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

Thomas Stützle [1]

Université Libre de Bruxelles
IRIDIA
Brussels, Belgium

Holger H. Hoos

University of British Columbia
Department of Computer Science
Vancouver, Canada

Abstract

Ant System, the first Ant Colony Optimization algorithm, showed to be a viable method for attacking hard combinatorial optimization problems. Yet, its performance, when compared to more fine-tuned algorithms, was rather poor for large instances of traditional benchmark problems like the Traveling Salesman Problem. To show that Ant Colony Optimization algorithms could be good alternatives to existing algorithms for hard combinatorial optimization problems, recent research in this ares has mainly focused on the development of algorithmic variants which achieve better performance than AS.

In this article, we present $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, an Ant Colony Optimization algorithm derived from Ant System. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System differs from Ant System in several important aspects, whose usefulness we demonstrate by means of an experimental study. Additionally, we relate one of the characteristics specific to $\mathcal{MM}$AS — that of using a greedier search than Ant System — to results from the search space analysis of the combinatorial optimization problems attacked in this paper. Our computational results on the Traveling Salesman Problem and the Quadratic Assignment Problem show that $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System is currently among the best performing algorithms for these problems.

Keywords: Ant Colony Optimisation, Search Space Analysis, Travelling Salesman Problem, Quadratic Assignment Problem, Combinatorial Optimisation.

1 Introduction

Ant Colony Optimization (ACO) [13,8,14,11] is a recently developed, population-based approach which has been successfully applied to several $\mathcal{NP}$-hard combina-

[1] On leave from FG Intellektik, TU Darmstadt, Germany.

torial optimization problems [5,7,12,19,20,29,35,45] (see [10,11] for an overview). As the name suggests, ACO has been inspired by the behavior of real ant colonies, in particular, by their foraging behavior. One of its main ideas is the indirect communication among the individuals of a colony of agents, called *(artificial) ants*, based on an analogy with trails of a chemical substance, called pheromone, which real ants use for communication. The (artificial) pheromone trails are a kind of distributed numeric information (called stigmergic information in [9]) which is modified by the ants to reflect their experience accumulated while solving a particular problem. Recently, the Ant Colony Optimization (ACO) meta-heuristic has been proposed to provide a unifying framework for most applications of ant algorithms [11,10] to combinatorial optimization problems. Algorithms which actually are instantiations of the ACO metaheuristic will be called ACO algorithms in the following.

The first ACO algorithm, called Ant System (AS) [13,8,14], was applied to the Traveling Salesman Problem (TSP). It gave encouraging results, yet its performance was not competitive with state-of-the-art algorithms for the TSP. Therefore, one important focus of research on ACO algorithms has been the introduction of algorithmic improvements to achieve a much better performance. Typically, these improved algorithms have been tested again on the TSP [12,47,6]. While they differ mainly in specific aspects of the search control, all these ACO algorithms are based on a stronger exploitation of the search history to direct the ants' search process. Recent research on the search space characteristics of some combinatorial optimization problems has shown that for many problems there exists a correlation between the solution quality and the distance from very good or optimal solutions [4,3,24,34]. Hence, it seems reasonable to assume that the concentration of the search around the best solutions found during the search is the key aspect that led to the improved performance shown by the modified ACO algorithms.

The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) algorithm discussed in this article achieves a strong exploitation of the search history by allowing only the best solutions to add pheromone during the pheromone trail update. Also, the use of a rather simple mechanism for limiting the strengths of the pheromone trails effectively avoids premature convergence of the search. Finally, $\mathcal{MM}$AS can easily be extended by adding local search algorithms. In fact, the best performing ACO algorithms for many different combinatorial optimization problems improve the solutions generated by the ants with local search algorithms [12,19,47,45,5]. As our empirical results show, $\mathcal{MM}$AS is currently one of the the best performing ACO algorithms for the TSP and the Quadratic Assignment Problem (QAP).

The remainder of this paper is structured as follows. In Section 2, we introduce ACO algorithms and discuss their application to the TSP, using Ant System as a starting point. Next, we review some results from the search space analysis of the TSP which show that solution quality and distance from a global optimum are tightly correlated and we give new results for a similar analysis of the QAP search space. In Section 4 we give details on the modifications of AS leading to $\mathcal{MM}$AS and present an experimental investigation showing the effectiveness of these modifications. Section 5 gives results of our extensive experimental analysis

of $\mathcal{MM}$AS with additional local search for the TSP. In Section 6 we show that $\mathcal{MM}$AS is one of the best available algorithms for the QAP. In the concluding Section 7 we briefly summarize our main results and point out directions for further research.

2 Ant colony optimization

2.1 ACO algorithms

ACO algorithms make use of simple agents called *ants* which iteratively construct candidate solution to a combinatorial optimization problem. The ants' solution construction is guided by (artificial) pheromone trails and problem-dependent heuristic information. In principle, ACO algorithms can be applied to any combinatorial optimization problem by defining *solution components* which the ants use to iteratively construct candidate solutions and on which they may deposit pheromone (see [10,11] for more details). An individual ant constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. We will call each point at which an ant has to decide which solution component to add to its current partial solution a *choice point*. After the solution construction is completed, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in their solution. Typically, solution components which are part of better solutions or are used by many ants will receive a higher amount of pheromone and, hence, will more likely be used by the ants in future iterations of the algorithm. To avoid the search getting stuck, typically before the pheromone trails get reinforced, all pheromone trails are decreased by a factor ρ.

The ants' solutions are not guaranteed to be optimal with respect to local changes and hence may be further improved using local search methods. Based on this observation, the best performing ACO algorithms for many $\mathcal{NP}$-hard *static combinatorial problems*[2] are in fact hybrid algorithms combining probabilistic solution construction by a colony of ants with local search algorithms [12,19,30,45,47,48]. In such hybrid algorithms, the ants can be seen as guiding the local search by constructing promising initial solutions, because ants preferably use solution components which, earlier in the search, have been contained in good locally optimal solutions.

In general, all ACO algorithms for static combinatorial problems follow a specific algorithmic scheme outlined in Figure 1. After the initialization of the pheromone trails and some parameters, a main loop is repeated until a termination condition — which may be a certain number of solution constructions or a given CPU-time limit — is met. In the main loop, first, the ants construct feasible solu-

[2] *Static combinatorial problems* are those in which all relevant problem data are available before the start of the algorithm and do not change during the algorithm's run. An example for the latter case is the network routing problem in communication networks.

tions, then the generated solutions are possibly improved by applying local search, and finally the pheromone trails are updated. It should be noted that the ACO metaheuristic [10,11] is more general than the algorithmic scheme given here. [3]

procedure *ACO algorithm for static combinatorial problems*
  Set parameters, initialize pheromone trails
  while (termination condition not met) **do**
    ConstructSolutions
    ApplyLocalSearch % optional
    UpdateTrails
  end
end

Fig. 1. Algorithmic skeleton for ACO algorithms applied to static combinatorial problems.

2.2 Combinatorial optimization problems

Almost all ACO algorithms have initially been tested on the TSP [13,14,12,47,6]. In this article we focus on the TSP and the QAP as application domains for $\mathcal{MM}$AS.

2.2.1 The Traveling Salesman Problem

The TSP can be represented by a complete graph $G = (N, A)$ with N being the set of nodes, also called cities, and A being the set of arcs fully connecting the nodes. Each arc $(i, j) \in A$ is assigned a value d_{ij} which represents the distance between cities i and j. The TSP then is the problem of finding a shortest closed tour visiting each of the $n = |N|$ nodes of G exactly once. For symmetric TSPs, the distances between the cities are independent of the direction of traversing the arcs, that is, $d_{ij} = d_{ji}$ for every pair of nodes. In the asymmetric TSP (ATSP) at least for one pair of nodes i, j we have $d_{ij} \neq d_{ji}$. All the TSP instances used in the empirical studies presented in this article are taken from the TSPLIB benchmark library accessible at `http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html`. These instances have been used in many other studies and partly stem from practical applications of the TSP.

2.2.2 The Quadratic Assignment Problem

The QAP is the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal is to place the facilities on locations in such a way that the sum of the products between flows and distances is minimal. Given n facilities and n locations, two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{rs}]$, where a_{ij} is the distance between locations

[3] For example, the algorithmic scheme of Figure 1 does not capture the application of ACO algorithms to network routing problems (for an example see [7]).

i and j and b_{rs} is the flow between facilities r and s, the QAP is the problem to minimize

$$f(\phi) = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{\phi(i)\phi(j)} \tag{1}$$

where ϕ is an arbitrary permutation of the set of integers $\{1, \ldots, n\}$ (corresponding to an assignment of facilities to locations), and $\phi(i)$ gives the location of facility i in ϕ. Intuitively, $b_{ij} a_{\phi(i)\phi(j)}$ represents the cost contribution of simultaneously assigning facility i to location $\phi(i)$ and facility j to location $\phi(j)$.

The QAP is an $\mathcal{NP}$-hard optimization problem [41] and it is considered one of the hardest optimization problems. To date, instances of size $n \geq 20$ can generally not be solved to optimality and one has to apply heuristic algorithms which find very high quality solutions in a reltively short computation time. The instances on which we will test $\mathcal{MM}$AS are taken from the QAPLIB benchmark library (accessible at `http://serv1.imm.dtu.dk/~sk/qaplib/`).

2.3 Applying Ant System to the TSP

When applying Ant System (AS) to the TSP, arcs are used as solution components. A pheromone trail $\tau_{ij}(t)$, where t is the iteration counter, is associated with each arc (i, j); these pheromone trails are modified during the run of the algorithm through pheromone trail evaporation and pheromone trail reinforcement by the ants. When applied to symmetric TSP instances, pheromone trails are also symmetric ($\tau_{ij}(t) = \tau_{ji}(t)$) while in applications to asymmetric TSPs (ATSPs) possibly $\tau_{ij}(t) \neq \tau_{ji}(t)$.

2.3.1 Tour Construction

Initially, m ants are placed on m randomly chosen cities. Then, in each construction step, each ant moves, based on a probabilistic decision, to a city it has not yet visited. This probabilistic choice is biased by the pheromone trail $\tau_{ij}(t)$ and by a locally available heuristic information η_{ij}. The latter is a function of the arc length; AS and all other ACO algorithms for the TSP use $\eta_{ij} = 1/d_{ij}$. Ants prefer cities which are close and connected by arcs with a high pheromone trail and in AS an ant k currently located at city i chooses to go to city j with a probability:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \qquad \text{if } j \in \mathcal{N}_i^k \tag{2}$$

where α and β are two parameters which determine the relative importance of the pheromone trail and the heuristic information, and $\mathcal{N}_i^k$ is the feasible neighborhood of ant k, that is, the set of cities which ant k has not visited yet. Each ant k stores the cities visited in its current partial tour in a *list*, that is, each ant has a limited memory which is used to determine $\mathcal{N}_i^k$ in each construction step and thus to guarantee that only valid Hamiltonian cycles are generated. Additionally, it allows the ant to

retrace its tour, once it is completed, so that it can deposit pheromone on the arcs it contains.

2.3.2 Pheromone Update.

After all ants have completed the tour construction, the pheromone trails are updated. This is done first by lowering the pheromone trails by a constant factor (evaporation) and then by allowing the ants to deposit pheromone on the arcs they have visited. In particular, the update follows this rule:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{3}$$

where the parameter ρ (with $0 \leq \rho < 1$) is the trail persistence (thus, $1 - \rho$ models the evaporation) and $\Delta\tau_{ij}^{k}(t)$ is the amount of pheromone ant k puts on the arcs it has used in its tour. The evaporation mechanism helps to avoid unlimited accumulation of the pheromone trails. While an arc is not chosen by the ants, its associated pheromone trail decreases exponentially; this enables the algorithm to "forget" bad choices over time. In AS, $\Delta\tau_{ij}^{k}(t)$ is defined as follows:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) & \text{if arc } (i,j) \text{ is used by ant } k \text{ in iteration } t \\ \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $L^{k}(t)$ is the tour length of the kth ant. By Equation 4, the better the ant's tour is, the more pheromone is received by the arcs belonging to this tour. In general, arcs which are used by many ants and which are contained in shorter tours will receive more pheromone and therefore will more likely be chosen in future iterations of the algorithm. In this sense the amount of pheromone $\tau_{ij}(t)$ represents the learned desirability of choosing the city j to move to when an ant is in city i.

2.4 Applying Ant System to the QAP

The AS application to the TSP can be extended to the QAP in a straightforward way. The main difference is in the definition of the solution components which for the QAP are given by the assignments of facilities to locations. Hence, the pheromone trails $\tau_{ij}(t)$ in the QAP application correspond to the desirability of assigning a facility i to a location j.

For the solution construction, it can be convenient to use a preordering of the facilities (or, equivalently, the locations) and assign facilities in the given order. The decision points are related to the assignments: at each decision point an ant probabilistically decides on which location the next facility should be put. In AS for the QAP, these decisions are done according to Equation 2 using a QAP-specific heuristic information [30]. In this case the feasible neighborhood $\mathcal{N}_i^k$ of ant k comprises those locations which are still free. The single construction steps are repeated until a complete assignment is obtained. The pheromone update is done as in the TSP application.

2.5 Improvements over Ant System

AS has been compared with other general purpose heuristics on some relatively small TSP instances with up to 75 cities. Some initial results were encouraging and have shown the viability of the approach; for example, AS could be shown to achieve better tour qualities than other nature-inspired algorithms, such as Simulated Annealing or Genetic Algorithms [14]. However, for larger TSP instances AS gives a very poor solution quality compared to state-of-the-art algorithms. A first improvement over AS, called the *elitist strategy* for Ant System (AS_e) [8,14], gives a strong additional reinforcement to the solution components belonging to the best solution found since the start of the algorithm; this solution is denoted as s^{gb} (global-best solution) in the following. This is realized by adding a quantity $e/f(s^{gb})$, where e is the number of elitist ants and $f(s^{gb})$ is the solution cost of s^{gb}, to the arcs used in s^{gb} after each iteration. Some limited results presented in [8,14] suggest that the use of the elitist strategy with an appropriate number of elitist ants allows AS to find better tours and to find them earlier in the run. Yet, if too many elitist ants are used, the search concentrates early around suboptimal solutions leading to a premature stagnation of the search. Search stagnation is defined in [14] as the situation where all ants follow the same path and construct the same solution over and over again, such that better solutions cannot be found anymore.

Other improvements over AS include Ant Colony System (ACS) [18,12] and the rank-based version of Ant System (AS_{rank}) [5]. In ACS and $\mathcal{MM}$AS, the best solutions found during the search are exploited by allowing only one ant to update the trails after each iteration, while in AS_{rank} a fixed number of ants of the current iteration – the better the ants are ranked in the current iteration, the more weight they are given for the trail update – and the global-best ant are allowed to update the pheromone trails.

3 Search space characteristics

All improved ACO algorithms have one important feature in common: they exploit the best solutions found during the search much more than what is done by Ant System. Also, they use local search to improve the solutions constructed by the ants. The fact that additional exploitation of the best found solutions provides the key for an improved performance, is certainly related to the shape of the search space of many combinatorial optimization problems. In this section, we report some results on the topology of search spaces of TSP and QAP instances which partly explain the observed performance differences and motivates important aspects of the algorithmic design of $\mathcal{MM}$AS.

3.1 Analysis of Fitness Landscapes

Central to the search space analysis of combinatorial optimization problems is the notion of *fitness landscape* [42,53]. Intuitively, the fitness landscape can be

imagined as a mountainous region with hills, craters, and valleys. A local search algorithm can be pictured as a wanderer that performs a biased walk in this landscape. In a minimization problem such as the TSP or the QAP, the goal is to find the lowest point in this landscape. The effectiveness of a given search strategy for the wanderer strongly depends on the ruggedness of the landscape, the distribution of the valleys and craters in the landscape, and the overall number of valleys and craters. Formally, the fitness landscape is defined by

(1) the set of all possible solutions $\mathcal{S}$;
(2) an objective function that assigns a fitness value $f(s)$ to every $s \in \mathcal{S}$;
(3) a neighborhood structure $\mathcal{N} \subseteq \mathcal{S} \times \mathcal{S}$.

The fitness landscape determines the shape of the search space as encountered by a local search algorithm. The neighborhood structure induces a distance metric on the set of solutions; the distance $d(s, s')$ between two solutions s and s' can be defined as the minimum number of moves that have to be performed to transform s into s'.

The distribution of local minima and their relative location with respect to global optima is an important criterion for the effectiveness of adaptive multi-start algorithms like ACO algorithms. For analyzing this aspect of the fitness landscape, the correlation between solution fitness and the distance to optimal solutions has been studied [3,24,34]; in the literature on genetic algorithms this correlation is also called the fitness-distance correlation (FDC) [24]. This correlation can be captured by the correlation coefficient, which is defined as:

$$\rho(F, D) = \frac{\mathbf{Cov}(F, D)}{\sqrt{\mathbf{Var}(F)} \cdot \sqrt{\mathbf{Var}(D)}} \tag{5}$$

where $\mathbf{Cov}(F, D)$ is the covariance between the random variables F and D which probabilistically describe the fitness and the distance of local optima to a global optimum, while **Var** denotes the variance. This correlation coefficient can be empirically estimated by substituting the covariance and the variance values by the respective empirically measured ones. The FDC analysis has shown to be very useful in the context of studying the effectiveness of adaptive algorithms and their design [3,4,34]. Note that for minimization problems, a high, positive correlation between the solution cost and the distance to the global optimum indicates that the smaller the solution cost, the closer are the solutions – on average – to a global optimum. Hence, if a problem shows a high FDC, algorithms combining adaptive solution generation and local search, may be expected to perform well. For ACO algorithms, this is the case because the most important guidance mechanism of ACO algorithms is the solution quality of the solutions constructed by the ants — the better a solution, the more its solution components will be reinforced. Yet, if no such correlation exists or, even worse, if cost and distance are negatively correlated, the fitness gives only little or no guidance towards better solutions and on such problems ACO algorithm may perform poorly.

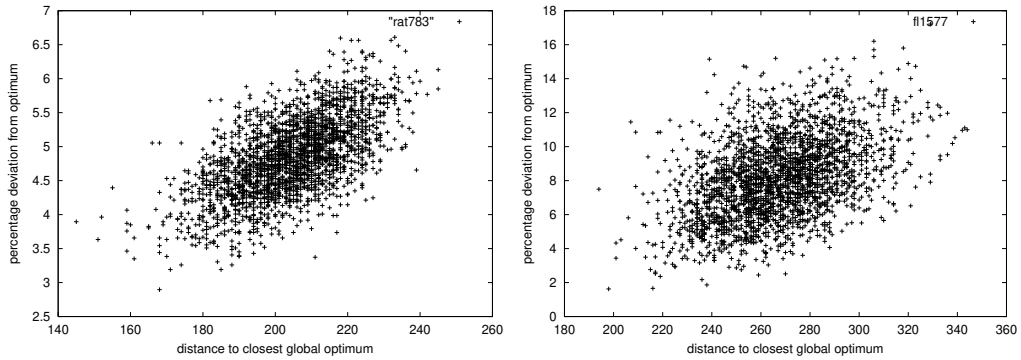

Fig. 2. Fitness-distance plots for symmetric TSP instances. Each of the plots is based on 2500 local optima. The plots are for the instances `rat783` (left) and `fl1577` (right). The x-axis gives the distance to the closest global optimum, the y-axis represents the percentage deviation from the minimal tour length.

3.2 FDC Analysis for the TSP

The symmetric TSP is one of the most widely studied problems in terms of search space analysis [25,37,3,4]. A good distance measure between two tours s and s' is given by the number of different arcs, that is, $d(s,s') = n - |\{(i,j) : (i,j) \in s \wedge (i,j) \in s'\}|$ (where n is the number of cities). A first study of the correlation between the solution quality and the distance to the global optimum has been done in [3]. Additionally, plots of the solution cost versus the distance to the closest global optimum have shown to be a very illustrative tool for the graphical presentation of the cost-distance relationship [3,24,34]. Here, we exemplify results on the FDC analysis using some instances which are larger than previously studied ones.

For our investigation we use a `3-opt` local search algorithm [27]. This local search algorithm proceeds by systematically testing whether the current tour can be improved by replacing at most three arcs. Straightforward `3-opt` implementations require $\mathcal{O}(n^3)$ exchanges to be examined. Since this is too time-consuming in practice, we use a number of standard speed-up techniques [1,32,23] which achieve a sub-quadratical growth of the local search time with instance size. In particular, we restrict the set of examined moves to a candidate list of a fixed number of nearest neighbors; here, as a default we set this number to 40. Additionally, we apply a fixed radius nearest neighbor search [1,23].

For some instances several globally optimal solution exist. To partially address this issue, for each of these problems we generated a number of globally optimal solutions, then eliminated doubles and in our FDC analysis we use the distance to the closest of these global optima. (Note that the number in the instance name gives the number of cities.) Figure 2 gives plots of the percentage deviation from the optimum versus the distance to the closest global optimum. All plots show a strong positive correlation between solution cost and the distance from the closest optimal solution — better local minima tend to be closer to the global optimum. Some summary results for the FDC analysis are given in Table 1, in particular, the average

Table 1
Results of the FDC analysis on symmetric TSP instances based on 2500 `3-opt` local optima. We report the instance name, the average percentage deviation from the optimum (Average (%)), the number of optimal solutions which are used in the FDC analysis (N_{opt}), the average distance between local optima (avg_{d-ls}), the ratio between avg_{d-ls} and the instance size n, the average distance to the closest global optimum (avg_{d-opt}), the ratio avg_{d-opt}/n and the fitness-distance correlation coefficient (ρ_{ls}).

instance	Average (%)	N_{opt}	avg_{d-ls}	avg_{d-ls}/n	avg_{d-opt}	avg_{d-opt}/n	ρ_{ls}
`lin318.tsp`	3.56	1	75.83	0.228	67.25	0.211	0.469
`rat783.tsp`	4.85	119	249.32	0.318	204.24	0.261	0.624
`pcb1173.tsp`	5.91	7	328.93	0.280	274.34	0.234	0.585
`d1291.tsp`	6.96	27	206.27	0.160	159.19	0.123	0.631
`fl1577.tsp`	8.16	27	330.75	0.210	267.25	0.169	0.450
`pr2392.tsp`	5.71	12	660.91	0.276	552.49	0.231	0.538

distance between local minima and to the average distance to global optima, the respective ratios to the maximum possible distance, and the correlation coefficients are given. Interestingly, the ratio between the average distance of tours and the instance size (column avg_{d-ls}/n in Table 1) is very small; this fact indicates that locally optimal tours in the TSP are concentrated around a small region of the whole search space (this particularity has also been observed before [25,37,3,4]). Also, the correlation coefficients are all statistically significant. Note that the generality of these results does not depend on the particular `3-opt` algorithm used. When using `2-opt` or the more powerful Lin-Kernighan heuristic (`LK`) for the local search, again a strongly significant FDC, which was slightly weaker when using `2-opt` and stronger when using `LK`, has been observed for the instances examined in [3].

3.3 FDC Analysis for the QAP

For the QAP it is known that there are several different types of instances and that the particular instance type has a considerable influence on the performance of heuristic methods [50]. According to [50], the instances of QAPLIB which we use in this article can be classified into the following four classes.

- (i) **Unstructured, randomly generated instances.** Instances with the distance and flow matrix entries generated randomly according to a uniform distribution. These instances are among the hardest to solve exactly. Nevertheless, most iterative search methods find solutions within $1-2\%$ from the best known solutions relatively fast [50].
- (ii) **Grid-based distance matrix.** In this class of instances the distance matrix stems from a $n_1 \times n_2$ grid and the distances are defined as the Manhattan distance between grid points. These instances have multiple global optima (at least 4 if $n_1 \neq n_2$ and at least 8 if $n_1 = n_2$) due to the definition of the distance matrices.
- (iii) **Real-life instances.** Instances from this class are instances from practical applications of the QAP. Real-life instances have in common that the flow matrices have many zero entries and the remaining entries are clearly not uniformly distributed.

Table 2
Results of the FDC analysis for QAP instances from the 4 classes defined in this section. Given are the instance identifier (the number in the instance name is the number of facilities), the flow dominance fd(A), the distance dominance dd(B) and the sparsity (*sp*). The remaining entries give summary results of the FDC analysis of the QAP search space. In particular, Average (%) is the average percentage deviation from the best known solution, N_{opt} is the number of optimal solutions used in the FDC analysis, avg_{d-opt} is the average distance to the closest optimum, avg_{d-opt}/n is the ratio between this average instance and the instance dimension, and ρ_{ls} is the correlation coefficient.

instance	$fd(A)$	$fd(B)$	*sp*	Average (%)	N_{opt}	avg_{d-opt}	avg_{d-opt}/n	ρ_{ls}
unstructured, randomly generated (i)								
tai60a	61.41	60.86	0.011	4.71	1	58.82	0.980	0.025
tai80a	59.22	60.38	0.009	3.76	1	78.90	0.986	0.022
Instances with grid-distances (ii)								
nug30	52.75	112.48	0.316	4.18	4	25.93	0.864	0.262
sko56	51.46	110.53	0.305	2.96	4	51.62	0.922	0.254
sko64	51.18	108.38	0.308	2.70	8	58.88	0.92	0.303
real-life instances (iii)								
bur26a	15.09	274.95	0.223	0.32	96	21.12	0.812	0.027
bur26c	15.09	228.40	0.257	0.42	96	22.31	0.858	0.569
els19	52.10	531.02	0.637	36.61	1	16.85	0.887	0.550
kra30a	49.22	149.98	0.6	7.76	257	25.23	0.841	0.251
ste36a	55.65	400.30	0.707	12.01	8	30.98	0.861	0.295
real-life like instances (iv)								
tai60b	76.83	317.82	0.548	7.92	1	56.88	0.948	0.366
tai80b	64.05	323.17	0.552	6.15	1	77.47	0.968	0.150
tai100b	80.42	321.34	0.552	5.34	1	95.23	0.952	0.546

(iv) **Real-life-like instances.** Since the real-life instances in QAPLIB are of a rather small size, a particular type of randomly generated problems has been proposed in [50]. These instances are generated in such a way that the matrix entries resemble the distributions found for real-life problems.

To differentiate between the classes of QAP instances, the flow dominance statistic *fd* can be used. It is defined as:

$$fd(A) = 100 \cdot \frac{\sigma}{\mu}, \text{ where}$$

$$\mu = \frac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \text{ and } \sigma = \sqrt{\frac{1}{n^2 - 1} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - \mu)}$$

A high flow dominance indicates that a large part of the overall flow is exchanged among relatively few items. Randomly generated problems from class (i) have a rather low flow dominance, whereas real-life problems, in general, have a rather high flow dominance. To capture the structure of the distance matrix, a *distance dominance* (*dd*) can be defined analogously. Additionally, real life problems often have sparse flow matrices, hence the *sparsity* of the flow matrix, defined as

$sp = n_0/n^2$, where n_0 is the number of zero matrix entries, can give additional information on the instance type.

Our FDC analysis of the QAP search space uses a `2-opt` algorithm which examines all possible exchanges of pairs of facilities. The distance between solutions ϕ and ϕ' is measured as the number of items placed on different locations, that is, $d(\phi, \phi') = |\{i : \phi_i \neq \phi'_i\}|$. We measure the distance to the optimal solutions if they are available, otherwise we use the best known solutions. Note that for instances of class (ii), (iii), and (iv) with up to 80 items the currently best known solutions are conjectured to be optimal.

For the FDC analysis of the QAP search space one has to take into consideration the fact that many instances have multiple optimal solutions which may, due to symmetries in the distance matrix like in instances of class (ii), be at maximal possible distance n. Hence, on such instances one has to measure the distance to the closest global optimum to get meaningful results. As the exact number of global optima for the QAP instances is not known, we determined a (possibly large) number of optimal (or best known) solutions. The fitness-distance analysis for the QAP is based on 5000 `2-opt` local optima (identical solutions have been eliminated). Some summary results are given in Table 2, where additionally the flow dominance, the distance dominance, and the sparsity of each instance are indicated. Figure 3 shows scatter plots of the fitness-distance correlation for one instance of each problem class.

The fitness-distance analysis shows clear differences between the different problem classes. For class (i), the correlation coefficients are almost zero for all instances. Hence, the solution quality gives only very little guidance and on these instances ACO algorithms can be expected to perform rather poorly. For the other three classes, significant correlations between the solution cost and the distance to an optimal solution exist. The only exception is instance `bur26a` for which the correlation coefficient is not significantly different from zero at the 0.01 level. Note that for instances with a high flow or distance dominance and high sparsity, also a significant FDC can be observed which suggests that these simpler measures may be used as indicators for a high FDC.

In summary, we can conclude that — on average — the better the solution quality the closer a solution is to an optimal solution in real-life QAP instances and also in those of classes (ii) and (iv). These instances show a structure in the following sense: The optimal solutions determine the preferred locations of items. Hence, the more locations for items a solution has in common with an optimal solution, the better will be that solution, on average. As we have argued before, such a significant correlation also indicates the potential usefulness of an ACO approach to the QAP.

Comparing the results of the FDC analysis for the TSP and the QAP one may observe two main differences. First, the ratio between the average distance of the local minima from the closest optimal solution and the instance dimension, for both problems given by n, is much smaller for the TSP than for the QAP. Second, the correlation coefficients for the TSP instances are somewhat larger than for the QAP. Based on these observations we can conclude that local minima in the QAP appear to be spread over large parts of the QAP search space, while for the TSP they are

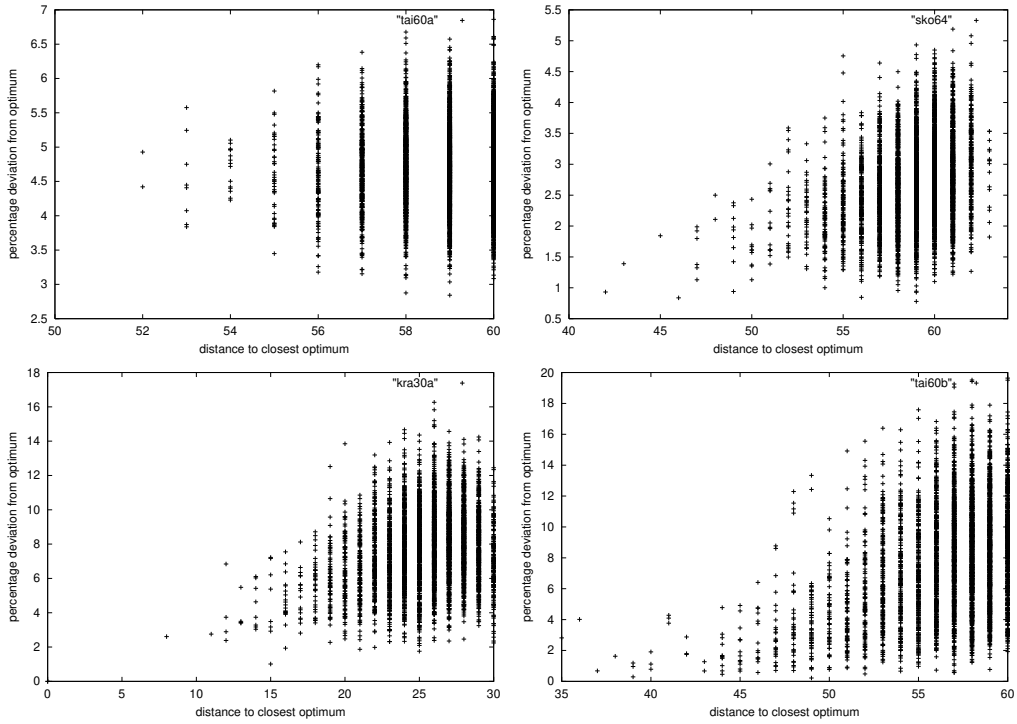

Fig. 3. Fitness-distance plots for four QAP instances, one from each instance class. From top left to bottom right: `tai60a` (class (i)), `sko64` (class (ii)), kra30a (class (iii)), and `tai60b` (class (iv)). The number in the instance name is the number of facilities. The plots show 5000 2-opt solutions for each instance; the x-axes gives the distance to the closest global optimum, while the y-axes indicates the absolute solution cost.

concentrated on a relatively small subspace; also, the solution quality of QAP local minima tends to give somewhat less guidance than for the TSP. Hence, the QAP should be relatively more difficult to solve than the TSP, which is in accordance with the observed hardness of these problems in practice. Additionally, the fact that local minima in the QAP search space are more scattered suggests that in the QAP case effective algorithms need to do a stronger search space exploration than in the TSP case.

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

Research on ACO has shown that improved performance may be obtained by a stronger exploitation of the best solutions found during the search and the search space analysis in the previous section gives an explanation of this fact. Yet, using a greedier search potentially aggravates the problem of premature stagnation of the search. Therefore, the key to achieve best performance of ACO algorithms is to combine an improved exploitation of the best solutions found during the search with an effective mechanism for avoiding early search stagnation. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, which has been specifically developed to meet these requirements, differs in three key aspects from AS.

(i) To exploit the best solutions found during an iteration or during the run of the algorithm, after each iteration only one single ant adds pheromone. This ant may be the one which found the best solution in the current iteration (*iteration-best* ant) or the one which found the best solution from the beginning of the trial (*global-best* ant).

(ii) To avoid stagnation of the search the range of possible pheromone trails on each solution component is limited to an interval $[\tau_{min}, \tau_{max}]$.

(iii) Additionally, we deliberately initialize the pheromone trails to τ_{max}, achieving in this way a higher exploration of solutions at the start of the algorithm.

In the next sections we discuss the differences between $\mathcal{MM}$AS and AS in more detail and report computational results which demonstrate the effectiveness of the introduced modifications in improving the performance of the algorithm.

4.1 Pheromone trail updating

In $\mathcal{MM}$AS only one single ant is used to update the pheromone trails after each iteration. Consequently, the modified pheromone trail update rule is given by

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best}, \tag{6}$$

where $\Delta\tau_{ij}^{best} = 1/f(s^{best})$ and $f(s^{best})$ denotes the solution cost of either the iteration-best (s^{ib}) or the global-best solution (s^{gb}). Using one single ant for the pheromone trail update was also proposed in ACS [12]. While in ACS typically only s^{gb} is used (although some limited experiments have also been performed using s^{ib}), $\mathcal{MM}$AS focuses on the use of the iteration-best solutions.

The use of only one solution, either s^{ib} or s^{gb}, for the pheromone update is the most important means of search exploitation in $\mathcal{MM}$AS. By this choice, solution elements which frequently occur in the best found solutions get a large reinforcement. Still, a judicious choice between the iteration-best and global-best ant for updating the pheromone trails controls the way the history of the search is exploited. When using only s^{gb}, the search may concentrate too fast around this solution and the exploration of possibly better ones is limited, with the consequent danger of getting trapped in poor quality solutions. This danger is reduced when s^{ib} is chosen for the pheromone trail update since the iteration-best solutions may differ considerably from iteration to iteration and a larger number of solution components may receive occasional reinforcement. Of course, one can also use mixed strategies like choosing s^{ib} as a default for updating the pheromones and using s^{gb} only every fixed number of iterations. In fact, as we will show later, when using $\mathcal{MM}$AS with local search for solving some of the larger TSP or QAP benchmark instances, the best strategy seems to be the use of a dynamical mixed strategy which increases the frequency of using s^{gb} for the pheromone update during the search (see Section 5 for details).

4.2 Pheromone trail limits

Independent of the choice between the iteration-best and the global-best ant for the pheromone trail update, search stagnation may occur. This can happen if at each choice point, the pheromone trail is significantly higher for one choice than for all the others. In the TSP case, this means that for each city, one of the exiting arcs has a much higher pheromone level than the others. In this situation, due to the probabilistic choice governed by Equation 2, an ant will prefer this solution component over all alternatives and further reinforcement will be given to the solution component in the pheromone trail update. In such a situation the ants construct the same solution over and over again and the exploration of the search space stops.

Obviously, such a stagnation situation should be avoided. One way for achieving this is to influence the probabilities for choosing the next solution component, which depend directly on the pheromone trails and the heuristic information. The heuristic information is typically problem-dependent and static throughout the algorithm run. But by limiting the influence of the pheromone trails one can easily avoid the relative differences between the pheromone trails from becoming too extreme during the run of the algorithm. To achieve this goal, $\mathcal{MM}$AS imposes explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone trails such that for all pheromone trails $\tau_{ij}(t)$, $\tau_{min} \leq \tau_{ij}(t) \leq \tau_{max}$. After each iteration one has to ensure that the pheromone trail respects the limits. If we have $\tau_{ij}(t) > \tau_{max}$, we set $\tau_{ij}(t) = \tau_{max}$; analogously, if $\tau_{ij}(t) < \tau_{min}$, we set $\tau_{ij}(t) = \tau_{min}$. Also note that by enforcing $\tau_{min} > 0$ and if $\eta_{ij} < \infty$ for all solution components, the probability of choosing a specific solution component is never 0.

Still, appropriate values for the pheromone trail limits have to be chosen. In the following we will propose a principled way of determining these values. Yet, first we introduce the notion of *convergence* for $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System which is needed in the following. We say that $\mathcal{MM}$AS has *converged* if for each choice point, one of the solution components has τ_{max} as associated pheromone trail, while all alternative solution components have a pheromone trail τ_{min}. If $\mathcal{MM}$AS has converged, the solution constructed by always choosing the solution component with maximum pheromone trail will typically correspond to the best solution found by the algorithm. The concept of convergence of $\mathcal{MM}$AS differs in one slight but important aspect from the concept of stagnation [14]. While stagnation describes the situation where all ants follow the same path, in convergence situations of $\mathcal{MM}$AS this is not the case due to the use of the pheromone trail limits.

We now refine our concept of convergence by showing that the maximum possible pheromone trail is asymptotically bounded.

Proposition 4.1 *For any τ_{ij} it holds:*

$$\lim_{t \to \infty} \tau_{ij}(t) = \tau_{ij} \leq \tau'_{max} = \frac{1}{1-\rho} \cdot \frac{1}{f(s^{opt})} \tag{7}$$

Proof: The maximum possible amount of pheromone added after any iteration is $1/f(s^{opt})$, where $f(s^{opt})$ is the optimal solution value for a specific problem.

Hence, by Equation 6 the discounted pheromone trail up to iteration t corresponds to

$$\tau_{ij}^{max}(t) = \sum_{i=1}^{t} \rho^{t-i} \cdot \frac{1}{f(s^{opt})} + \rho^t \cdot \tau_{ij}(0)$$

Asymptotically, because $\rho < 1$, this sum converges to

$$\frac{1}{1-\rho} \cdot \frac{1}{f(s^{opt})}$$

In $\mathcal{MM}$AS, we set the maximum pheromone trail τ_{max} to an estimate of the asymptotically maximum value. This is achieved by using $f(s^{gb})$ instead of $f(s^{opt})$ in Equation 7; each time a new best solution is found, τ_{max} is updated, leading actually to a dynamically changing value of $\tau_{max}(t)$.

To determine reasonable values for τ_{min}, we use the following assumptions (the first is based on empirical observations in some initial experiments for the TSP):

(i) The best solutions are found shortly before search stagnation occurs. In such a situation the probability of re-constructing the global-best solution in one algorithm iteration is significantly higher than zero. Better solutions may be found close to the best solution found.

(ii) The main influence on the solution construction is determined by the relative difference between upper and lower pheromone trail limits, rather than by the relative differences of the heuristic information.

Note that the validity of the first assumption depends strongly on the search space characteristics of the problem as discussed in the previous section. It implicitely means that around good solutions there is a reasonable chance to find even better ones. In fact, for the TSP this is true (see also Section 3.2). The second assumption is taken because in the following derivation of a systematic way of setting τ_{min} we will neglect the influence of the heuristic information on the probabilities given by Equation 2. This is possible if the influence of the heuristic information is low, which, as is typically done in $\mathcal{MM}$AS, is the case if the parameter β is chosen rather low or if no heuristic information is used at all.

Given these assumptions, good values for τ_{min} can be found by relating the convergence of the algorithm to the minimum trail limit. When $\mathcal{MM}$AS has *converged*, the best solution found is constructed with a probability p_{best} which is significantly higher than 0.[4] In this situation, an ant constructs the best solution found if it makes at each choice point the "right" decision and chooses a solution component with maximum pheromone trail τ_{max}. In fact, the probability p_{dec} of choosing the corresponding solution component at a choice point directly depends on τ_{max} and τ_{min}. For the sake of simplicity, let us assume that p_{dec} is constant at all decision points. Then, an ant has to make n times the "right" decision and, hence, it will

[4] Intuitively, we require p_{best} to be relatively large and later give numeric examples of reasonable values for p_{best} for the the $\mathcal{MM}$AS application to the TSP.

construct the best solution with a probability of p_{dec}^n. By setting

$$p_{dec}^n = p_{best} \tag{8}$$

we can determine p_{dec} as

$$p_{dec} = \sqrt[n]{p_{best}}. \tag{9}$$

So, given a value for p_{best}, we can now determine appropriate settings for τ_{min}. On average, at each choice point an ant has to choose among $avg = n/2$ solution components. Then, the probability p_{dec} of making the right decision according to Equation 2 can be calculated as [5]

$$p_{dec} = \frac{\tau_{max}}{\tau_{max} + (avg - 1) \cdot \tau_{min}} \tag{10}$$

Solving this equation for τ_{min} yields

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - p_{dec})}{(avg - 1) \cdot p_{dec}} = \frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(avg - 1) \cdot \sqrt[n]{p_{best}}} \tag{11}$$

Note that if $p_{best} = 1$, then $\tau_{min} = 0$. If p_{best} is too small, it may happen that by Equation 11 $\tau_{min} > \tau_{max}$. In this case we set $\tau_{min} = \tau_{max}$ which corresponds to using only the heuristic information in the solution construction. Based on Equation 11, we can determine τ_{min}, given a value for p_{best}. Choosing values for p_{best} is directly related to the amount of exploration done by $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System when it has converged. Thus, p_{best} provides a good way of investigating the effect of the lower trail limits on the performance of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System.

In Section 4.4.2 we will investigate the proposed settings of τ_{min} and we experimentally show the usefulness of the lower trail limits.

4.3 Pheromone trail initialization

In $\mathcal{MM}$AS we initialize the pheromone trails in such a way that after the first iteration all pheromone trails correspond to $\tau_{max}(1)$. This can easily be achieved by setting $\tau(0)$ to some arbitrarily high value. After the first iteration of $\mathcal{MM}$AS, the trails will be forced to take values within the imposed bounds, in particular, they will be set to $\tau_{max}(1)$. This type of trail initialization is chosen to increase the exploration of solutions during the first iterations of the algorithm. To illustrate this fact, consider the following example: Due to the trail evaporation (determined by parameter ρ), after the first iteration the relative difference between the pheromone trails on solution components will differ by a ratio of at most ρ, after the second by ρ^2, etc. If, on the contrary, the pheromone trails would be initialized to their lower limits τ_{min}, the relative differences between the pheromone

[5] Equation 10 is obtained from Equation 2 by requiring that the solution component with pheromone trail τ_{max} is chosen and we have $avg-1$ other solution components with associated pheromone trail τ_{min}.

trails would increase much more strongly; in particular, in this latter case, the ratio between τ_{min} and the amount of pheromone deposited on a solution element is $(1-\rho)\cdot(avg \cdot p_{dec})/(1-p_{dec})$. With the empirically chosen parameter settings, this ratio is significantly higher than the relative difference among the pheromone trail when initializing the pheromone trails to τ_{max}. For example, with the parameter settings chosen for the experimental investigation in the next section, in the first case this factor would amount to 6.44, while when initializing the pheromone trails to τ_{max} it corresponds to 1.02. Thus, the selection probabilities of Equation 2 evolve more slowly when initializing the pheromone trails to τ_{max} and, hence, the exploration of solutions is favored. The experimental results presented in Section 4.4.3 confirm the conjecture that the larger exploration of the search space due to setting $\tau(0) = \tau_{max}$ improves $\mathcal{MM}$AS' performance.

4.4 Experiments with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

In this section we experimentally study the effectiveness of the three main modifications of $\mathcal{MM}$AS compared to AS and the influence of specific parameter settings on $\mathcal{MM}$AS performance. The experimental study uses the TSP as example application and here we use $\mathcal{MM}$AS without local search; for a detailed overview of the results obtained with $\mathcal{MM}$AS with local search for the TSP we refer to Section 5. All the experiments were performed with a *ceteris paribus* assumption, that is, in each experiment only one single factor is varied and, hence, performance differences can only be attributed to the variation of this single factor.

Unless explicitly indicated otherwise, the following default parameter settings are used. We choose $\beta = 2, \alpha = 1, m = n$ (where m is the number of ants) and $\rho = 0.98$, an evaporation rate which results in a rather slow convergence for $\mathcal{MM}$AS. The pheromone update is done using only the iteration-best ant. The pheromone trail limits were chosen as proposed in Section 4.2 with $p_{best} = 0.05$. The ants start their solution construction from a randomly chosen city and they use candidate lists of length 20 which contain the nearest neighbors ordered according to nondecreasing distances [1,28,40]. When constructing a tour, an ant chooses probabilistically according to Equation 2 the next city among those in the candidate list, if possible. Only if all the members of the candidate list of a city have already been visited, one of the remaining cities is chosen. In this latter case we deterministically choose the city for which $[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}$ is maximum.

The TSP benchmark instances are all taken from TSPLIB; for all instances the optimal solution value is known. We will refer to the benchmark instances by the identifier used in TSPLIB which indicates the number of cities (instance `eil51` has 51 cities, etc.).

4.4.1 Parameter values for ρ

To examine the influence of different values of the pheromone trail evaporation rate ρ, which determines the convergence speed of $\mathcal{MM}$AS towards good solutions, we present curves for the tradeoff between the average solution quality versus the number of tour constructions for the two TSP instances `kroA100` and `d198`

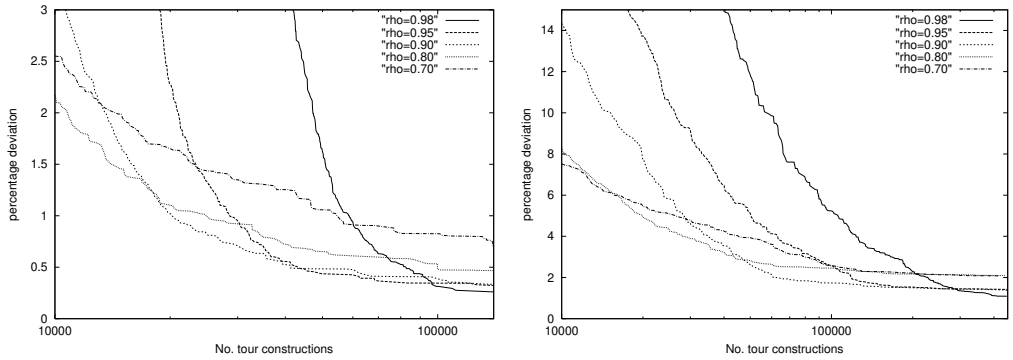

Fig. 4. Influence of the parameter ρ on the tradeoff between the number of tour constructions (given on x-axis) and the solution quality (given on y-axis) on TSP instances `kroA100` (left) and `d198` (right). Note the log-scale on the x-axis; the upper and leftmost parts of the curves were cut off to focus on the important details.

using different settings of ρ averaged over 25 and 10 independent executions of the algorithms, respectively. The maximum number of tour constructions is $2500 \cdot n$ and ρ is varied between 0.7 and 0.99.

In Figure 4, it can be observed that for a low number of tour constructions, better tours are found when using lower values of ρ. This is due to the fact that for lower ρ the pheromone trails on arcs which are not reinforced decrease faster and, hence, the search concentrates earlier around the best tours seen so far. If ρ is high, too few iterations are performed to reach marked differences between the pheromone trails on arcs contained in high quality tours and those which are not part of the best tours. For a larger number of tour constructions, however, using higher ρ values pays off, because the algorithm is able to explore longer the search space. Additionally, it is interesting to note that with more tour constructions the average performance increases generally for all values of ρ. This is mainly due to the effect of the lower trail limits (see also next section).

4.4.2 Lower pheromone trail limits

To investigate the effectiveness of the lower trail limits, we compare experimental results obtained by systematically varying p_{best} (as proposed in Section 4.2) and without using lower pheromone trail limits ($\tau_{min} = 0$). As before, we allow a maximum $2500 \cdot n$ tour constructions, which is sufficient to achieve convergence of $\mathcal{MM}$AS on every instance.

The average solution qualities obtained on four symmetric TSP instances are given in Table 3. For *all* instances the average solution quality is always better if lower trail limits are used. $\mathcal{MM}$AS's performance seems to be quite robust with respect to the particular value chosen for the pheromone trail limits (via p_{best}). Even when no lower trail limits are used, the results are quite good (compared, for example, with the results given for longer runs in Section 4.6 for other AS variants). Hence, facilitating a slow convergence by setting ρ to some large value and introducing elitism seems to be effective in practice. Yet, it should be noted that the relative difference between the average solution quality obtained with or without lower

Table 3
Computational results for systematically varying p_{best} and without lower pheromone trail limits ($\tau_{min} = 0$). Given are the average tour length, averaged over 25 runs, and in parenthesis the percentage deviation from the optimal tour length. Note that the smaller p_{best} the tighter are the trail limits. The best results are indicated in bold-face.

instance	$p_{best} = 0.0005$	$p_{best} = 0.005$	$p_{best} = 0.05$	$p_{best} = 0.5$	$\tau_{min} = 0$
eil51	428.5 (0.59%)	428.0 (0.46%)	427.8 (0.43%)	**427.7** (0.39%)	427.8 (0.43%)
kroA100	21344.8 (0.29%)	21352.8 (0.33%)	**21336.9** (0.26%)	21353.9 (0.34%)	21373.2 (0.43%)
d198	16024.9 (1.55%)	15973.2 (1.22%)	**15952.3** (1.09%)	16002.3 (1.41%)	16047.6 (1.70%)
lin318	42363.4 (0.80%)	**42295.7** (0.64%)	42346.6 (0.75%)	42423.0 (0.94%)	42631.8 (1.43%)

pheromone trail limits appears to increase with increasing instance size. Hence, the use of the lower trail limits in $\mathcal{MM}$AS is definitely advantageous.

4.4.3 Pheromone trail initialization

In $\mathcal{MM}$AS the trails are initialized to their upper trail limit. To show the usefulness of the proposed trail initialization we compare it to a trail initialization at the lower pheromone trail limits; the computational results are given in Table 4. We find that with the proposed trail initialization for all instances, except the smallest one, a better solution quality can be obtained; again the differences appear to increase with increasing instance size. Hence, the higher exploration of the search space achieved in this way seems to be important to achieve a better solution quality.

Table 4
Computational results for pheromone initialization to the upper trail limit ($\tau(0) = \tau_{max}$) and to the lower trail limit ($\tau(0) = \tau_{min}$). Given are the average tour length, averaged over 25 runs, and in parenthesis the percentage deviation from the optimal tour length. The results for setting $\tau(0) = \tau_{max}$ are reproduced from the previous section. The best results are indicated in bold-face.

instance	$\tau(0) = \tau_{max}$	$\tau(0) = \tau_{min}$
eil51	427.8 (0.43%)	**427.7 (0.39%)**
kroA100	**21336.9 (0.26%)**	21362.3 (0.37%)
d198	**15952.3 (1.09%)**	16051.8 (1.72%)
lin318	**42346.6 (0.75%)**	42737.6 (1.68%)

4.4.4 Global versus iteration-best update

As mentioned before, updating the pheromone trails with s^{ib} may give advantages over using s^{gb}. We compare these two choices by running the same experiments as before, but always using s^{gb} for the pheromone trail update. Additionally, we investigate the influence of the lower pheromone trail limits by running each of the experiments with and without imposing lower pheromone trail limits.

The results are given in Table 5. The average performance when using the iteration-best ants for pheromone update is significantly better than using only the global-best ant. For example, a closer examination of the results (not reported here) showed that the worst solution obtained with the standard settings for $\mathcal{MM}$AS was better than the average solution quality when using s^{gb} with pheromone trail limits

for all instances. In general, using exclusively s^{gb} for the pheromone trail update seems not to be a very good idea for $\mathcal{MM}$AS. Yet, the lower pheromone trail limits help to significantly improve the performance when using s^{gb}. Nevertheless, mixed strategies which sometimes use s^{gb} may be helpful for achieving better exploitation of the search results. Experiments on larger instances have shown that using such mixed strategies with a frequency of s^{gb} increasing over time may yield a faster convergence of the algorithm and produce improved results.

Table 5
Computational results for comparison of global-best update (s^{gb}) versus iteration-best update (s^{ib}) with and without using lower pheromone trail limits (indicated by either $+$limits or $-$no-limits). Given are the average tour length, averaged over 25 runs, and in parenthesis the percentage deviation from the optimal tour length. The best results are indicated in bold-face.

instance	$s^{ib}+$limits	$s^{gb}+$limits	$s^{ib}-$no-limits	$s^{gb}-$no-limits
eil51	**427.8 (0.43%)**	429.2 (0.75%)	**427.8 (0.43%)**	434.1 (1.89%)
kroA100	**21336.9 (0.26%)**	21417.1 (0.64%)	21373.2 (0.43%)	21814.7 (2.50%)
d198	**15952.3 (1.09%)**	16136.1 (2.26%)	16047.6 (1.70%)	16473.7 (4.40%)
lin318	**42346.6 (0.75%)**	42901.0 (2.08%)	42631.8 (1.43%)	44558.5 (6.02%)

4.5 Smoothing of the pheromone trails

An additional mechanism, called *pheromone trail smoothing* (PTS), may be useful to increase $\mathcal{MM}$AS performance and, more generally, of any elitist versions of AS. When $\mathcal{MM}$AS has converged or is very close to convergence (as indicated by the average branching factor [17]), this mechanism increases the pheromone trails proportionally to their difference to the maximum pheromone trail limit:

$$\tau^*_{ij}(t) = \tau_{ij}(t) + \delta \cdot (\tau_{max}(t) - \tau_{ij}(t)) \quad \text{with } 0 < \delta < 1, \tag{12}$$

where $\tau_{ij}(t)$ and $\tau^*_{ij}(t)$ are the pheromone trails before and after the smoothing. The basic idea of PTS is to facilitate the exploration by increasing the probability of selecting solution components with low pheromone trail. The proposed mechanism has the advantage that for $\delta < 1$, the information gathered during the run of the algorithm (which is reflected in the pheromone trails), is not completely lost but merely weakened. For $\delta = 1$ this mechanism corresponds to a reinitialization of the pheromone trails, while for $\delta = 0$ PTS is switched off.

PTS is especially interesting if long runs are allowed, because it helps achieving a more efficient exploration of the search space. At the same time, PTS makes $\mathcal{MM}$AS less sensitive to the particular choice of the lower pheromone trail limit.

4.6 Comparison of ant algorithms

In this section we compare the performance of the proposed improvements over AS based on longer runs for some symmetric and asymmetric TSP instances which had been proposed for the First International Contest on Evolutionary Optimization

Table 6
Computational results for symmetric (upper part) and asymmetric TSP (lower part) instances from TSPLIB, details on the parameter settings are given in the text. *opt* indicates the known optimal solution value of each instance. All algorithms are using the same maximum number of tour constructions. Results for ACS are taken from [18]. For each instance we report the average solution quality, best results are indicated in bold-face. "+pts" indicates that pheromone trail smoothing was used. The best results are indicated in bold-face.

instance	opt	$\mathcal{MM}$AS+pts	$\mathcal{MM}$AS	ACS	AS_{rank}	AS_{rank}+pts	AS_e	AS_e+pts	AS
eil51	426	**427.1**	427.6	428.1	434.5	428.8	428.3	427.4	437.3
kroA100	21282	**21291.6**	21320.3	21420.0	21746.0	21394.9	21522.8	21431.9	22471.4
d198	15780	**15956.8**	15972.5	16054.0	16199.1	16025.2	16205.0	16140.8	16702.1
ry48p	14422	14523.4	14553.2	14565.4	**14511.4**	14644.6	14685.2	14657.9	15296.4
ft70	38673	**38922.7**	39040.2	39099.0	39410.1	39199.2	39261.8	39161.0	39596.3
kro124p	36230	**36573.6**	36773.5	36857.0	36973.5	37218.0	37510.2	37417.7	38733.1
ftv170	2755	**2817.7**	2828.8	2826.5	2854.2	2915.6	2952.4	2908.1	3154.5

[2]. The comparison is done based on the same number of tour constructions for all algorithms; this number is chosen as $k \cdot n \cdot 10000$, where $k = 1$ for symmetric TSPs and $k = 2$ for ATSPs and n is the number of cities of an instance.

We compare the performance of $\mathcal{MM}$AS to that obtained with AS, AS$_e$, AS$_{rank}$, and ACS. The computational results obtained with ACS are taken directly from [12] while the results for AS$_e$, AS$_{rank}$, and AS are obtained using our implementation of these algorithms.

For $\mathcal{MM}$AS the parameters were set to their default values as described before, except that in every 10th iteration s^{gb} is used to reinforce the pheromone trails. Additionally, we run $\mathcal{MM}$AS with and without the PTS mechanism (PTS is indicated by +pts); in the former case, we chose $\delta = 0.5$. PTS has been added in an ad-hoc manner without fine-tuning parameters. In AS we set $\alpha = 1.0$, $\beta = 5.0$. In AS$_e$ additionally $e = n$ elitist ants give reinforcement to s^{gb}, which showed to give best peroformance. For AS$_{rank}$ we used the parameter settings proposed in [6], that is, $\alpha = 1, \beta = 5,$ and $e = 6$. Additionally, we also run AS$_e$ and AS$_{rank}$ with $\beta = 1.0$ for ATSPs and $\beta = 2.0$ for symmetric TSPs using the PTS mechanism; in this case we directly reinitialize the trails since no trail limits are used in these two algorithms.

The computational results in Table 6 show that generally $\mathcal{MM}$AS achieves the best performance. The only exception is ATSP instance ry48p, for which AS$_{rank}$ has a better average peformance. Yet, differently from $\mathcal{MM}$AS and ACS, AS$_{rank}$ never found the optimal solution for that instance. Also, except for one single instance (d198) the overall best solution for each instance was always found by $\mathcal{MM}$AS. We also mention that on two larger symmetric TSP instances (att532 and rat783), the average solution quality produced by $\mathcal{MM}$AS was even better than the best solution found by ACS [45]. Regarding the performance of AS it can be clearly seen that AS performs very poorly compared to the other algorithms.

The advantage of $\mathcal{MM}$AS over ACS (overall the second best performing ACO algorithm) with respect to solution quality is more notable on the symmetric instances, while on the ATSP instances they perform similarly. Interestingly, the so-

lution quality obtained with AS_e and AS_{rank} without PTS is, in general, significantly worse than that of $\mathcal{MM}$AS and ACS. If PTS together with a lower influence of the heuristic function is used, both algorithms catch up (the only exception being AS_{rank} on the ATSP instances) and roughly reach the solution quality obtained by ACS on the symmetric TSPs, but they are still worse on most ATSP instances. Thus, the PTS mechanism seems to be effective for increasing the performance the algorithms. It also helps to slightly improve the performance of $\mathcal{MM}$AS, yet, not as strongly as for AS_e and AS_{rank}.

In [12] it was shown that ACS shows generally good performance and that it is competitive with other nature-inspired algorithms applied to the TSP. Since $\mathcal{MM}$AS achieves better solution qualities than ACS on most instances, our computational results demonstrate the competitive performance of $\mathcal{MM}$AS when compared to other improvements on AS as well as other nature-inspired algorithms. Yet, to obtain results competitive with the best performing algorithms for the TSP, local search has to be used to improve solutions.

5 Experimental results for the TSP

In this section we present computational results of $\mathcal{MM}$AS when combined with local search on some larger TSP instances from TSPLIB. For the symmetric TSPs we use the `3-opt` local search algorithm (see Section 3.2). In addition to the techniques described there, we use *don't look bits* associated with each node [1]. The use of don't look bits leads to a further, significant speed-up of the local search algorithm at only a small loss in solution quality.

For ATSPs, we use a restricted form of `3-opt`, called `reduced 3-opt`. It considers only those moves which do not lead to a reversal of the city order in which a subtour is traversed. If subtours are reversed, one would have to re-calculate the lenght of this subtour, leading to high computation times.

Note that, when applying ACO algorithms to the TSP, pheromone trails are stored in a matrix with $\mathcal{O}(n^2)$ entries (one for each arc). Because of pheromone trail evaporation (according to Formula 3), all the entries of this matrix have to be updated after each iteration (this is not the case in ACS). Obviously, this is a very expensive operation if large TSP instances with several hundreds of cities are attacked. To speed up the pheromone update, in $\mathcal{MM}$AS we apply pheromone evaporation only to arcs connecting a city i to cities belonging to i's candidate list. This reduces the cost of updating the pheromone trails to $\mathcal{O}(n)$.

5.1 Parameter settings and applied variants

In preliminary experiments we noted that, if only the iteration-best solution is chosen for the pheromone trail update, $\mathcal{MM}$AS takes long time to converge and to find very high quality solutions when applied to large instances; a discussion of this issue can be found in [47]. Yet, when giving the global-best solution a high frequency f^{gb} for the pheromone trail update (let f^{gb} indicate that every f^{gb} iterations

s^{gb} is allowed to deposit pheromone), the initial exploration of the search space may be rather limited and worse results are obtained. Then, the best performance was obtained by using a mixed strategy in which f^{gb} increases over time within a single run. To realize this, we apply a specific schedule to alternate the pheromone trail update between s^{gb} and s^{ib}. In the first 25 iterations only s^{ib} is used to update the pheromone trails; we set f^{gb} to 5 for $25 < t \leq 75$ (where t is the iteration counter), to 3 for $75 < t \leq 125$, to 2 for $125 < t \leq 250$, and to 1 for $t > 250$. By gradually shifting the emphasis from the iteration-best to the global-best solution for the pheromone trail update, we achieve a transition between a stronger exploration of the search space early in the search to a stronger exploitation of the overall best solution later in the run.

The other parameters were chosen as follows. We use $m = 25$ ants (all ants apply a local search to their solution), $\rho = 0.8$, $\alpha = 1$, and $\beta = 2$. During the tour construction the ants use a candidate list of size 20. We set τ_{max} as proposed in Section 4.2. Since here the solutions constructed by the ants are improved by additional local search, we used somewhat tighter bounds on the allowed pheromone trail strength by setting $\tau_{min} = \tau_{max}/2n$, which roughly corresponds to $p_{best} = 0.005$.

In the following we will further examine whether for the hybrid algorithm using the pheromone trail limits is sufficient to achieve very high solution quality or whether with additional diversification mechanisms based on pheromone trail reinitialization a better solution quality can be achieved. In particular, we study two further variants which differ in the degree of search diversification. In the first of these, we reinitialize the pheromone trails to τ_{max} (this corresponds to setting $\delta = 1$ in Equation 12) whenever the pheromone trail strengths on almost all arcs not contained in s^{gb} are very close to τ_{min} (as indicated by the average branching factor [17] which is calculated every 100 iterations) and no improved solution could be found for 50 iterations. After the restart, the schedule for f^{gb} is applied as done at the start of the algorithm. This variant will be referred to as $\mathcal{MM}$AS+ri (for reinitialization), while the original version without pheromone trail reinitialization will be referred to as $\mathcal{MM}$AS-nri.

Even more search diversification is achieved if additionally after a pheromone trail reinitialization, the best solution found since the reinitialisation of the pheromone trails is used instead of s^{gb}. This allows $\mathcal{MM}$AS to converge to another very high quality solution. Still, s^{gb} could be better than the best solution found after the reinitialisation. Thus, to re-focus the search around s^{gb}, we use s^{gb} for the pheromone trail update if more than 250 iterations have been executed without a reinitialization of the pheromone trails and for 25 iterations no improved solution has been found. (Note that these values are chosen such that $\mathcal{MM}$AS may have already converged.) This latter version of $\mathcal{MM}$AS will be referred to as $\mathcal{MM}$AS+rs (for restart).

5.2 Experimental results for symmetric TSPs

In this section we report on experimental results obtained with $\mathcal{MM}$AS-nri, $\mathcal{MM}$AS+ri, and $\mathcal{MM}$AS+rs on symmetric TSP instances from TSPLIB. The ex-

periments are performed on a Sun UltraSparc I 167MHz processors with 0.5MB external cache and 192 MB RAM.

The computational results given in Table 7 show that $\mathcal{MM}$AS, in general, is able to find very high quality solutions for all instances; furthermore, for almost all instances $\mathcal{MM}$AS finds the optimal solution in at least one of the runs. This is an encouraging results which shows the viability of the ant approach to generate very high quality solutions for the TSP. Note that the computational results with local search are also much better than those obtained without local search, as can be seen when comparing the computational results given in Table 7 with those of Table 6. Additionally, the computation times with local search are much smaller. When comparing the computational results of the three variants, we find that $\mathcal{MM}$AS+rs performs best; on most instances it achieves the best average solution quality and the worst solution found with $\mathcal{MM}$AS+rs is typically much better than for the other two variants. Only on instances `d198` and `fl1577` the average solution quality of $\mathcal{MM}$AS-nri is slightly better. $\mathcal{MM}$AS-nri and $\mathcal{MM}$AS+ri show a very similar performance. Only on instance `lin318` $\mathcal{MM}$AS+ri performs significantly better than $\mathcal{MM}$AS-nri and found the optimal solution in all runs. Hence, the stronger search diversification of $\mathcal{MM}$AS+rs is mainly responsible for the improved performance.

According to these results, $\mathcal{MM}$AS is currently the best performing ant approach for the TSP. In particular, it shows better performance than ACS, when comparing the computational results reported here to those presented in [12] or to those of our own implementation of ACS using the same local search algorithm as $\mathcal{MM}$AS. One factor which may be responsible for this fact is that ACS concentrates the search too strongly around s^{gb}.

5.3 Experimental results with the Lin-Kernighan heuristic

The best performing local search algorithm with respect to solution quality for symmetric TSPs is the Lin-Kernighan heuristic (`LK`) [28] which considers a variable number of arcs to be exchanged. Yet, the `LK` heuristic is much more difficult to implement than `2-opt` or `3-opt`, and careful fine-tuning is necessary to get it to run very fast and produce high quality solutions [40,23]. Here, we used an `LK` implementation provided by Olivier Martin to give an indication of the solution quality which may be reached by using the `LK` local search in $\mathcal{MM}$AS.

The motivation for applying the `LK` heuristic is the observation made in [51] that for genetic local search algorithms a much better solution quality is obtained with the `LK` heuristic than by using `2-opt` [51]. These results suggest that the solution quality of $\mathcal{MM}$AS can be further increased using the `LK` heuristic. To confirm this conjecture we present computational results for $\mathcal{MM}$AS with `LK` local search allowing a maximum number of 5000 `LK` applications on each instance. Since not as many local searches as with our `3-opt` implementation can be done, we used slightly different parameter settings than before. Most significantly, we use only 10 ants and the schedule for using the global-best for the pheromone update is shortened. Here, we use $u_{gb} = 3$ for $t \leq 19$, $u_{gb} = 2$ for $20 < t \leq 35$, and $u_{gb} = 1$

Table 7
Comparison of different variants of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System on symmetric TSP instances. Given are the instance name (the number in the name gives the problem dimension, that is, the number of cities), the algorithm used, the best solution, the average solution quality (its percentage deviation from the optimum in parentheses), the worst solution generated, the average number of iterations i_{avg} and the average time t_{avg} to find the best solution in a run, and the maximum allowed computation time t_{max}. Averages are taken over 25 trials for $n < 1000$, over 10 trials on the larger instances. Best average results are printed in bold-face.

instance	Algorithm	Best	Average	Worst	i_{avg}	t_{avg}	t_{max}
d198	$\mathcal{MM}$AS+rs	15780	15780.3 (0.00%)	15781	121.2	54.9	170
	$\mathcal{MM}$AS+ri		15780.4 (0.00%)	15784	134.2	61.7	
	$\mathcal{MM}$AS-nri	15780	**15780.2 (0.00%)**	15781	106.7	59.9	
lin318	$\mathcal{MM}$AS+rs	42029	**42029.0 (0.00%)**	42029	131.16	87.8	450
	$\mathcal{MM}$AS+ri	42029	**42029.0 (0.00%)**	42029	139.0	94.2	
	$\mathcal{MM}$AS-nri	42029	42061.7 (0.08%)	42163	77.9	65.9	
pcb442	$\mathcal{MM}$AS+rs	50778	50905.3 (0.25%)	50931	603.8	217.1	600
	$\mathcal{MM}$AS+ri	50778	50911.2 (0.26%)	51047	522.0	308.9	
	$\mathcal{MM}$AS-nri	50778	**50900.9 (0.24%)**	50931	449.7	319.5	
att532	$\mathcal{MM}$AS+rs	27686	**27701.9 (0.06%)**	27709	481.0	521.8	1250
	$\mathcal{MM}$AS+ri	27686	27707.9 (0.08%)	27756	335.8	387.3	
	$\mathcal{MM}$AS-nri	27686	27708.6 (0.08%)	27741	289.4	309.6	
rat783	$\mathcal{MM}$AS+rs	8806	**8810.9 (0.06%)**	8823	870.3	1336.8	2100
	$\mathcal{MM}$AS+ri	8806	8814.4 (0.10%)	8837	631.5	965.2	
	$\mathcal{MM}$AS-nri	8806	8816.8 (0.12%)	8848	805.5	1395.2	
pcb1173	$\mathcal{MM}$AS+rs	56892	**56906.8 (0.03%)**	56939	1697.2	3171.2	5400
	$\mathcal{MM}$AS+ri	56896	56956.0 (0.11%)	57120	1669.2	3219.5	
	$\mathcal{MM}$AS-nri	56892	56946.3 (0.10%)	57040	1138.3	2051.0	
d1291	$\mathcal{MM}$AS+rs	50801	**50812.9 (0.02%)**	50833	1747.8	3203.7	5400
	$\mathcal{MM}$AS+ri	50801	50821.6 (0.04%)	50838	1035.0	1894.4	
	$\mathcal{MM}$AS-nri	50801	50828.8 (0.05%)	50870	669.6	1206.9	
fl1577	$\mathcal{MM}$AS+rs	22289	22305.6 (0.25%)	22323	1681.7	5348.3	7200
	$\mathcal{MM}$AS+ri	22286	22311.0 (0.28%)	22358	690.7	3001.8	
	$\mathcal{MM}$AS-nri	22261	**22271.8 (0.10%)**	22279	1409.2	4473.9	

for $t > 35$. The other parameter settings are the same as before.

The computational results with respect to solution quality obtained by combining $\mathcal{MM}$AS with the LK heuristic are significantly better with respect to solution quality than those using our 3-opt implementation (see Table 8). Yet, the run-times are higher due to the local search.

5.4 Experimental results for ATSPs

We applied the same versions of $\mathcal{MM}$AS also to the ATSP; the computational results are given in Table 9. Here we only present results obtained with $\mathcal{MM}$AS-nri and $\mathcal{MM}$AS+rs, being the computational results with $\mathcal{MM}$AS+ri almost identical

Table 8
Experimental results of $\mathcal{MM}$AS when using the Lin-Kernighan local search procedure. Given are the best, the average, and the worst solution obtained averaged over 25 independent runs for $n < 1000$ and 10 runs otherwise. We allowed a maximum of 5000 LK applications. Additionally, are given the average number of iterations (i_{avg}) to find the best solution in a run, and the average time t_{avg} to do so.

instance	Best	Average	Worst	i_{avg}	t_{avg}
lin318	42029	42029.0 (0.0%)	42029	29.9	298.3
pcb442	50778	50778.0 (0.0%)	50778	65.7	978.9
att532	27686	27692.7 (0.002%)	27704	299.1	2444.6
rat783	8806	8806.5 (0.006%)	8809	345.4	1001.8
pcb1173	56892	56893.5 (0.003%)	56897	289.9	3581.7

Table 9
Comparison of $\mathcal{MM}$AS with and without pheromone trail reinitialization on some ATSP instance. Given are the instance name (the number in the name gives the problem dimension, that is, the number of cities; an exception is instance kro124p which has 100 cities), the algorithm used, the best solution, the average solution quality (its percentage deviation from the optimum in parentheses), the worst solution generated, the average number of iterations i_{avg} and the average time t_{avg} to find the best solution in a run, and the maximally allowed computation time t_{max}. Averages are taken at least over 25 trials. Best average results are printed in bold-face.

instance	Algorithm	Best	Average	Worst	i_{avg}	t_{avg}	t_{max}
ry48p	$\mathcal{MM}$AS+rs	14422	**14422.0 (0.0%)**	14422	22.5	2.5	120
	$\mathcal{MM}$AS-nri	14422	**14422.0 (0.0%)**	14422	34.9	4.3	
ft70	$\mathcal{MM}$AS+rs	38673	**38673.0 (0.0%)**	38673	140.5	24.6	300
	$\mathcal{MM}$AS-nri	38673	38686.6 (0.04%)	38707	214.4	47.3	
kro124p	$\mathcal{MM}$AS+rs	36230	**36230.0 (0.0%)**	36230	22.9	6.2	300
	$\mathcal{MM}$AS-nri	36230	**36230.0 (0.0%)**	36230	23.9	7.4	
ftv170	$\mathcal{MM}$AS+rs	2755	**2755.0 (0.0%)**	2755	147.6	46.7	600
	$\mathcal{MM}$AS-nri	2755	2757.8 (0.08%)	2764	124.5	39.9	

to those of $\mathcal{MM}$AS+rs. On the asymmetric instances $\mathcal{MM}$AS-nri and $\mathcal{MM}$AS+rs show the same performance on instances ry48p and kro124p. Yet, $\mathcal{MM}$AS+rs could solve the two instances ft70 and ftv170 in all runs to optimality, which could not be achieved without the additional diversification mechanism based on pheromone trail reinitialization. The reason for the performance difference between $\mathcal{MM}$AS-nri and $\mathcal{MM}$AS+rs may be that, despite the pheromone trail limits, $\mathcal{MM}$AS gets stuck at solutions corresponding to local minima with large attraction regions and for an escape from these regions the currently best found solution has to be strongly restructured. Since at convergence of $\mathcal{MM}$AS the arcs with maximal amount of pheromone will be rather frequently chosen, by a reinitialisation of the pheromone trails the chances of escaping from such attraction regions are higher.

5.5 Discussion and related work

According to the presented results, $\mathcal{MM}$AS is currently the best performing ACO algorithm for the TSP. In particular, it shows better performance for symmetric TSP instances than ACS while ACS and $\mathcal{MM}$AS reach the same level of performance on ATSPs. In summary, the computational results with the three $\mathcal{MM}$AS variants suggest that (i) for the TSP very high solution quality can be obtained with $\mathcal{MM}$AS, (ii) the best computational results are obtained when in addition to the pheromone trail limits effective diversification mechanisms based on pheromone re-initialization are used. In general, we found that an effective search diversification is necessary to achieve best performance when applying AS_e or AS_{rank} with additional local search.

Because the TSP is a standard benchmark problem for meta-heuristic algorithms, it has received considerable attention from the research community. Here, we only mention some of the most recent work, for a discussion of earlier work we refer to the overview article by Johnson and McGeoch [23]. Currently, the iterated `LK` heuristic (ILK) is the most efficient approach to symmetric TSPs for short to medium run-times [23]. Recently, several new approaches and improved implementations have been presented which appear to perform as well or better than ILK for longer run-times. Among these algorithms we find the genetic local search approach of Merz and Freisleben [16,33], a new genetic local search approach using a repair-based crossover operator and brood selection by Walters [52], a genetic algorithm using a specialized crossover operator, called edge assembly crossover, due to Nagata and Kobayashi [38], and finally a specialized local search algorithm for the TSP called Iterative Partial Transcription by Möbius et.al. [36]. Some of these algorithms achieve better computational results than the ones presented here. For example, the genetic local search approach presented in [33], which uses the `LK` heuristic for the local search, reaches on average a solution of 8806.2 on instance `rat783` in 424 seconds on a DEC Alpha station 255 MHz. Obviously, the computational results for $\mathcal{MM}$AS would also benefit from a faster implementation of the `LK` heuristic like the one used in [33,23]. Yet, it is an open question whether the performance (with respect to computation time) of the currently best algorithms for symmetric TSPs can be reached.

Applied to asymmetric TSP instances, our computational results with respect to solution quality compare more favorably to these approaches. For example, the solution quality we obtain with $\mathcal{MM}$AS is better than that of the genetic local search approach of [33] and the same as reported in [52], but at the cost of higher run-times.

6 Experimental results for the QAP

In this section we report on the experimental results obtained with $\mathcal{MM}$AS when applied to the QAP and compare it with other well known algorithms from literature.

As outlined in Section 2.4, ACO algorithm applications to the TSP can straightforwardly be extended to the QAP. When applied to the QAP, in $\mathcal{MM}$AS we construct solutions by assigning facilities to locations in random order. Differently from the TSP application, for the QAP we do not use any heuristic information for the solution construction. In fact, it has been shown that the heuristic information is not necessary for the hybrid $\mathcal{MM}$AS algorithm which combines solution construction with local search to obtain high quality solutions [45]. For example, when running $\mathcal{MM}$AS on the TSP without heuristic information (which is simply achieved by setting $\beta = 0$ in Equation 2) only a very slight solution degradation could be noted. The $\mathcal{MM}$AS approach for the QAP is a straightforward extension of the $\mathcal{MM}$AS+ri version which has been discussed in the previous section.

6.1 Parameter settings

Suitable parameter settings for $\mathcal{MM}$AS-QAP were determined in some preliminary experiments. We use $m = 5$ ants (all ants apply local search to the solution they generate) and set $\rho = 0.8, \alpha = 1.0$. The low number of ants is motivated by the fact that local search for large QAP instances is computationally demanding, but a reasonable number of iterations should be performed to learn the pheromone trails. The trail limits are determined according to Equation 7, that is, we set $\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{f(\phi^{gb})}$, and according to Equation 11 by setting $p_{best} = 0.005$ and $avg = n/2$ (at a choice point an ant has to assign — on average — a facility to one of $n/2$ locations).

For $\mathcal{MM}$AS applied to the QAP we use two different local search algorithms. One is the simple `2-opt` algorithm briefly described in Section 3; in the following, we refer to the version of $\mathcal{MM}$AS using this local search algorithm as $\mathcal{MM}\text{AS}_{\texttt{2-opt}}$. Alternatively, we apply short runs of the robust tabu search algorithm (Ro-TS) [49], a possibility which was first considered in a genetic local search algorithm [15]. This version is called $\mathcal{MM}\text{AS}_{\texttt{TS}}$; it is motivated by the fact that short runs of Ro-TS typically give higher quality solutions than `2-opt`, although at the cost of higher run times. In a sense, Ro-TS is a more powerful local search algorithm and one may expect better results when using it.

We apply the following schedule to alternate the pheromone trail update between ϕ^{gb} and ϕ^{ib}. For the first 9 iterations, we set f^{gb} to 3, for iterations 10 to 24 we use $f^{gb} = 2$, and from iteration 25 on $f^{gb} = 2$; as before, $f^{gb} = k$ indicates that every k iterations ϕ^{gb} updates the trails. Note that $\mathcal{MM}$AS is rather robust to the particular schedule chosen; the schedule given here is not fine-tuned at all. Because we are actually using $\mathcal{MM}$AS+rs, we use ϕ^{gb} for the trail update if more than 30 cycles have passed since the last pheromone trail reinitialization and for 5 iterations no improved solution has been found.

6.2 Computational results

$\mathcal{MM}$AS was applied to a wide range of QAP instances taken from QAPLIB. We only used instances with $n \geq 20$, since smaller instances are too easily solved.

Table 10
Experimental results for heuristic algorithms on QAP instances from classes (i) and (ii). We give the average excess from the best known solutions over 10 independent runs of the algorithms. Best results are printed in boldface. See text for details.

Problem instance	Ro-TS	GH	HAS-QAP	$\mathcal{MM}$AS-QAP$_{\texttt{TS}}$	$\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$
random problems with uniformly distributed matrix entries (i)					
tai20a	**0.108**	0.268	0.675	0.191	0.428
tai25a	**0.274**	0.629	1.189	0.488	1.751
tai30a	**0.426**	0.439	1.311	0.459	0.966
tai35a	**0.589**	0.698	1.762	0.715	1.128
tai40a	0.990	0.884	1.989	**0.794**	1.509
tai50a	1.125	**1.049**	2.800	1.060	1.795
tai60a	1.203	1.159	3.070	**1.137**	1.882
tai80a	0.900	**0.796**	2.689	0.836	1.402
random flows on grids (ii)					
nug30	0.013	**0.007**	0.098	0.013	0.039
sko42	0.025	**0.003**	0.076	0.032	0.051
sko49	0.076	**0.040**	0.141	0.068	0.115
sko56	0.088	**0.060**	0.101	0.075	0.098
sko64	**0.071**	0.092	0.129	**0.071**	0.099
sko72	0.146	0.143	0.277	**0.090**	0.172
sko81	0.136	0.136	0.144	**0.062**	0.124
sko90	0.128	0.196	0.231	**0.114**	0.140

For $\mathcal{MM}$AS$_{\texttt{TS}}$ we applied 250 times short Ro-TS runs of length $4n$. $\mathcal{MM}$AS$_{\texttt{2-opt}}$ is then stopped after the same computation time as taken by $\mathcal{MM}$AS$_{\texttt{TS}}$.

We compare the performance of $\mathcal{MM}$AS to the robust tabu search (Ro-TS) algorithm [49], to a genetic hybrid (GH) method which uses short tabu search runs for the local search [15], and to HAS-QAP [21], another ant-based algorithm. In [50] it was shown that GH performed best the instances of classes (iii) and (iv) (see Section 3), whereas tabu search algorithms like Ro-TS performed best on instances of classes (i) and (ii). Ro-TS is allowed $1000 \cdot n$ iterations, resulting in similar runtimes to $\mathcal{MM}$AS$_{\texttt{TS}}$. In GH, 250 short robust tabu search runs of the same length as in $\mathcal{MM}$AS$_{\texttt{TS}}$ are applied. Hence, the computation times are comparable. HAS-QAP is allowed 1000 applications of a truncated first-improvement `2-opt` local search. Since in [21] it is detailed that the computation times for HAS-QAP are similar to those of GH, the results of HAS-QAP are also roughly comparable with respect to computation times to those of $\mathcal{MM}$AS. In fact, our own implementation of that local search algorithm suggests that HAS-QAP takes roughly 75% of the computation time $\mathcal{MM}$AS is given. The computational results for HAS-QAP and GH are taken directly from [21].

The computational results are presented in Table 10 for instances of classes (i) and (ii), and in Table 11 for those of classes (iii) and (iv). In general, which method performs best depends strongly on the instance class. For the instances of classes (i) and (ii) the hybrids using short tabu search runs and Ro-TS show

the best performance; $\mathcal{MM}\text{AS}_{2-\texttt{opt}}$ and HAS-QAP perform significantly worse than Ro-TS, GH, and $\mathcal{MM}\text{AS}_{\texttt{TS}}$ on instances of class (i) and slightly worse on instances from class (ii).

Table 11
Experimental results for heuristic algorithms on QAP instances from classes (iii) and (iv). We give the average excess from the best known solutions over 10 independent runs of the algorithms. "n.a." indicates that an algorithm has not been applied to a specific instance. Best results are printed in boldface. See text for details.

Problem instance	Ro-TS	GH	HAS-QAP	$\mathcal{MM}$AS-QAP$_{\texttt{TS}}$	$\mathcal{MM}$AS-QAP$_{2-\texttt{opt}}$
real-life instances (iii)					
bur26a-h	0.002	0.043	**0.0**	0.006	**0.0**
kra30a	0.268	**0.134**	0.630	**0.134**	0.157
kra30b	**0.023**	0.054	0.071	0.044	0.066
ste36a	0.155	n.a.	n.a.	**0.061**	0.126
ste36b	0.081	n.a.	n.a.	**0.0**	**0.0**
randomly generated real-life like instances (iv)					
tai20b	**0.0**	**0.0**	0.091	**0.0**	**0.0**
tai25b	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
tai30b	0.107	0.0003	**0.0**	**0.0**	**0.0**
tai35b	0.064	0.107	0.026	0.051	**0.0**
tai40b	0.531	0.211	**0.0**	0.402	**0.0**
tai50b	0.342	0.214	0.192	0.172	**0.009**
tai60b	0.417	0.291	0.048	**0.005**	**0.005**
tai80b	1.031	0.829	0.667	0.591	**0.266**
tai100b	0.512	n.a.	n.a.	0.230	**0.114**

On the real-life (like) instances, the performance characteristics of the algorithms are very different. Here, $\mathcal{MM}\text{AS}_{2-\texttt{opt}}$ and HAS-QAP show much improved performance and, in fact, $\mathcal{MM}\text{AS}_{2-\texttt{opt}}$ is the best algorithm for instances taixxb and bur26x. For example, for all instances bur26x the best known solution value is found in every run, something that could not be achieved neither by $\mathcal{MM}\text{AS}_{\texttt{TS}}$ nor by Ro-TS. Additionally, $\mathcal{MM}\text{AS}_{2-\texttt{opt}}$ finds these best known solutions (which are conjectured to be optimal) on average in 3.8 seconds on a SUN UltraSparc I processor (167Mhz) while Ro-TS finds the best solutions in each run only after 18.5 seconds on average. For instances taixxb, Ro-TS performs, except for the smallest instances, significantly worse than the $\mathcal{MM}$AS hybrids or HAS-QAP. Only on kra30x and ste36a Ro-TS can catch up with the other algorithms.

Interestingly, using a simple local search procedure is sufficient to yield very high quality solutions on the real-life (like) instances and, for example, for the instances taixxb with $n \leq 60$ in almost every run the best-known solutions, which are conjectured to be optimal, are found. Hence, for the instances with a relatively high fitness-distance correlation it seems to be better to apply more often a local search to identify promising regions of the search space. In fact, on these instances $\mathcal{MM}$AS is able to efficiently exploit the structure of the real-life (like) QAP instances and

is able to guide the local search towards very high quality solutions.

One might conjecture that the flow (distance) dominance could be used to identify which algorithm should be used on a particular instance. If the flow and distance dominance are low, the best choice appears to be the use algorithms like $\mathcal{MM}$AS$_{\texttt{TS}}$, GH, or Ro-TS, while for high flow and/or distance dominance, the best would be to apply a hybrid algorithm with a fast local search. Although such a simple rule would work reasonably well, exceptions do occur. For example, although instance `ste36a` has the highest flow dominance among the real-life instances, $\mathcal{MM}$AS$_{\texttt{TS}}$ and even Ro-TS give slightly better average performance than $\mathcal{MM}$AS$_{\texttt{2-opt}}$. Thus, more sophisticated measures of the problem structure have to be developed to predict the relative performance of different algorithmic approaches more reliably.

7 Conclusions

Recent research in ACO algorithms has strongly focused on improving the performance of ACO algorithms. In this paper we have presented $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, an algorithm based on several modifications to AS which aim (i) to exploit more strongly the best solutions found during the search and to direct the ants' search towards very high quality solutions and (ii) to avoid premature convergence of the ants' search. We have justified these modifications by a computational study of $\mathcal{MM}$AS and have shown that all main modifications are important for obtaining peak performance. Our results demonstrate that $\mathcal{MM}$AS achieves a strongly improved performance compared to AS and to other improved versions of AS for the TSP; furthermore, $\mathcal{MM}$AS is among the best available algorithms for the QAP.

One of the main ideas introduced by $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, the utilization of pheromone trail limits to prevent premature convergence, can also be applied in a different way, which can be interpreted as a hybrid between $\mathcal{MM}$AS and Ant Colony System (ACS): During solution construction, the ants in ACS make the best possible choice, as indicated by the pheromone trail and heuristic information, with a fixed probability p and with probability $1 - p$ they make a probabilistic choice as in Equation 2. With high parameter values of p and the fact that only the iteration-best or the global-best solution is chosen for the trail update, a very strong exploitation of the search history results. When combining ACS's *action choice rule* with *tight* pheromone trail limits, we observed a very promising performance (see [48]). Apart from the TSP and the QAP, this latter version has also been applied to the permutation Flow Shop Problem [43] and to the Generalized Assignment Problem [39] obtaining very good results.

One feature $\mathcal{MM}$AS has in common with other improved AS algorithms is the fact that the best solutions found during the search are strongly exploited to direct the ants' search. We have related this feature to recent results of the analysis of search space characteristics for combinatorial optimization problems. Earlier research has shown that there exists a strong correlation between the solution quality and the distance to a global optimum for the TSP and for some other problems.

Here, we performed a fitness-distance correlation analysis for the QAP and found that for real-life and randomly generated real-life like QAP instances there is a significant correlation between the quality of candidate solutions and their distance to optimal solutions while this is not the case for instances with matrix entries generated according to uniform distributions. This suggests that for the first two instances classes $\mathcal{MM}$AS may provide an effective guidance mechanism to direct the search towards the best solutions while this is not the case on the latter instance class. Yet, exploitation of the best solutions is not the only remedy to achieve very high performing ACO algorithms. To avoid premature convergence, the exploitation of the best solutions has to be combined with effective mechanisms for performing search space exploration. $\mathcal{MM}$AS explicitly addresses this aspect, which is possibly the main reason why it is currently one of the best performing ACO algorithms.

There are several issues which seem to be worth further investigation. At the moment, several ACO algorithms show a promising performance on various combinatorial optimization problems. We strongly believe that future ACO applications will combine features of these ACO algorithms. Here, $\mathcal{MM}$AS may be a very good starting point, since it is one of the best ACO algorithms for combinatorial optimization problems which are often used as benchmarks to test algorithmic ideas. An example for such combinations between several ACO algorithms are the hybrids between $\mathcal{MM}$AS and ACS mentioned above. Further promising ideas are the use of lower bounds on the completion of partial solutions for the computation of the heuristic values as proposed in the ANTS algorithm [29] or the use of ranking for the trail updates [6]. Another issue deals whith the setting of parameters in ACO algorithms. In our experience, the parameters given here for the TSP and QAP applications performed very well over a wide range of instances. Nevertheless, in other applications adaptive versions which dynamically tune the parameters during algorithm execution may increase algorithm robustness. Finally, we definitely need a more thorough understanding of the features the successful application of ACO algorithm depend on and how ACO algorithms should be configured for specific problems. Particularly, the following questions need to be answered: Which solution components should be used? What is the best way of managing the pheromones? Should the ACO algorithm always be combined with local search algorithms? Which problems can be efficiently solved by ACO algorithms? To answer some of these questions, the investigation of search space characteristics and their relation to algorithm performance may give useful insights.

In this article, we have taken initial steps in addressing these issues and provided starting points and directions for further research. It is our hope that by following these routes, ultimately the performance and applicability of ACO algorithms can be further improved.

Acknowledgements

We thank Marco Dorigo and Gianni Di Caro for the fruitful discussions on Ant Colony Optimization. Special thanks also to Olivier Martin for making available

his implementation of the Lin-Kernighan heuristic and for discussions on the subject of this research. A large part of this research was done while both authors were at FG Intellektik, TU Darmstadt in Germany and we thank Wolfgang Bibel for the support during this time, the members for the Intellectics group for discussions, and the members of the FG Inferenzsysteme which allowed us to perform much of the experimental work on their hardware. This work was in part supported by a Marie Curie Fellowship awarded to Thomas Stützle (CEC-TMR Contract No. ERB4001GT973400), a fellowship of the Deutsche Forschunggemeinschaft (DFG) via Graduiertenkolleg ISIA awarded to Thomas Stützle, and a Postdoctoral Fellowship awarded by the University of British Columbia to Holger H. Hoos.

References

[1] J. L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4 (1992) 387–411.

[2] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. Gambardella. Results of the first international contest on evolutionary optimisation. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*, (IEEE Press, Piscataway, USA, 1996) 611–615.

[3] K. D. Boese. *Models for iterative global optimization*. PhD thesis, Computer Science Department, University of California, Los Angeles, USA, 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 (1994) 101–113.

[5] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89 (1999).

[6] B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank based version of the ant system – a computational study. *Central European Journal for Operations Research and Economics*, 7 (1999) 25–38.

[7] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9 (1998) 317–365.

[8] M. Dorigo. *Optimization, learning, and natural algorithms (in Italian)*. PhD thesis, Dip. Elettronica, Politecnico di Milano, Italy, 1992.

[9] M. Dorigo, E. Bonabeau, and G. Theraulaz. *From Natural to Artificial Swarm Intelligence*. *Future Generation Computer Systems*, this issue.

[10] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, (McGraw-Hill, London, UK, 1999) 11–32.

[11] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5 (1999) 137–172.

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

[13] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dip. Elettronica, Politecnico di Milano, Italy, 1991.

[14] 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 (1996) 29–42.

[15] C. Fleurent and J. A. Ferland. Genetic hybrids for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, Vol. 16 (American Mathematical Society, 1994) 173–187.

[16] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*, (IEEE Press, Piscataway, USA, 1996) 616–621.

[17] L. M. Gambardella and M. Dorigo. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, (Morgan Kaufmann, San Francisco, USA, 1995) 252–260.

[18] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*, (IEEE Press, Piscataway, USA, 1996) 622–627.

[19] L. M. Gambardella and M. Dorigo. HAS-SOP: Hybrid ant system for the sequential ordering problem. Technical Report IDSIA 11-97, IDSIA, Lugano, Switzerland, 1997.

[20] L. M. Gambardella, É. D. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, (McGraw-Hill, London, UK, 1999) 63–76.

[21] L. M. Gambardella, É. D. Taillard, and M. Dorigo. Ant colonies for the QAP. *Journal of the Operational Research Society*, 50 (1999) 167–176.

[22] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of $\mathcal{NP}$-completeness*. (Freeman, San Francisco, USA, 1979).

[23] D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, (John Wiley & Sons, Chichester, England, 1997) 215–310.

[24] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L.J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, (Morgan Kaufman, San Francisco, USA, 1995) 184–192.

[25] S. Kirkpatrick and G. Toulouse. Configuration space analysis of travelling salesman problems. *Journal de Physique*, 46 (1985) 1277–1292.

[26] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The traveling salesman problem*. (John Wiley & Sons, Chichester, England), 1985.

[27] S. Lin. Computer solutions for the traveling salesman problem. *Bell Systems Technology Journal*, 44 (1965) 2245–2269.

[28] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the travelling salesman problem. *Operations Research*, 21 (1973) 498–516.

[29] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*. To appear.

[30] V. Maniezzo, M. Dorigo, and A. Colorni. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Université de Bruxelles, Belgium, 1994.

[31] O. Martin and S. W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63 (1996) 57–75.

[32] O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5 (1991) 299–326.

[33] P. Merz and B. Freisleben. Genetic local search for the TSP: New results. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*, (IEEE Press, Piscataway, USA, 1997) 159–164.

[34] P. Merz and B. Freisleben. Fitness Landscapes and Memetic Algorithm Design. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. (McGraw-Hill, London, UK, 1999).

[35] R. Michel and M. Middendorf. An island based ant system with lookahead for the shortest common supersequence problem. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Vol. 1498 (Springer Verlag, Berlin, Germany, 1998) 692–708.

[36] A. Möbius, B. Freisleben, P. Merz, and M. Schreiber. Combinatorial optimization by iterative partial transcription. *Physical Review E*, 59 (1999) 4667–4674.

[37] H. Mühlenbein. Evolution in time and space – the parallel genetic algorithm. In G. J. E. Rawlings, editor, *Foundations of Genetic Algorithms*, (Morgan Kaufmann, San Francisco, USA, 1991) 316–337.

[38] Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*, (Morgan Kaufmann, San Francisco, USA, 1997) 450–457.

[39] H. Ramalhinho Lourenço and D. Serra. Adaptive approach heuristics for the generalized assignment problem. Technical Report Economic Working Papers Series No.304, Universitat Pompeu Fabra, Dept. of Economics and Management, Barcelona, Spain, May 1998.

[40] G. Reinelt. *The traveling salesman: Computational solutions for TSP applications*, Lecture Notes in Computer Science, Vol. 840 (Springer Verlag, Berlin, Germany, 1998).

[41] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23 (1976) 555–565.

[42] P. F. Stadler. Towards a theory of landscapes. Technical Report SFI–95–03–030, Santa Fe Institute, USA, 1995.

[43] T. Stützle. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, vol. 3, (Verlag Mainz, Aachen, Germany, 1998) 1560–1564.

[44] T. Stützle. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the quadratic assignment problem. Technical Report AIDA–97–4, FG Intellektik, TU Darmstadt, Germany, July 1997.

[45] T. Stützle. *Local search algorithms for combinatorial problems — analysis, improvements, and new applications*. PhD thesis, Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany, 1998.

[46] T. Stützle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, (McGraw-Hill, London, UK, 1999) 33–50.

[47] T. Stützle and H. H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for the traveling salesman problem. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*, (IEEE Press, Piscataway, USA, 1997) 309–314.

[48] T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for combinatorial optimization problems. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, (Kluwer Academic Publishers, Boston, 1999) 313–329.

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

[50] É. D. Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3 (1995) 87–105.

[51] 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 travelling salesman problem. In H.-P. Schwefel and R. Männer, editors, *Proceedings 1st International Workshop on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Vol. 496 (Springer Verlag, Berlin, Germany, 1998) 109–116.

[52] T. Walters. Repair and brood selection in the traveling salesman problem. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of Parallel Problem Solving from Nature – PPSN V*, Lecture Notes in Computer Science, Vol. 1498 (Springer Verlag, Berlin, Germany, 1998) 813–822.

[53] E. D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63 (1990) 325–336.

Short Bio Thomas Stützle

Thomas Stützle received the Master of Science in Industrial Engineering and Management Science from the University of Karlsruhe, Germany in 1994 and a PhD in Computer Science from Darmstadt University of Technology in 1998. From 1998 to February 2000 he was working as a Marie Curie fellow at IRIDIA, Université Libre de Bruxelles and is now assistant professor at the Computer Science Department of Darmstadt University of Technology. His research interests are metaheuristics, combinatorial optimization, and empirical analysis of algorithms.

Short Bio Holger Hoos

Holger H. Hoos works on different topics in AI and Computer Music since 1994. He received his Ph.D. 1998 from the Department of Computer Science at the Darmstadt University of Technology (Germany); his Ph.D. thesis on stochastic local search algorithms was recently awarded with the 1999 Best Dissertation Award of the German Informatics Society. Since 1998, Holger Hoos is working as a Postdoctoral Fellow at the University of of British Columbia (Canada).

Correspondence:
Thomas Stützle
IRIDIA
Université Libre de Bruxelles
Avenue Franklin Roosevelt 50
CP 194/6
1050 Brussels
Belgium
Phone: + 32 2 - 6503167
Fax: + 32 2 - 6502715
Email: tstutzle@ulb.ac.be

Co-author:
Holger H. Hoos
University of British Columbia
Computer Science Department
2366 Main Mall
Vancouver, BC, V6T 1Z4
Canada
Phone: + 1 604 - 822-5109
Fax: + 1 604 - 822-5485
Email: hoos@cs.ubc.ca