1 ACO Algorithms for the Traveling Salesman Problem[†]

Thomas STÜTZLE[‡] and Marco DORIGO
IRIDIA, Université Libre de Bruxelles, Belgium
{tstutzle,mdorigo}@ulb.ac.be

1.1 INTRODUCTION

Ant algorithms [18, 14, 19] are a recently developed, population-based approach which has been successfully applied to several $\mathcal{NP}$-hard combinatorial optimization problems [6, 13, 17, 23, 34, 40, 49]. As the name suggests, ant algorithms have been inspired by the behavior of real ant colonies, in particular, by their foraging behavior. One of the main ideas of ant algorithms is the indirect communication of a colony of agents, called (artificial) ants, based on pheromone trails (pheromones are also used by real ants for communication). The (artificial) pheromone trails are a kind of distributed numeric information which is modified by the ants to reflect their experience while solving a particular problem. Recently, the Ant Colony Optimization (ACO) meta-heuristic has been proposed which provides a unifying framework for most applications of ant algorithms [15, 16] to combinatorial optimization problems. In particular, all the ant algorithms applied to the TSP fit perfectly into the ACO meta-heuristic and, therefore, we will call these algorithms also ACO algorithms.

The first ACO algorithm, called Ant System (AS) [18, 14, 19], has been applied to the Traveling Salesman Problem (TSP). Starting from Ant System, several improvements of the basic algorithm have been proposed [21, 22, 17, 51, 53, 7]. Typically, these improved algorithms have been tested again on the TSP. All these improved versions of AS have in common a stronger exploita-

[†]To appear in K. Miettinen and M.M. Mäkelä, P. Neittaanmäki, J. Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*. John Wiley & Sons, 1999.
[‡]Presently on leave from FG Intellektik, TU Darmstadt, Germany.

tion of the best solutions found to direct the ants' search process; they mainly differ in some aspects of the search control. Additionally, the best performing ACO algorithms for the TSP [17, 49] improve the solutions generated by the ants using local search algorithms.

In this paper we give an overview on the available ACO algorithms for the TSP. We first introduce, in Section 1.2, the TSP. In Section 1.3 we outline how ACO algorithms can be applied to that problem and present the available ACO algorithms for the TSP. Section 1.4 briefly discusses local search for the TSP, while Section 1.5 presents experimental results which have been obtained with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, one of the improved versions of Ant System. Since the first application of ACO algorithms to the TSP, they have been applied to several other combinatorial optimization problems. On many important problems ACO algorithms have proved to be among the best available algorithms. In Section 1.6 we give a concise overview of these other applications of ACO algorithms.

1.2 THE TRAVELING SALESMAN PROBLEM

The TSP is extensively studied in literature [29, 31, 45] and has attracted since a long time a considerable amount of research effort. The TSP also plays an important role in Ant Colony Optimization since the first ACO algorithm, called Ant System [18, 14, 19], as well as many of the subsequently proposed ACO algorithms [21, 17, 52, 53, 7] have initially been applied to the TSP. The TSP was chosen for many reasons: (*i*) it is a problem to which ACO algorithms are easily applied, (*ii*) it is an $\mathcal{NP}$-hard [26] optimization problem, (*iii*) it is a standard test-bed for new algorithmic ideas and a good performance on the TSP is often taken as a proof of their usefulness, and (*iv*) it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities.

Intuitively, the TSP is the problem of a salesman who wants to find, starting from his home town, a shortest possible trip through a given set of customer cities and to return to its home town. More formally, it can be represented by a complete weighted graph $G = (N, A)$ with N being the set of nodes, representing the cities, and A the set of arcs fully connecting the nodes N. Each arc is assigned a value d_{ij}, which is the length of arc $(i,j) \in A$, that is, the distance between cities i and j, with $i,j \in N$. The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where an Hamiltonian circuit is a closed tour visiting exactly once each of the $n = |N|$ nodes of G. 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 more general asymmetric TSP (ATSP) at least for one pair of nodes i,j we have $d_{ij} \neq d_{ji}$.

In case of symmetric TSPs, we will use *Euclidean* TSP instances in which the cities are points in the Euclidean space and the inter-city distances are calculated using the Euclidean norm. All the TSP instances we use are taken from the TSPLIB Benchmark library [44] which contains a large collection of instances; these have been used in many other studies or stem from practical applications of the TSP. TSPLIB is accessible on the WWW at the address http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html

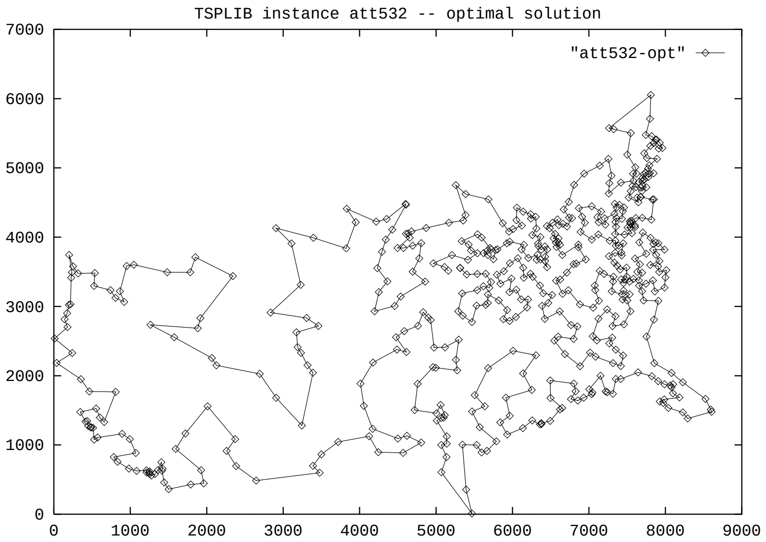

Fig. 1.1 TSP instance `att532` from TSPLIB; it comprises 532 cities in the USA. The given tour is of length 27686, which in fact is an optimal tour.

1.3 AVAILABLE ACO ALGORITHMS FOR THE TSP

1.3.1 Applying ACO algorithms to the TSP

In ACO algorithms ants are simple agents which, in the TSP case, construct tours by moving from city to city on the problem graph. The ants' solution construction is guided by (artificial) pheromone trails and an a priori available heuristic information. When applying ACO algorithm to the TSP, a pheromone strength $\tau_{ij}(t)$ is associated to each arc (i, j), where $\tau_{ij}(t)$ is a numerical information which is modified during the run of the algorithm and t is the iteration counter. If an ACO algorithm is applied to symmetric TSP instances, we always have $\tau_{ij}(t) = \tau_{ji}(t)$; in applications to asymmetric TSPs (ATSPs), we will possibly have $\tau_{ij}(t) \neq \tau_{ji}(t)$.

Initially, each of the m ants is placed on a randomly chosen city and then iteratively applies at each city a state transition rule. An ant constructs a tour as follows. At a city i, the ant chooses a still unvisited city j probabilistically, biased by the pheromone trail strength $\tau_{ij}(t)$ on the arc between city i and city j and a locally available heuristic information, which is a function of the arc length. Ants probabilistically prefer cities which are close and are connected by arcs with a high pheromone trail strength. To construct a feasible solution each ant has a limited form of memory, called *tabu list*, in which the current partial tour is stored. The memory is used to determine at each construction step the set of cities which still has to be visited and to guarantee that a feasible solution is built. Additionally, it allows the ant to retrace its tour, once it is completed.

After all ants have constructed a tour, the pheromones are updated. This is typically done by first lowering the pheromone trail strengths by a constant factor and then the ants are allowed to deposit pheromone on the arcs they have visited. The trail update is done in such a form that arcs contained in shorter tours and/or visited by many ants receive a higher amount of pheromone and are therefore chosen with a higher probability in the following iterations of the algorithm. In this sense the amount of pheromone $\tau_{ij}(t)$ represents the learned desirability of choosing next city j when an ant is at city i.

The best performing ACO algorithms for the TSP improve the tours constructed by the ants applying a local search algorithm [17, 49, 53]. Hence, these algorithms are in fact hybrid algorithms combining probabilistic solution construction by a colony of ants with standard local search algorithms. Such a combination may be very useful since constructive algorithms for the TSP often result in a relatively poor solution quality compared to local search algorithms [29]. Yet, it has been noted that repeating local search from randomly generated initial solutions results in a considerable gap to the optimal solution [29]. Consequently, on the one hand local search algorithms may improve the tours constructed by the ants, while on the other hand the ants may guide the local search by constructing promising initial solutions. The initial solutions generated by the ants are promising because the ants use with higher probability those arcs which, according to the search experience, have more often been contained in shorter tours.

In general, all the ACO algorithms for the TSP follow a specific algorithmic scheme, which is outlined in Figure 1.2. 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 tours, then the generated tours are improved by applying local search, and finally the pheromone trails are updated. In fact, most of the best performing ACO algorithms for $\mathcal{NP}$-hard combinatorial optimization problems follow this algorithmic scheme [17, 23, 34, 49, 53]. It must be noted that the ACO meta-heuristic [15, 16] is more general than the algorithmic scheme given above. For example, the later does not capture the application of ACO algorithms to network routing problems.

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

Fig. 1.2 Algorithmic skeleton for ACO algorithm applied to the TSP.

1.3.2 Ant System

When Ant System (AS) was first introduced, it was applied to the TSP. Initially, three different versions of AS were proposed [14, 9, 18]; these were called *ant-density*, *ant-quantity*, and *ant-cycle*. While in *ant-density* and *ant-quantity* the ants updated the pheromone directly after a move from a city to an adjacent one, in *ant-cycle* the pheromone update was only done after all the ants had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality. Because *ant-cycle* performed much better than the other two variants, here we only present the ant-cycle algorithm, referring to it as Ant System in the following. In AS each of m (artificial) ants builds a solution (tour) of the TSP, as described before. In AS no local search is applied. (Obviously, it would be straightforward to add local search to AS.)

Tour construction. Initially, each ant is put on some randomly chosen city. At each construction step, ant k applies a probabilistic action choice rule. In particular, the probability with which ant k, currently at city i, chooses to go to city j at the tth iteration of the algorithm is:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{if } j \in \mathcal{N}_i^k \tag{1.1}$$

where $\eta_{ij} = 1/d_{ij}$ is an a priori available heuristic value, α and β are two parameters which determine the relative influence 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 yet visited. The role of the parameters α and β is the following. If $\alpha = 0$, the closest cities are more likely to be selected: this corresponds to a classical stochastic greedy algorithm (with multiple starting points since ants are initially randomly distributed on the cities). If $\beta = 0$, only pheromone amplification is at work: this method will lead to the rapid emergence of a *stagnation* situation with the corresponding generation of tours which, in general, are strongly suboptimal [14]. Search stagnation is defined in [19] as the situation where all the ants follow the same path and construct the same solution. Hence, a tradeoff between the influence of the heuristic information and the pheromone trails exists.

Pheromone update. After all ants have constructed their tours, the pheromone trails are updated. This is done by first lowering the pheromone strength on *all* arcs by a constant factor and then allowing each ant to add pheromone on the arcs it has visited:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{1.2}$$

where $0 < \rho \leq 1$ is the pheromone trail evaporation. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to "forget" previously done bad decisions. If an arc is not chosen by the ants, its associated pheromone strength decreases exponentially. $\Delta\tau_{ij}^{k}(t)$ is the amount of pheromone ant k puts on the arcs it has visited; it is defined as follows:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^k(t) & \text{if arc } (i,j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

where $L^k(t)$ is the length of the kth ant's tour. By Equation 1.3, the better the ant's tour is, the more pheromone is received by arcs belonging to the tour. In general, arcs which are used by many ants and which are contained in shorter tours will receive more pheromone and therefore are also more likely to be chosen in future iterations of the algorithm.

AS has been compared with other general purpose heuristics on some relatively small TSP instances with maximally 75 cities. Despite encouraging initial results, for larger instances AS gives a rather poor solution quality. Therefore, a substantial amount of recent research in ACO has focused on improvements over AS. A first improvement on the initial form of AS, called the *elitist strategy* for Ant System, has been introduced in [14, 19]. The idea is to give a strong additional reinforcement to the arcs belonging to the best tour found since the start of the algorithm; this tour is denoted as T^{gb} (global-best tour) in the following. This is achieved by adding to the arcs of tour T^{gb} a quantity $e \cdot 1/L^{gb}$, where e is the number of elitist ants, when the pheromone trails are updated according to Equation 1.2. Some limited results presented in [14, 19] suggest that the use of the elitist strategy with an appropriate number of elitist ants allows AS: (i) to find better tours, and (ii) to find them earlier in the run. Yet, for too many elitist ants the search concentrates early around suboptimal solutions leading to an early stagnation of the search.

1.3.3 Ant Colony System

Ant Colony System (ACS) has been introduced in [17, 22] to improve the performance of AS. ACS is based on an earlier algorithm proposed by the same authors, called Ant-Q, which exploited connections of ACO algorithms with Q-learning [55], a specific type of reinforcement learning algorithm. ACS and Ant-Q differ only in one specific aspect, which is described below. We concentrate on ACS, since it is somewhat simpler and it is preferred by its authors.

ACS differs in three main aspects from ant system. First, ACS uses a more aggressive action choice rule than AS. Second, the pheromone is added only to arcs belonging to the global-best solution. Third, each time an ant uses an arc (i, j) to move from city i to city j it removes some pheromone from the arc. In the following we present these modifications in more detail.

Tour construction. In ACS ants choose the next city using the *pseudo-random-proportional action choice rule*: When located at city i, ant k moves, with probability q_0, to city l for which $\tau_{il}(t) \cdot [\eta_{il}]^\beta$ is maximal, that is, with probability q_0 the best possible move as indicated by the learned pheromone trails and the heuristic information is made (exploitation of learned knowledge). With probability $(1 - q_0)$ an ant performs a biased exploration of the arcs according to Equation 1.1.

Global pheromone trail update. In ACS only the global best ant is allowed to add pheromone after each iteration. Thus, the update according to Equation 1.2 is modified to

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}^{gb}(t), \tag{1.4}$$

where $\Delta\tau_{ij}^{gb}(t) = 1/L^{gb}$. It is important to note that the trail update only applies to the arcs of the global-best tour, not to all the arcs like in AS. The parameter ρ again represents the pheromone evaporation. In ACS only the global best solution receives feedback. Initially, also using the iteration best solution was considered for the pheromone update. Although for smaller TSP instances the difference in solution quality between using the global-best solution or the iteration-best solution is minimal, for larger instances the use of the global-best tour gives by far better results.

Local pheromone trail update. Additionally to the global updating rule, in ACS the ants use a local update rule that they apply immediately after having crossed an arc during the tour construction:

$$\tau_{ij} = (1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0 \tag{1.5}$$

where ξ, $0 < \xi < 1$, and τ_0 are two parameters. The effect of the local updating rule is to make an already chosen arc less desirable for a following ant. In this way the exploration of not yet visited arcs is increased.

The only difference between ACS and Ant-Q is the definition of the term τ_0. Ant-Q uses a formula for τ_0 which was inspired by Q-learning [55]. In Ant-Q the term τ_0 corresponds to the discounted evaluation of the next state and is set to $\gamma \cdot \max_{l \in \mathcal{N}_j^k} \{\tau_{jl}\}$ [21]. It was found that replacing the discounted evaluation of the next state by a small constant resulted in approximately the same performance and therefore, due to its simplicity, ACS is preferably used.

1.3.4 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [52, 51] is a direct improvement over AS. The solutions in $\mathcal{MM}$AS are constructed in exactly the same way as in AS, that is, the selection probabilities are calculated as in Equation 1.1. Additionally, in [53] a variant of $\mathcal{MM}$AS is considered which uses the pseudo-random-proportional action choice rule of ACS. Using that action choice rule, very good solutions could be found faster, but the final solution quality achieved was worse.

The main modifications introduced by $\mathcal{MM}$AS with respect to AS are the following. First, to exploit the best solutions found, after each iteration only one ant (like in ACS) is allowed to add pheromone. Second, to avoid search stagnation, the allowed range of the pheromone trail strengths is limited to the interval $[\tau_{\min}, \tau_{\max}]$, that is, $\forall \tau_{ij}\ \tau_{\min} \leq \tau_{ij} \leq \tau_{\max}$. Last, the pheromone trails are initialized to the upper trail limit, which causes a higher exploration at the start of the algorithm [49].

Update of pheromone trails. After all ants have constructed a solution, the pheromone trails are updated according to

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{1.6}$$

where $\Delta\tau_{ij}^{best} = 1/L^{best}$. The ant which is allowed to add pheromone may be the *iteration-best* solution T^{ib}, or the *global-best* solution T^{gb}. Hence, if specific arcs are often used in the best solutions, they will receive a larger amount of pheromone. Experimental results have shown that the best performance is obtained by gradually increasing the frequency of choosing T^{gb} for the trail update [49].

Trail limits. In $\mathcal{MM}$AS lower and upper limits on the possible pheromone strengths on any arc are imposed to avoid search stagnation. In particular, the imposed trail limits have the effect of indirectly limiting the probability

p_{ij} of selecting a city j when an ant is in city i to an interval $[p_{\min}, p_{\max}]$, with $0 < p_{\min} \leq p_{ij} \leq p_{\max} \leq 1$. Only if an ant has one single possible choice for the next city, then $p_{\min} = p_{\max} = 1$. Experimental results [49] suggest that the lower trail limits used in $\mathcal{MM}$AS are the more important ones, since the maximum possible trail strength on arcs is limited in the long run due to pheromone evaporation.

Trail initialization. The pheromone trails in $\mathcal{MM}$AS are initialized to their upper pheromone trail limits. Doing so the exploration of tours at the start of the algorithms is increased, since the relative differences between the pheromone trail strengths are less pronounced.

1.3.5 Rank-Based Version of Ant System

Another improvement over Ant System is the *rank-based* version of Ant System (AS_{rank}) [7]. In AS_{rank}, always the global-best tour is used to update the pheromone trails, similar to the elitist strategy of AS. Additionally, a number of the best ants of the current iteration are allowed to add pheromone. To this aim the ants are sorted by tour length ($L^1(t) \leq L^2(t) \leq \ldots \leq L^m(t)$), and the quantity of pheromone an ant may deposit is weighted according to the rank r of the ant. Only the $(w-1)$ best ants of each iteration are allowed to deposit pheromone. The global best solution, which gives the strongest feedback, is given weight w. The rth best ant of the current iteration contributes to pheromone updating with a weight given by $\max\{0, w-r\}$. Thus the modified update rule is:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{r=1}^{w-1} (w-r) \cdot \Delta\tau_{ij}^{r}(t) + w \cdot \Delta\tau_{ij}^{gb}(t), \tag{1.7}$$

where $\Delta\tau_{ij}^{r}(t) = 1/L^r(t)$ and $\Delta\tau_{ij}^{gb}(t) = 1/L^{gb}$.

In [7] AS_{rank} has been compared to AS, AS with elitist strategy, to a genetic algorithm, and to a simulated annealing algorithm. For the larger TSP instances (the largest instance considered had 132 cities) the AS-based approaches showed to be superior to the genetic algorithm and the simulated annealing procedure. Among the AS-based algorithms, both, AS_{rank} and elitist AS performed significantly better than AS, with AS_{rank} giving slightly better results than elitist AS.

1.3.6 Synopsis

Obviously, the proposed ACO algorithms for the TSP share many common features. Ant System can mainly be seen as a first study to demonstrate the viability of ACO algorithms to attack $\mathcal{NP}$-hard combinatorial optimization problems, but its performance compared to other approaches is rather poor. Therefore, several ACO algorithms have been proposed which strongly increase the performance of Ant System. The main common feature among the proposed improvements is that they exploit more strongly the best solutions found during the ants' search. This is typically achieved by giving a higher weight to better solutions for the pheromone update and often allowing to deposit additional pheromone trail on the arcs of the global-best solution. In particular, in elitist AS and in AS$_{rank}$ the global best solution adds a strong additional reinforcement (in AS$_{rank}$ additionally some of the best ants of an iteration add pheromone); in ACS and $\mathcal{MM}$AS only one single ant, either the global-best or the iteration-best ant, deposit pheromone. Obviously, by exploiting the best solutions more strongly, arcs contained in the best tours receive a strong additional feedback and therefore are chosen with higher probability in subsequent algorithm iterations.

Yet, a problem associated with the stronger exploitation of search experience may be search stagnation, that is, the situation in which all ants follow the same path. Hence, some of the proposed ACO algorithms, in particular ACS and $\mathcal{MM}$AS, introduce additional features to avoid search stagnation. In ACS stagnation is avoided by the local updating rule which decreases the amount of pheromone on an arc and makes this arc less and less attractive for following ants. In this way the exploration of not yet visited arcs is favored. In $\mathcal{MM}$AS search stagnation is avoided by imposing limits on the allowed pheromone strength associated with an arc. Hence, since the pheromone trail limits influence directly the selection probabilities given by Equation 1.1, the selection probability for an arc cannot fall below some lower value. By appropriate choices for the trail limits, it is very unlikely that all ants follow the same path.

All proposed improvements over AS show significantly better performance on the TSP [19, 7, 21, 49] and they all make use of a stronger exploitation of the best solutions found by the ants. It is therefore reasonable to think that the concentration of the search around the best solutions found during the search is the key to the improved performance shown by these algorithms.

The characteristics of the TSP search space may explain this phenomenon. In particular, recent research has addressed the issue of the correlation between the solution cost and the distance from very good or from the optimal solution [4, 3] (an obvious distance measure for the TSP is given by the number of different arcs in two tours). Similarly, the Fitness-Distance Correlation (FDC) has been introduced in the context of genetic algorithm research [30]. The FDC measures the correlation between the distance of a solution from the closest optimal solution and the solution costs. Plots of the solution cost versus the distance from optimal solutions have been particularly useful to visualize the correlation. In Figure 1.3 we present such a plot for a randomly generated Euclidean TSP instance with 500 cities. The plot is based on 5000 local optima which have been obtained by a `3-opt` local search algorithm (see next section). The x-axis gives the distance from the optimal solution, while on the y-axis the tour length is given. Obviously, a significant correlation between the solution cost and the distance from the global optimum exists (the correlation coefficient is 0.52); the closer a local optimum is to the known global optimum, the better, on average, is the solution.

The task of algorithms like ACO algorithms is to guide the search towards regions of the search space containing high quality solutions and, possibly, the global optimum. Clearly, the notion of *search space region* is tightly coupled to the notion of distance, defined by an appropriate distance measure, between solutions. The most important guiding mechanism of ACO algorithms is the objective function value of solutions; the better a solution the more feedback it is allowed to give. This guiding mechanism relies on the general intuition that the better a solution is, the more likely it is to find even better solutions close to it. The fitness-distance correlation describes this intuitive relation between the fitness (cost) of solutions and their distance from very good solutions or from the global best solution. Since for the TSP such a correlation obviously exists, it appears to be a good idea to concentrate the search around the best solutions found.

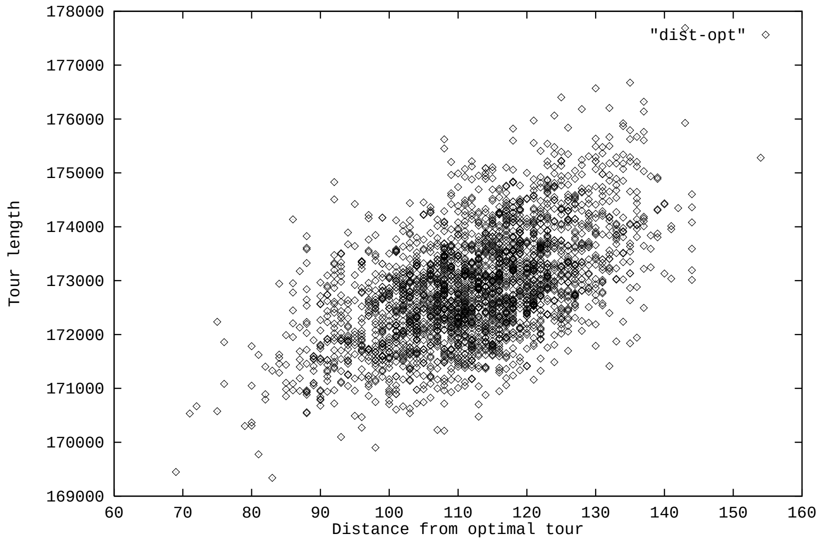

Fig. 1.3 Fitness distance plot for a Euclidean TSP instances with 500 cities distributed randomly according to an uniform distribution on a square. The plot is based on 5000 `3-opt` tours. The distance from the optimal tour is measured by the number of different arcs.

1.4 LOCAL SEARCH FOR THE TSP

Local Search starts from some initial assignment and repeatedly tries to improve the current assignment by local changes. If in the neighborhood of the current tour T a better tour T' is found, it replaces the current tour and the local search is continued from T'. The most widely known iterative improvement algorithms for the TSP are certainly `2-opt` [12] and `3-opt` [32]. They proceed by systematically testing whether the current tour can be improved by replacing 2 or at most 3 arcs, respectively. Both local search algorithms are widely studied in the literature [45, 29] and have been shown empirically to yield good solution quality. Local search algorithms using $k > 3$ arcs to be exchanged are not used commonly, due to the high computation times involved and the low additional increase in solution quality. Already for `2-opt` and `3-opt` a straightforward implementation would require $\mathcal{O}(n^2)$ or $\mathcal{O}(n^3)$ exchanges to be examined. This is clearly infeasible when trying to solve instances with several hundreds of cities in reasonable computation time.

Fortunately, there exist quite a few speed-up techniques [1, 45, 29] achieving, in practice, run-times which grow sub-quadratically. This effect is obtained by examining only a small part of the whole neighborhood. We use three techniques to reduce the run-time of `2-opt` and `3-opt` implementations. One, consists in restricting the set of moves which are examined to those contained in a candidate list of the nearest neighbors ordered according to nondecreasing distances [1, 33, 45]. Using candidate lists, for a given starting node i we only consider moves which add a new arc between i and one of the nodes in its candidate list. Hence, by using a neighborhood list of bounded length, an improving move can be found in constant time.

An additional speed-up is achieved by performing a fixed radius nearest neighbor search [1]. For `2-opt` at least one newly introduced arc has to be shorter than any of the two removed arcs (i, j) and (k, l). Due to symmetry reasons, it suffices to check whether $d_{ij} > d_{ik}$. A similar argument also holds for `3-opt` [1].

To yield a further reduction in run-time we use *don't look bits* associated with each node. Initially, all don't look bits are turned off (set to 0). If for a node no improving move can be found, the don't look bit is turned on (set to 1) and the node is not considered as a starting node for finding an improving move in the next iteration. In case an arc incident to a node is changed by a move, the node's don't look bit is turned off again.

For asymmetric TSPs, `2-opt` is not directly applicable because one part of the tour has to be traversed in the opposite direction. In case a sub-tour is reversed, the length of this sub-tour has to be computed from scratch. Yet, one of the three possible `3-opt` moves does not lead to a reversal of a sub-tour. We call `reduced 3-opt` this special form of `3-opt`. The implementation of `reduced 3-opt` uses the same speed-up techniques as described before.

The local search algorithm producing the highest quality solutions for symmetric TSPs is the Lin-Kernighan heuristic (`LK`) [33]. The Lin-Kernighan heuristic considers in each step a variable number of arcs to be exchanged. Yet, a disadvantage of the `LK` heuristic is that its implementation is more difficult than that of `2-opt` and `3-opt`, and careful fine-tuning is necessary to get it to run fast and produce high quality solutions [45, 29].

1.5 EXPERIMENTAL RESULTS

In this section we present some computational results obtained with $\mathcal{MM}$AS on some symmetric and asymmetric TSP instances from TSPLIB. Since we apply $\mathcal{MM}$AS to larger TSP instances with several hundreds of cities, we first present two techniques which decrease the complexity of the ACO algorithm iterations.

1.5.1 Additional techniques

Candidate lists. Each tour construction has complexity $O(n^2)$, which would lead to substantial run-times for larger instances. A standard technique used when solving large TSPs is the use of candidate lists [29, 38, 45]. Typically, a candidate list comprises a fixed number of nearest neighbors for each city in order of nondecreasing distances. The use of candidate lists for the tour construction by the ants has first been proposed for the application of ACS [22] to the TSP. When constructing a tour an ant chooses the next city among those of 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. Note that using nearest neighbor lists is a reasonable approach when trying to solve TSPs. For example, in many TSP instances the optimal tour can be found within a surprisingly low number of nearest neighbors. For example, an optimal solution is found for instance `pcb442.tsp`, which has 442 cities, within a subgraph of the 6 nearest neighbors and for instance `pr2392.tsp` (2392 cities) within a subgraph of the 8 nearest neighbors [45].

Fastening the trail update. When applying ACO algorithms to the TSP, pheromone trails are stored in a matrix with $\mathcal{O}(n^2)$ entries (one for each arc). All the entries of this matrix should be updated at every iteration because of pheromone trail evaporation implemented by Formula 1.2 (the update of the whole trail matrix does not happen in ACS). Obviously, this is a very expensive operation if large TSP instances should be solved. To avoid this, in $\mathcal{MM}$AS pheromone evaporation is applied only to arcs connecting a city i to cities belonging to i's candidate list. Hence, the pheromone trails can be updated in $\mathcal{O}(n)$.

1.5.2 Parameter settings

Suitable parameter settings were determined in some preliminary experiments. We use $m = 25$ ants (all ants apply a local search to their solution), $\rho = 0.2$, $\alpha = 1$, and $\beta = 2$. During the tour construction the ants use a candidate list of size 20. The most important aspect concerning the allowed interval of the trail values is that they have to be in some reasonable interval around the expected amount of pheromone deposited by the trail update. Hence, the interval for the allowed values of the pheromone trail strength is determined to reflect this intuition. We set $\tau_{\max} = \frac{1}{\rho} \cdot \frac{1}{T^{gb}}$, where T^{gb} is the tour length of the global best tour found (this setting corresponds to the maximum possible trail level for longer runs). The lower trail limit is chosen as $\tau_{\min} = \tau_{\max}/2 \cdot n$, where n is the number of cities of an instance.

As indicated in Section 1.3.4, the best results with $\mathcal{MM}$AS are obtained when the frequency with which the global-best solution is used to update pheromone trails increases at run-time. To do so, we apply a specific schedule to alternate the pheromone trail update between T^{gb} and T^{ib}. Let u^{gb} indicate that every u^{gb} iterations T^{gb} is chosen to deposit pheromone. In the first 25 iterations only T^{ib} is used to update the trails; we set u^{gb} to 5 if $25 < t < 75$ (where t is the iteration counter), to 3 if $75 < t < 125$ to 3 (where t is the iteration counter), to 2 if $125 < t < 250$ to 2, and from then on to 1. By shifting the emphasis from the iteration-best to the global-best solution for the trail update, we in fact are shifting from a stronger exploration of the search space to an exploitation of the best solution found so far. Additionally, we reinitialize the pheromone trails to $\tau_{\max}$ if the pheromone trail strengths on almost all arcs not contained in T^{gb} is very close to $\tau_{\min}$ (as indicated by the average branching factor [21]). After a reinitialization of the pheromone trails, the schedule is applied from the start again.

For the local search we use `3-opt`, enhanced by the speed-up techniques described in Section 1.4. The size of the candidate list was set to 40 for the local search, following recommendations of [29]. Initially, all the don't look bits are turned off.

1.5.3 Experimental results for symmetric TSPs

In this section we report on experimental results obtained with $\mathcal{MM}$AS, on some symmetric TSP instances from TSPLIB. Most of the instances were proposed as benchmark instances in the First International Contest on Evolutionary Computation (1st ICEO) [2]. We compare the computational results obtained with $\mathcal{MM}$AS to a standard iterated local search algorithm [38, 37, 29] for the TSP using the same `3-opt` implementation and the same maximal computation time t_{max} for each trial as $\mathcal{MM}$AS.

Iterated local search (ILS) [38, 37, 29] is well known to be among the best algorithms for the TSP. In ILS a local search is applied iteratively to starting solutions which are obtained by mutations of a previously found local optimal solution, typically the best solution found so far. In particular, the ILS algorithm we use for comparison first locally optimizes an initial tour produced by the nearest neighbor heuristic. Then it applies iteratively `3-opt` to initial solutions which are obtained by the application of random, sequential 4-opt moves (called double-bridge moves in [38]) to the best solution found so far. The runs are performed on a Sun UltraSparc II Workstation with two UltraSparc I 167MHz processors with 0.5MB external cache. Due to the sequential implementation of the algorithm only one processor is used.

The computational results show that $\mathcal{MM}$AS is able to find very high quality solutions for all instances and on most instances $\mathcal{MM}$AS achieves a better average solution quality than ILS. This result is very encouraging, since ILS is cited in the literature as a very good algorithm for the TSP [37, 29].

Table 1.1 Comparison of $\mathcal{MM}$AS with iterated `3-opt` (ILS) applied to some symmetric TSP instances (available in TSPLIB). 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 time t_{avg} to find the best solution in a run, and the maximally 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 in boldface.

Instance	Algorithm	Best	Average	Worst	t_{avg}	t_{max}
d198	$\mathcal{MM}$AS	15780	15780.4 (0.0)	15784	61.7	170
	ILS	15780	**15780.2 (0.0)**	15784	18.6	
lin318	$\mathcal{MM}$AS	42029	**42029.0 (0.0)**	42029	94.2	450
	ILS	42029	42064.6 (0.09)	42163	55.6	
pcb442	$\mathcal{MM}$AS	50778	**50911.2 (0.26)**	51047	308.9	600
	ILS	50778	50917.7 (0.28)	51054	180.7	
att532	$\mathcal{MM}$AS	27686	**27707.9 (0.08)**	27756	387.3	1250
	ILS	27686	27709.7 (0.09)	27759	436.2	
rat783	$\mathcal{MM}$AS	8806	**8814.4 (0.10)**	8837	965.2	2100
	ILS	8806	8828.4 (0.34)	8850	1395.2	
u1060	$\mathcal{MM}$AS	224455	224853.5 (0.34)	225131	2577.6	3600
	ILS	224152	**224408.4 (0.14)**	224743	1210.4	
pcb1173	$\mathcal{MM}$AS	56896	**56956.0 (0.11)**	57120	3219.5	5400
	ILS	56897	57029.5 (0.24)	57251	1892.0	
d1291	$\mathcal{MM}$AS	50801	**50821.6 (0.04)**	50838	1894.4	5400
	ILS	50825	50875.7 (0.15)	50926	1102.9	
fl1577	$\mathcal{MM}$AS	22286	**22311.0 (0.28)**	22358	3001.8	7200
	ILS	22311.0	22394.6 (0.65)	22505	3447.6	

Table 1.2 Comparison of $\mathcal{MM}$AS with iterated `3-opt` (ILS) applied to some asymmetric TSP instances (available in TSPLIB). 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 with 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 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 in boldface.

Instance	Algorithm	Best	Average	Worst	t_{avg}	t_{max}
ry48p	$\mathcal{MM}$AS	14422	**14422.0 (0.0)**	14422	2.3	120
	ILS	14422	14425.6 (0.03)	14507	24.6	
ft70	$\mathcal{MM}$AS	38673	**38673.0 (0.0)**	38673	37.2	300
	ILS	38673	38687.9 (0.04)	38707	3.1	
kro124p	$\mathcal{MM}$AS	36230	**36230.0 (0.0)**	36230	7.3	300
	ILS	36230	36542.5 (0.94)	37114	23.4	
ftv170	$\mathcal{MM}$AS	2755	**2755.0 (0.0)**	2755	56.2	600
	ILS	2755	2756.8 (0.07)	2764	21.7	

1.5.4 Experimental results for asymmetric TSPs

We applied the same algorithms also to the more general asymmetric TSP; the computational results are given in Table 1.2. On the asymmetric instances the performance difference between $\mathcal{MM}$AS and ILS becomes more pronounced. While $\mathcal{MM}$AS solves all instances in all runs to optimality, ILS gets stuck at suboptimal solutions in all the four instances tested. Among the instances tested, `ry48p` and `kro124p` are easily solved by $\mathcal{MM}$AS, while on these the relatively poor performance of ILS is most striking. Only the two instances `ft70` and `ftv170` are somewhat harder. Computational results from other researchers suggest that in particular instance `ft70` is, considering its small size, relatively hard to solve [39, 54, 17].

1.5.5 Related work on the TSP

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 [29].

Currently, the iterated `LK` heuristic (ILK), that is, ILS using the `LK` heuristic for the local search, is the most efficient approach to symmetric TSPs for short to medium run-times [29]. Recently, several new approaches and improved implementations have been presented which appear to perform as well or better than ILK for larger run-times. Among these algorithms we find the genetic local search approach of Merz and Freisleben [20, 39], an improved implementation of ASPARAGOS (one of the first genetic local search approaches to the TSP) of Gorges-Schleuter [27], a new genetic local search approach using a repair-based crossover operator and brood selection by Walters [54], a genetic algorithm using a specialized crossover operator, called edge assembly crossover, due to Nagata and Kobayashi [42], and finally a specialized local search algorithm for the TSP called Iterative Partial Transcription by Möbius et.al. [41]. Some of these algorithms [39, 41, 42, 54] achieve better computational results than the ones presented here. For example, the genetic local search approach presented in [39], 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. Yet, the computational results with $\mathcal{MM}$AS also would benefit from the application of a more powerful local search algorithm like the `LK` heuristic. In fact, some initial experimental results presented in [49] confirm this conjecture, but still the computation times are relatively high compared to, for example, the results given in [39].

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 [39] and the same as reported in [27, 54], but at the cost of higher run-times.

1.6 OTHER APPLICATIONS OF ACO ALGORITHMS

Research on ACO algorithms, and on ant algorithms more in general, has led to a number of other successful applications to combinatorial optimization problems. We call ant algorithm those algorithms that take inspiration by some aspects of social insects behavior. In general, ant algorithms are characterized by being multi-agent systems using only local information to make stochastic decisions and interacting with each other via a form of indirect communication called stigmergy [28]. A general introduction to ant algorithms can be found in Bonabeau, Dorigo, and Theraulaz [5]. In this paper we limit our attention to ACO algorithms, a particular class of ant algorithms which follow the meta-heuristic described in Dorigo, Di Caro, and Gambardella [16] and Dorigo and Di Caro [15].

The most widely studied problems using ACO algorithms are the traveling salesman problem and the quadratic assignment problem (QAP). Applications of ACO algorithms to the TSP have been reviewed in this paper. As in the TSP case, the first application of an ACO algorithm to the QAP has been that of Ant System [36]. In the last two years several ant and ACO algorithms for the QAP have been presented by Maniezzo and Colorni [35], Maniezzo [34], Gambardella, Taillard, and Dorigo [25], and Stützle [48]. Currently, ant algorithms are among the best algorithms for attacking real-life QAP instances. We refer to [50] for an overview of these approaches.

The sequential ordering problem is closely related to the asymmetric TSP, but additional precedence constraints between the nodes have to be satisfied. Gambardella and Dorigo have tackled this problem by an extension of ACS enhanced by a local search algorithm [23]. They obtained excellent results and were able to improve the best known solutions for many benchmark instances.

A first application of AS to the job-shop scheduling problem has been presented in [10]. Despite showing the viability of the approach, the computational results are not competitive with state-of-the-art algorithms. More recently, $\mathcal{MM}$AS has been applied to the flow shop scheduling problem (FSP) in [47]. The computational results have shown that $\mathcal{MM}$AS outperforms earlier proposed Simulated Annealing algorithms and performs comparably to Tabu Search algorithms applied to the FSP.

Costa and Herz [11] have proposed an extension of AS to assignment type problems and present an application of their approach to the graph coloring problem obtaining results comparable to those obtained by other heuristic approaches.

Applications of AS$_{rank}$ to vehicle routing problems are presented by Bullnheimer, Hartl, and Strauss [8, 6]. They obtained good computational results for standard benchmark instances, slightly worse than the best performing Tabu Search algorithms. A recent application by Gambardella, Taillard, and Agazzi to vehicle routing problems with time windows, improves the best known solutions for some benchmark instances [24].

A further application of AS to the shortest common supersequence problem has been proposed by Michel and Middendorf [40]. They introduce the novel aspect of using a lookahead function during the solution construction by the ants. Additionally, they present a parallel implementation of their algorithm based on an island model often also used in parallel genetic algorithms.

$\mathcal{MM}$AS has recently been applied to the generalized assignment problem by Ramalhinho Lorençou and Serra [43], obtaining very promising computational results. In particular, their algorithm is shown to find optimal and near optimal solutions faster than a GRASP algorithm which was used for comparison.

Applications of ACO algorithms to telecommunication networks, in particular to network routing problems, have recently received a strongly increased interest in the ACO community (see, for example, Schoonderwoerd et al. [46] and Di Caro and Dorigo [13]). The application of ACO algorithms to network optimization problems is appealing, since these problems have characteristics like distributed information, non-stationary stochastic dynamics, and asynchronous evolution of the network status which well match some of the properties of ACO algorithms like the use of local information to generate solutions, indirect communication via the pheromone trails and stochastic state transitions. A detailed overview of routing applications of ACO algorithms can be found in Dorigo, Di Caro and Gambardella [16].

Acknowledgments

This work was supported by a Madame Curie Fellowship awarded to Thomas Stützle (CEC-TMR Contract No. ERB4001GT973400). Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Research Associate.

REFERENCES

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

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)*, pages 611–615. IEEE Press, 1996.

3. K.D. Boese. *Models for Iterative Global Optimization.* PhD thesis, University of California, Computer Science Department, Los Angeles, 1996.

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

5. E. Bonabeau, M. Dorigo, and G. Theraulaz. *From Natural to Artificial Swarm Intelligence.* Oxford University Press, 1999.

6. B. Bullnheimer, R.F. Hartl, and C. Strauss. An Improved Ant System Algorithm for the Vehicle Routing Problem. *Annals of Operations Research.* To appear.

7. 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.* To appear.

8. B. Bullnheimer, R.F. Hartl, and C. Strauss. Applying the Ant System to the Vehicle Routing Problem. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 285–296. Kluwer, Boston, 1999.

9. A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In *Proceedings of the First European Conference on Artificial Life (ECAL 91)*, pages 134–142. Elsevier, 1991.

10. A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant System for Job-Shop Scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.

11. D. Costa and A. Hertz. Ants Can Colour Graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

12. G.A. Croes. A Method for Solving Traveling Salesman Problems. *Operations Research*, 6:791–812, 1958.

13. G. Di Caro and M. Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, December 1998. Available at http://www.jair.org.

14. M. Dorigo. *Optimization, Learning, and Natural Algorithms (in Italian)*. PhD thesis, Dip. Elettronica, Politecnico di Milano, 1992.

15. 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, 1999.

16. M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant Algorithms for Distributed Discrete Optimization. *Artificial Life*. To appear.

17. M. Dorigo and L.M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

18. M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy. Technical Report 91-016, Dip. Elettronica, Politecnico di Milano, 1991.

19. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–42, 1996.

20. 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)*, pages 616–621. IEEE Press, 1996.

21. 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*, pages 252–260. Morgan Kaufmann, 1995.

22. 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)*, pages 622–627. IEEE Press, 1996.

23. 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.

24. L.M. Gambardella, É. 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, 1999.

25. L.M. Gambardella, É.D. Taillard, and M. Dorigo. Ant Colonies for the QAP. *Journal of the Operational Research Society.* To appear.

26. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness.* Freeman, San Francisco, CA, 1979.

27. M. Gorges-Schleuter. Asparagos96 and the Travelling Salesman Problem. In T. Baeck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 171–174, 1997.

28. P. P. Grassé. La Reconstruction du Nid et les Coordinations Interindividuelles chez *bellicositermes natalensis et cubitermes sp.* La Théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

29. 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*, pages 215–310. John Wiley & Sons, 1997.

30. 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*, pages 184–192. Morgan Kaufman, 1995.

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

32. S. Lin. Computer Solutions for the Traveling Salesman Problem. *Bell Systems Technology Journal*, 44:2245–2269, 1965.

33. S. Lin and B.W. Kernighan. An Effective Heuristic Algorithm for the Travelling Salesman Problem. *Operations Research*, 21:498–516, 1973.

34. V. Maniezzo. Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. Technical Report CSR 98-1, Scienze dell'Informazione, Universitá di Bologna, Sede di Cesena, 1998.

35. V. Maniezzo and A. Colorni. The Ant System Applied to the Quadratic Assignment Problem. *IEEE Transactions on Knowledge and Data Engineering.* To appear.

36. 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.

37. O. Martin and S.W. Otto. Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research*, 63:57–75, 1996.

38. O. Martin, S.W. Otto, and E.W. Felten. Large-Step Markov Chains for the Traveling Salesman Problem. *Complex Systems*, 5(3):299–326, 1991.

39. 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)*, pages 159–164. IEEE Press, 1997.

40. R. Michel and M. Middendorf. An Island Based Ant System with Lookahead for the Shortest Common Supersequence Problem. In *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498, pages 692–708. Springer Verlag, 1998.

41. A. Möbius, B. Freisleben, P. Merz, and M. Schreiber. Combinatorial Optimization by Iterative Partial Transcription. Submitted to *Physical Review E*, 1998.

42. Y. Nagata and S. Kobayashi. Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem. In *Proc. of ICGA'97*, pages 450–457, 1997.

43. 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, May 1998.

44. G. Reinelt. TSPLIB — A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3:376–384, 1991.

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

46. R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1996.

47. T. Stützle. An Ant Approach to the Flow Shop Problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564. Verlag Mainz, Aachen, 1997.

48. T. Stützle. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the Quadratic Assignment Problem. Technical Report AIDA–97–4, FG Intellektik, TU Darmstadt, July 1997.

49. T. Stützle. *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications.* PhD thesis, Darmstadt University of Technology, Department of Computer Science, 1998.

50. T. Stützle. ACO Algorithms for the Quadratic Assignment Problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization.* McGraw-Hill, 1999.

51. T. Stützle and H.H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem. In T. Baeck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 309–314, 1997.

52. T. Stützle and H.H. Hoos. Improvements on the Ant System: Introducing the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. In R.F. Albrecht G.D. Smith, N.C. Steele, editor, *Artificial Neural Networks and Genetic Algorithms*, pages 245–249. Springer Verlag, Wien New York, 1998.

53. 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*, pages 313–329. Kluwer, Boston, 1999.

54. T. Walters. Repair and Brood Selection in the Traveling Salesman Problem. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proc. of Parallel Problem Solving from Nature – PPSN V*, volume 1498 of *LNCS*, pages 813–822. Springer Verlag, 1998.

55. C.J.C.H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 8:279–292, 1992.