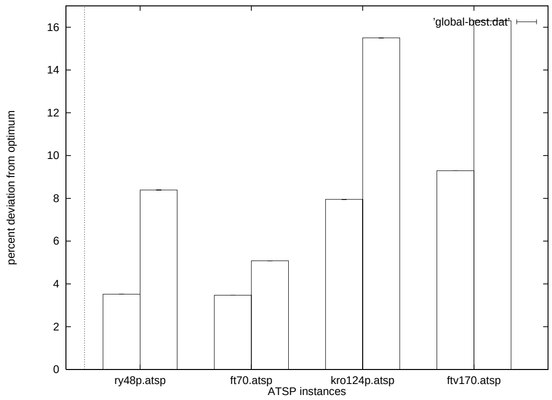


Figure 4: Global-best update rule combined with lower trail limits. On the left results for symmetric TSPs, for ATSPs on the right side. For each problem instance the left column gives results for $\mathcal{MM}$AS with lower trail limits, the right column for $\mathcal{MM}$AS without trail limits. Averages over 25 runs. Note the different scales.

5.2.4 ... elitism versus communism

Another question to investigate is whether in the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System only the best ant (this is the elitist case, call it the *elitist–$\mathcal{MAX}$–$\mathcal{MIN}$* Ant System for the moment) or all ants (this seems to be something like communism – all are the same, let's call it the *communist–*

[6] On the side, other experiments verified that the results with global-best update can be improved by using tighter trails limits, i.e, by setting p^{best} to a lower value. But it never reached the solution quality using iteration-best update.

$\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*) should be allowed to update the trails.[7] The results showed that for communist–$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System the parameter ρ should be chosen much smaller, as the best performance was obtained with values of ρ around 0.35. Note, that communist–$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System corresponds to Ant System with additional upper and lower limits for the trail strength. Yet, for communist–$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System the results (not presented here) are rather poor. Roughly, they are the same as the ones obtained with the basic version of Ant System. Thus, elitism is really necessary to obtain good performance in the TSP domain.

5.2.5 .. Values for α and β

We compared different parameter settings for α and β on several problems. These two parameters determine the mutual influence of the trail strength and the heuristic information on the choice of the next node. As can be observed from the results in Table 5.2.5 the parameter settings of α and β have a considerable influence on the resulting tours. This is especially clear if the average tour lengths are taken into consideration. High values for α have a negative influence on the performance of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. On the other side higher values for β do not seem to have such a strong impact, values of β around 1.0 to 5.0 seem to be appropriate. This confirms earlier results in [9]. It is interesting to observe that the optimal values for β depend on the problem at hand, whereas $\alpha = 1.0$ always seems to be best. If the influence of the trail strength is too high, early convergence hinders the search for better solutions.

Table 1: Influence of α and β on the $\mathcal{MM}$AS for `ry48p.atsp` (upper left side), `ft53.atsp` (upper right side), `ftv47.atsp` (lower left side) and `p43.atsp` (lower right side). Averages over 25 tries, $\rho = 0.99$, 2500 iterations. Best average results are indicated in boldface

β \ α	1.0	2.0	3.5	5.0
1.0	14422 14576.1 101.4	14538 14989.9 291.3	14990 15473.3 270.3	14837 15565.6 331.6
2.0	14422 14522.4 57.5	14606 14820.6 197.4	14612 15123.6 239.2	14764 15111.0 190.5
3.5	14507 14530.0 24.4	14507 14722.6 184.2	14534 14883.5 249.4	14597 14994.6 264.1
5.0	14495 **14517.0** 20.8	14549 14630.6 105.9	14549 14825.9 217.7	14687 14975.8 209.9

β \ α	1.0	2.0	3.5	5.0
1.0	6996 **7081.0** 54.6	7099 7272.4 139.2	7240 7541.9 245.1	7109 7625.8 237.5
2.0	6982 7087.2 57.3	7021 7170.4 78.2	7140 7269.6 81.2	7138 7363.8 138.2
3.5	6984 7099.0 34.7	7037 7181.6 90.2	7043 7276.4 132.7	7056 7338.6 157.2
5.0	7096 7118.2 16.7	7081 7169.2 30.31	7037 7243.2 152.9	7189 7399.7 133.7

5.3 Smoothing of the trails

In the previous sections we have shown that imposing trail limits is helpful to increase solution quality that may be obtained with $\mathcal{MM}$AS. Here we present an additional mechanism that may be useful to increase the performance of elitist versions of AS. For $\mathcal{MM}$AS it may be especially helpful to make the actual choice of the lower trail limits more robust.

[7] Also cases in between could be considered, e.g. allow the ten best ants to update the trails (something like an oligarchy) but we didn't consider this further. Also note that as the ant system is already far from it's natural counterpart no conclusions for "real" ants or even human societies should be drawn from the results presented here.

β \ α	1.0	2.0	3.5	5.0
1.0	1777 1788.2 5.5	1778 1817.6 35.0	1808 1881.0 43.7	1854 1915.9 42.7
2.0	1776 **1781.4** 11.2	1784 1820.5 27.4	1793 1850.2 42.8	1806 1884.6 37.2
3.5	1776 1813.8 33.4	1777 1839.2 23.0	1814 1857.4 25.6	1811 1863.4 30.1
5.0	1777 1828.1 32.7	1777 1815.1 32.9	1816 1853.8 22.0	1777 1868.9 31.8

β \ α	1.0	2.0	3.5	5.0
1.0	5620 **5622.6** 2.3	5621 5624.8 3.7	5621 5228.8 4.7	5621 5628.5 4.8
2.0	5620 5624.3 3.5	5621 5624.7 2.8	5620 5627.0 4.4	5620 5628.5 6.2
3.5	5621 5626.8 2.2	5620 5625.3 2.1	5620 5627.7 3.8	5620 5628.5 3.5
5.0	5626 5629.6 2.0	5623 5628.1 2.9	5621 5631.7 5.8	5625 5634.8 3.9

The mechanism that we called *smoothing of trails* is thought to overcome stagnation of search. It is based on the following observation: If stagnation of search occurs not enough new tours are explored. Thus, we have to adjust the trail intensities in such a way that new tours are explored again to a higher extent. To achieve this aim we propose the following two possibilities:

- *proportional update:* For each edge the trail intensity is increased proportional to the difference between the maximal possible trail intensity and the current trail intensity, i.e.,

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

- *lift-minimum:* In this case simply the minimal trail limit $\tau^*_{\min}$ is increased during one iteration and the trail intensities below this value are set to this increased minimal limit. The minimal limit is set to

$$\tau^*_{\min} = \tau_{\min} + (\text{const} \cdot (\tau_{\max}(t) - \tau_{\min}(t)) \quad \text{with } 0 < \text{const} < 1$$

The basic idea behind this approach is that by increasing the trails either proportionally or up to a constant level, the probability distributions for the selection of the next cities according to Equation 1 is influenced, biasing it towards cities that before had small trail intensities. An additional advantage of this approach is that information, as reflected indirectly by the trail strength, gathered during the run of the algorithm is not completely lost. Note, that in the extreme case the lower trail limit is set to $\tau_{min} = \tau_{max}$ corresponding to a reinitialisation of the trails. In this case still some information of the run is kept by, for example, storing the best found solution so far and using it to reinforce the trails.

Smoothing of the trails is especially interesting if one allows longer runs as in this way a more efficient exploration of the search space can be guaranteed leading to a higher solution quality. The results presented in the following section are based on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System with the proposed trail limits using the proportional update version of smoothing of trails.

5.4 Results for some longer runs

We also applied $\mathcal{MM}$AS to some problem instances proposed for the First International Contest on Evolutionary Optimization [3], allowing for rather long runs. The parameters were set to their default values as detailed in the beginning, except for the two largest instances, $\rho = 0.96$ and $m = n/2$ was chosen to force convergence within the number of tour constructions allowed. In

Table 2: Results for $\mathcal{MM}$AS on benchmark problems for some longer runs, see text for details. We present the best solution found and the average solution quality with their percentage deviation (`quality`) from the known optimal solution. Averages over 25 runs for $n \leq 100$, otherwise averages over 10 runs. On the right side results obtained with ACS reproduced from [8,12] are given, averages over 15 runs.

Problem	$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System			Ant Colony System		
	best (% dev)	avg.best (% dev)	σ	best(% dev)	avg.best (% dev)	σ
ry48p.atsp	14422 (0.0)	14523.4 (0.70)	96.7	14422 (0.0)	14565.4 (0.99)	115.2
ft70.atsp	38679 (0.02)	38922.7 (0.65)	201.5	38781 (0.28)	39099.0 (1.10)	170.3
kro124p.atsp	36230 (0.0)	36573.6 (0.97)	272.1	36241 (0.03)	36857.0 (1.73)	521.9
ftv170.atsp	2761 (0.22)	2817.7 (2.27)	13.9	2774 (0.69)	2826.5 (2.59)	33.8
eil51.tsp	426 (0.0)	427.1 (0.26)	0.9	426 (0.0)	428.1 (0.49)	2.8
kroA100.tsp	21282 (0.0)	21291.6 (0.05)	26.7	21282 (0.0)	21420.0 (0.64)	141.7
d198.tsp	15940 (1.01)	15956.8 (1.12)	14.9	15888 (0.68)	16054.0 (1.74)	71.5
att532.tsp	27971 (1.03)	28194.8 (1.84)	148.3	28147 (1.67)	28522.8 (3.02)	275.7
rat783.tsp	8920 (1.29)	8951.5 (1.65)	18.5	9015 (2.37)	9066.0 (2.95)	28.2

all runs we let in every 10th iteration the global-best tour reinforce the trails. We compare our results to those obtained with Ant Colony System in [12] for the same number of tour constructions and to our implementation of elitist Ant System.[8] In the elitist version of Ant System we used as default parameter settings $\alpha = 1.0$, $\beta = 5.0$ and the number of elitist ants is chosen as $e = n$. We found these parameter values to yield a good performance. Additionally we run one version of elitist Ant System with $\beta = 1.0$ for ATSP and $\beta = 2.0$ for symmetric TSP enhanced by *smoothing of trails*. The total number of tour constructions for all algorithms corresponds to $k \cdot n \cdot 10000$, where $k \in \{1, 2\}$, with 1 for symmetric TSPs and 2 for ATSPs and n is the number of cities of an instance. For the two largest instances `att532` and `rat783` 10^6 tour constructions were allowed, the same as for ACS.

The results in Table 2 and 3 show that with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System a better average performance than with ACS and elitist AS is obtained and the corresponding standard deviations are much smaller, indicating a more predictable performance. Only for one problem instance (`d198.tsp`) the best solution found by ACS or elitist-AS is slightly better than the one with $\mathcal{MM}$AS. The solution quality obtained with the elitist version is, in general, worse than the results obtained with $\mathcal{MM}$AS and ACS. Note, that elitist AS with trail smoothing gives better performance than the other version. The results show the very good performance of $\mathcal{MM}$AS compared to the best so far performing improvement over Ant System.

For AS without elitist strategy the solution quality for these instances is rather poor. So, e.g, with $\alpha = 1.0$, $\beta = 5.0$, $\rho = 0.5$, and $n = 70$ ants for `ft70.atsp` we obtained an average solution quality (over 10 runs) of 39596.3 (2.39%), and for `kro124p.atsp` an average solution quality of 38733.1 (6.91%). For larger problems the solution quality for AS is even worse.

Some runs with the rank-based version of Ant System [5] showed that for most problems some improvements on the elitist Ant System are obtained. Yet, the results with the rank-based version for the long runs are still worse than those obtained with $\mathcal{MM}$AS or ACS. The main advantage of the rank-based version of AS lies in the fast discovery of reasonably good solutions. The rank-based version of AS seems to be a promising approach that should receive further investigation.

[8]Note, that the contest was won by a Genetic Algorithm hybrid that incorporates the Lin-Kernighan local search algorithm to improve the tours of the individuals [11].

Table 3: Results obtained with the *elitist strategy* for Ant System. On the left side results with $\alpha = 1, \beta = 5$, on the right side $alpha = 1, \beta = 1$ for ATSPs and $\alpha = 1, \beta = 2$ for symmetric TSP with additional trail smoothing. Averages over 25 runs for $n \leq 100$, otherwise averages over 10 runs.

Problem	elitist AS, $\beta = 5$			elitist AS + smoothing		
	best (% dev)	avg.best (% dev)	σ	best(% dev)	avg.best (% dev)	σ
ry48p.atsp	14495 (0.51)	14685.2 (1.82)	148.4	14446 (0.17)	14657.9 (1.63)	159.5
ft70.atsp	38820 (0.04)	39261.8 (1.52)	176.9	38719 (0.12)	39161.0 (1.26)	263.9
kro124p.atsp	36928 (1.93)	37510.2 (3.53)	387.1	36416 (0.51)	37417.7 (3.28)	838.1
ftv170.atsp	2845 (3.27)	2952.4 (7.14)	93.8	2822 (2.43)	2908.1 (5.56)	61.5
eil51.tsp	426 (0.0)	428.3 (0.54)	2.2	426 (0.0)	427.4 (0.33)	1.4
kroA100.tsp	21320 (0.18)	21522.8 (1.13)	130.8	21282 (0.0)	21431.9 (0.70)	116.6
d198.tsp	16058 (1.76)	16205.0 (2.69)	96.2	15903 (0.78)	16140.7 (2.28)	151.1

6 Adding Local Search

As shown, it is possible to find near optimal solution to TSPs using only tour constructions with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Yet, it can not compete with more specialized algorithms for the TSP. For larger instances the run time needed to find good solutions is very high and solutions are not guaranteed to be locally optimal. Thus, we add local search procedures to improve the tours constructed by the ants. We do so for mainly two reasons.

- Enhance the performance of $\mathcal{MM}$AS by improving solutions through a local search phase. The goal is a faster convergence of the algorithm and an earlier detection of high quality solutions.
- Use $\mathcal{MM}$AS to construct good initial tours for the following local search phase, such that near optimal tours can be found.

We implemented the *2-opt* heuristic for the symmetric TSPs. In 2-opt, two edges of the current solution are removed and the two resulting partial tours are reconnected by two other edges. Note, that there is only one possible way to reconnect the partial tours. For larger TSPs a straightforward implementation of the local search procedure is computationally too expensive. As $\mathcal{MM}$AS especially for symmetric TSP is run on larger problems we restricted the number of 2-opt moves that are checked to lie in a neighborhood of the 40 nearest neighbors for each city [17] and search for an improving move via a fixed radius nearest neighbor search, as described in [2]. In our current implementation of 2-opt we perform a best-improvement local search. In case of ATSPs the 2-opt heuristic is not directly applicable as the direction in which the cities are traversed is important.[9] Only one specific exchange move of the 3-opt heuristic is applicable such that the direction of traversal is the same as before. The 3-opt procedure for ATSPs was implemented in the most trivial form, as we are not solving very large problem instances. We use a slightly modified first-improvement pivoting rule. We scan for one fixed city the whole neighborhood and choose the best move of this scan, in case improving moves exist.

For the addition of a local search heuristic we used the rather simple *2-opt* heuristic for symmetric TSPs. It is well known that other local search heuristics like *3-opt* or the Lin-Kernighan heuristic

[9] In a 2-opt exchange move one partial path has to be traversed in the opposite direction.

produce a higher tour quality. So, starting from random tours, 2-opt usually results in an average tour length of roughly 8-10% over the optimal solution for the TSPLIB instances, whereas 3-opt averages roughly 5% and Lin-Kernighan 2% over the known optimal tour length [27].[10] Nevertheless, we used 2-opt as it is very easy to implement and gives enough information on the potential of combining $\mathcal{MM}$AS with local search procedures.

6.1 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and local search

In our basic version we added local search to $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System with trail smoothing. One question is which ants should be allowed to perform a local search. Here we consider two options. In one option we use $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System like before with a number of ants proportional to the size of the problem instance, refering to it as $\mathcal{MM}$AS`+ls`, and allow only the iteration-best ant to improve its solution by a local search. In the other option we choose a constant number of 10 ants and every ant is allowed to perform local search, we refer to it as `10+all-ls`. In some preliminary runs appropriate parameter values especially for the second approach were determined. We found values for ρ around 0.8 to 0.9 to give good results. For $\mathcal{MM}$AS`+ls` parameters were chosen slightly different as in 5.4. We choose $\rho = 0.95$ for problems with $n < 400$ and $\rho = 0.925$ otherwise, the number of ants is $m = n/3$. These choices are made to yield a faster convergence of the algorithm. To investigate the influence of the choice of ants for trail update, we allow in one series of experiments only the iteration-best ant to update the trails. In another series of experiments in every second iteration the global-best tour is allowed to reinforce the trails.

For the smaller problem instances we allowed with less than 400 nodes a forth part of the allowed steps in Section 5.4, for the larger symmetric TSP only 1/10th of the maximally allowed number of steps is allowed. Every tour construction and every improving move found by the 2-opt procedure is counted as one step. Thus, one application of 2-opt usually costs more than one step, unless the initial solution is already a local minimum. In case of 3-opt we count every iteration of the first-improvement procedure as n steps.

The results are presented in Tables 4 and 5. The overall best performance is obtained by using only a constant number of 10 ants and use sometimes the global-best ant to reinforce the trails. Having a closer look at the results in Table 5, one notices that using only the iteration-best version is slightly better for smaller problems up to $n = 442$. For larger problems, the global-best ant should sometimes be used to update the trails. If the global-best tour is not used, in some runs very good solutions were found, whereas in others the solution quality is rather poor. This means that for larger problems the iteration-best version of `10+all-ls` does not converge fast enough. Note, that also for $\mathcal{MM}$AS`+ls` using the global-best ant to update the trails increases performance.

Version $\mathcal{MM}$AS`+ls` gives a rather poor performance, as only the best ant is allowed to perform a local search and this ant in later stages of the search often will be already local optimal or only very few steps away from a local optimum. Thus, it should be possible to increase performance by adapting the number of ants that are allowed to apply local search. By increasing the number of local searches, tours that leave more room for improvement are used as starting tours for the local search. We increased the number of local searches dependent on $\bar{\lambda}$. We performed some experiments on the symmetric TSP instances with the following scheme of adapting the number of local searches: In case $\bar{\lambda} > 2.0$, only the best ant is allowed to perform local search. For values of $\bar{\lambda} < 2.0$ we set the number of local searches after each iteration as $(1 - (\bar{\lambda} - 1)^{0.5}) \cdot m + 1$. With this scheme, better results could be obtained for $\mathcal{MM}$AS`+ls`, yet still with `10+all-ls` and occasional

[10] See also [17] for a discussion of several local search approaches for the solution of TSPs.

Table 4: Results for symmetric and asymmetric TSPs for $\mathcal{MM}$AS+1s. On the left side the results for the iteration-best version appear, on the right side results for which every second iteration the global-best tour modifies the trail intensities. Averages over 10 runs.

Problem	$\mathcal{MM}$AS+1s, iteration-best		
	best (% dev)	avg.best (% dev)	σ
d198	15806 (0.16%)	15821.2 (0.26%)	12.7
lin318	42155 (0.30%)	42159.0 (0.31%)	8.4
fl417	11874 (0.11%)	11884.8 (0.20%)	8.4
pcb442	51148 (0.73%)	51189.2 (0.81%)	24.7
d493	35476 (1.35%)	35645.5 (1.84%)	73.4
att532	27841 (0.56%)	27895.2 (0.76%)	52.0
u574	37310 (1.09%)	37500.6 (1.61%)	144.5
rat783	8841 (0.39%)	8950.5 (1.64%)	70.9
p43	5620 (0.0%)	5620.0 (0.0%)	0.0
ry48p	14422 (0.0%)	14429.6 (0.30%)	43.75
ft70	38673 (0.0%)	38684.2 (0.029%)	18.8
kro124p	36230 (0.0%)	36242.8 (0.035%)	32.8
ftv170	2758 (0.11%)	2782.4 (0.99%)	12.7

Problem	$\mathcal{MM}$AS+1s, global-best		
	best (% dev)	avg.best (% dev)	σ
d198	15806 (0.16%)	15821.2 (0.26%)	12.7
lin318	42029 (0.0%)	42130.2 (0.24%)	60.5s
fl417	11865 (0.03%)	11873.0 (0.20%)	6.7
pcb442	50956 (0.35%)	51133.0 (0.70%)	95.7
d493	35212 (0.60%)	35358.5 (1.02%)	66.4
att532	27716 (0.11%)	27829.1 (0.52%)	68.9
u574	37092 (0.51%)	37254.3 (0.95%)	114.4
rat783	8841 (0.39%)	8884.3 (0.89%)	22.9
p43	5620 (0.0%)	5620.0 (0.0%)	0.0
ry48p	14422 (0.0%)	14422.0 (0.0%)	0.0
ft70	38673 (0.0%)	38679.5 (0.020%)	19.4
kro124p	36230 (0.0%)	36237.6 (0.021%)	31.1
ftv170	2755 (0.0%)	2765.5 (0.38%)	10.1

global-best update for most problems is slightly better. Thus, this later approach is preferable for TSPs.

7 Cooperation among Ants is Useful!

For Ant System it has been shown that *cooperation* among ants leads to improved solution quality. It is interesting to see whether the success of cooperation still holds if local search algorithms are added. Here we use the same number of steps as detailed in Section 6, varying the number of ants. Every ant is allowed to improve its tour by a local search phase. To investigate the effect of cooperation, only the iteration-best ant is allowed to update the trail strength. As can be seen from Figure 5, the best results are usually obtained for a small number of roughly 4 to 16 ants.

Note, that we are concerned with a special kind of "cooperation". Basically we are using a competitive learning scheme. As only the iteration-best ant is allowed to reinforce its tour, we have a competition after each iteration among the ants to determine the "fittest" of them. Thus, we have to have a balance among the number of ants in competition and the number of iterations to learn the trail levels for the arcs. For a too large number of ants not enough iterations can be made to learn correct weights, thus the results are similar to those obtained by a randomized construction heuristic only taking the heuristic information into account. If the number of ants in one iteration is too low, especially if we only use one ant, the competition among the ants to select the right ant to update the trails is not high enough and worse tours may reinforce the trails leading to a worse performance. Yet, note that performance with only one ant is usually better than the performance obtained with the randomized construction heuristic. For a randomized tour construction we found that, using the constructive process as used by the ants, $\beta = 5$ is a good value providing reasonably good and diverse starting tours[11]. For the same run time, the resulting

[11] Clearly, for a randomized construction heuristic in Equation 1 we have to set $\alpha = 0$. Note, that the performance using a randomized construction heuristic with local search like proposed here is significantly better than starting from randomly chosen tours and results are better than the best obtained by a nearest neighbor starting tour with subsequent local search. An additional advantage is that many different starting tours can be constructed, especially more than the n possible nearest neighbor tours

Table 5: Results for symmetric and asymmetric TSPs for `10+all-ls`, $m = 10$. On the left side appear the results for the iteration-best version, on the right side results for which every second iteration the global-best tour modifies the trail intensities. Averages over 10 runs.

Problem	`10+all-ls`, iteration-best		
	best (% dev)	avg.best (% dev)	σ
d198	15780 (0.0%)	15780.8 (0.005%)	1.8
lin318	42029 (0.0%)	42044.5 (0.037%)	26.3
fl417	11861 (0.0%)	11861 (0.0%)	0
pcb442	50935 (0.31%)	51070.6 (0.57%)	78.9
d493	35168 (0.47%)	35200.5 (0.57%)	19.2
att532	27714 (0.10%)	27825.4 (0.50%)	22.2
u574	36952 (0.13%)	37189.1 (0.77%)	200.1
rat783	8828 (0.25%)	9005.1 (2.26%)	65.1
p43	5620 (0.0%)	5620.0 (0.0%)	0.0
ry48p	14422 (0.0%)	14422.3 (0.0%)	1.4
ft70	38673 (0.0%)	38675.6 (0.0%)	8.4
kro124p	36230 (0.0%)	36230 (0.0%)	0.0
ftv170	2755 (0.0%)	2760.0 (0.18%)	6.7

Problem	`10+all-ls`, global-best		
	best (% dev)	avg.best (% dev)	σ
d198	15780 (0.0%)	15782.5 (0.016%)	4.7
lin318	42029 (0.17%)	42079.3 (0.12%)	58.1
fl417	11861 (0.0%)	11864.5 (0.029%)	6.7
pcb442	50938 (0.31%)	51091.7 (0.62%)	87.7
d493	35043 (0.12%)	35107.7 (0.30%)	34.5
att532	27722 (0.13%)	27744.4 (0.21%)	16.5
u574	36938 (0.089%)	37043.8 (0.37%)	84.8
rat783	8822 (0.18%)	8851.4 (0.52%)	16.8
p43	5620 (0.0%)	5620.0 (0.0%)	0.0
ry48p	14422 (0.0%)	14422.0 (0.0%)	0.0
ft70	38673 (0.0%)	38673 (0.0%)	0.0
kro124p	36230 (0.0%)	36236.7 (0.018%)	31.2
ftv170	2755 (0.0%)	2760.8 (0.21%)	6.5

solution quality with a randomized starting procedure for ATSP instance **`kro124p.atsp`** was 0.84%, and for ATSP instance **`ft70.atsp`** 0.18%.

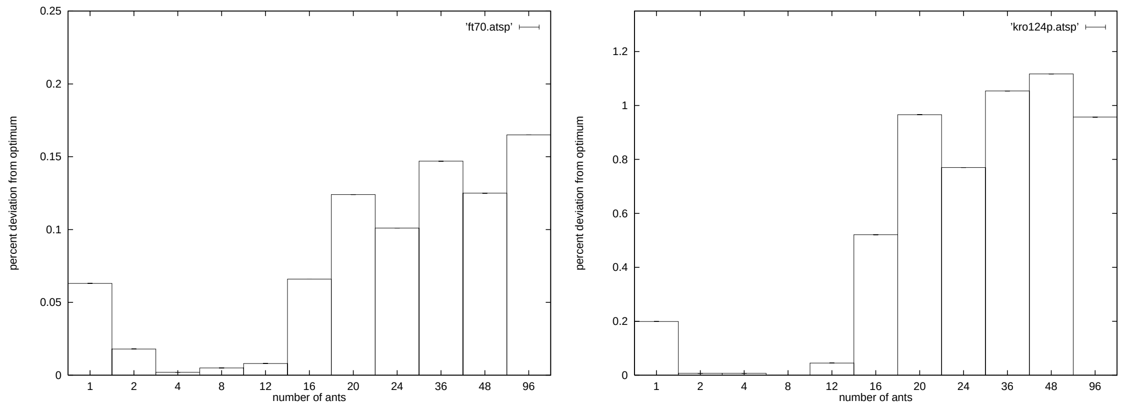


Figure 5: Cooperation among ants for `ft70.atsp` (left side) and `kro124p.atsp` (right side). On the x-axis the number of ants is indicated, the y-axis represents the percentage deviation from the known optimal solution.

8 Conclusion

In this Technical Report we presented $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, discussed its differences to Ant System and investigated the importance of different design issues. One main difference is that for the trail update some form of elitism, like allowing only the best ant to modify the trails, is used. To prevent early stagnation of search, we showed that especially the lower trail limits are effective in increasing exploration of the search space. One of the most important parameters seems to be the parameter ρ. It can be adjusted in such a way that for a given number of cycles the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System yields relatively good results.

Our comparison to Ant Colony System showed, that for rather long runs on the problem instances proposed for the First International Contest on Evolutionary Computation [3] for all tested problem instances a better average solution quality could be obtained. We also could show that the performance of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System can be increased significantly by adding a local search phase, improving the solutions constructed by the ants. So, e.g, the solution quality obtained with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for asymmetric TSP was even higher than that obtained with the genetic algorithm that won the First International Contest on Evolutionary Computation. Yet, our run times are much higher, the main reason being our straightforward implementation of the 3-opt heuristic. For symmetric TSPs very encouraging results could be obtained considering that we only used the 2-opt heuristic as a local search algorithm. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System proved to be very helpful in guiding the local search heuristic by constructing good initial tours, yielding considerable improvements over repeated local search starting from random tours or greedily constructed tours.

Note, that since the first version of this Technical Report, Ant Colony System was extended by adding a local search phase [8]. As a local search procedure they used 3-opt enhanced by a fixed neighborhood search and don't look bits obtaining better results than the ones presented here. Compared to other approaches to solve the TSP, there is still a considerable gap to close. Today, the most efficient heuristics are the large step Markov chain approach presented in [23], its variant, called iterated Lin-Kernighan [16] and the Genetic Local Search approach of [24]. We do not claim that the current form of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System with local search is the best choice for solving TSPs. Our main intent here is to investigate some properties of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and not to improve over the best solution technique for TSPs. We showed that with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System good solutions to instances of the TSP can be obtained. We feel that there is still a large potential for improvements for $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. One, for example, is the addition of more powerful local search methods like the Lin Kernighan heuristic. To support our claims, in [28] it was shown for Genetic Algorithms that with a more powerful local search heuristic better solutions can be obtained within the same time limits than with the more simple 2-opt heuristic used here. Currently we are studying the addition of a fast implementation of 3-opt to $\mathcal{MM}$AS. First results show a greatly improved performance over the results presented here. So, all problem instances with $n < 400$ of TSPLIB can be solved to optimality in a few minutes and for larger problems up to $n = 2000$, the average solution quality lies within 0.1% of the optimal solutions.

Another way to continue our research is the adaptive adjustment of some parameters of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System during the run of the algorithm. The most important parameters to be considered for a dynamic adjustment are β, ρ and the number of ants. With these parameters mainly the speed of convergence of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and the obtainable solution quality can be influenced. In the beginning with higher values for β better tours can be found. Afterwards the influence of the heuristic information should decrease as our experiments with different values for α and β have shown that the best results for rather long runs are obtained when β is around the same level as α. Good solutions can be found more quickly if the global-best tour is used more often, e.g. every second iteration, to reinforce the trails.

In the meanwhile we also applied $\mathcal{MM}$AS to the Quadratic Assignment Problem and the Flow Shop Problem. In these application the positive experiences made by the application to the TSP could be confirmed. As it seems to date, Ant Systems and its extensions are very promising tools for optimization task giving an adaptive framework for the solution of combinatorial problems.

References

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding Cuts in the TSP. Technical Report

95-05, DIMACS, mar 1995.

[2] J.L. Bentley. Fast Algorithms for Geometric Traveling Salesman Problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

[3] 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.

[4] 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.

[5] B. Bullnheimer, Richard F. Hartl, and C. Strauss. A New Rank Based Version of the Ant System — A Computational Study. Technical report, University of Viena, Institute of Management Science, 1997.

[6] 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.

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

[8] 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.

[9] 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.

[10] T. Starkweather et.al. A comparison of genetic sequencing operators. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, Los Altos, CA, 1991. Morgan Kaufmann Puplishers.

[11] 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.

[12] L. Gambardella and M. Dorigo. Solving Symmetric and Asymmetric TSPs by Ant Colonies. In *IEEE Conference on Evolutionary Computation (ICEC'96)*, pages 622–627. IEEE Press, 1996.

[13] 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.

[14] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.

[15] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company Inc., 1989.

[16] 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.

[17] D.S. Johnson and L.A. McGeoch. The Travelling Salesman Problem: A Case Study in Local Optimization. In Emile H.L. Aarts and Jan Karel Lenstra, editors, *to appear in:Local Search in Combinatorial Optimization.* John-Wiley and Sons, Ltd., 1996.

[18] L.P. Kaelbling, M.L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[19] J. Knox. Tabu Search Performance on the Symmetric Traveling Salesman Problem. *Computers & Operations Research*, 21(8):867–876, 1994.

[20] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem.* John Wiley & Sons, 1985.

[21] M. Malek, M. Guruswamy, and M. Pandya. Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 21:59–84, 1989.

[22] 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.

[23] 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.

[24] P. Merz and B. Freisleben. Genetic Local Search for the TSP: New Results. In Thomas Baeck, Zbigniew Michalewicz, and Xin Yao, editors, *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 159–164, 1997.

[25] H. Mühlenbein, M. Gorges-Schleuter, and O.Krämer. Evolution Algorithms in Combinatorial Optimization. *ParallelComputing*, 7:65–85, 1988.

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

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

[28] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. 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.