Mathware & Soft Computing 7 (2000)

Experiments with Variants of Ant Algorithms

Thomas Stützle and Sebastian Linke
Darmstadt University of Technology
Department of Computer Science
Intellectics Group
Alexanderstr. 10, 64283 Darmstadt (Germany)
e-mail: stuetzle@informatik.tu-darmstadt.de, mail@sebastian-linke.com

Abstract

A number of extensions of Ant System, the first ant colony optimization (ACO) algorithm, were proposed in the literature. These extensions typically achieve much improved computational results when compared to the original Ant System. However, many design choices of Ant System are left untouched including the fact that solutions are constructed, that real-numbers are used to simulate pheromone trails, and that explicit pheromone evaporation is used. In this article we experimentally investigate adaptations of ant algorithms to the traveling salesman problem that use alternative choices for these latter features: we consider using pheromones to modify solutions and different schemes for manipulating pheromone trails based on integer pheromone trails without recurring to pheromone evaporation.

1 Introduction

Ant algorithms are a class of algorithmic approaches, loosely inspired by models of real ant behavior, for the solution of optimization and distributed control problems. Currently, the most successful of these approaches is ant colony optimization (ACO), a particular class of ant algorithms that follows the rules of the ACO metaheuristic. The ACO metaheuristic was proposed to provide a unifying framework for many applications of ant algorithms [5, 4] to combinatorial optimization problems. ACO algorithms have been successfully applied to several $\mathcal{NP}$-hard combinatorial optimization problems [1, 4, 5, 6, 10, 13, 15, 16]; we refer to [9] for a recent overview of ACO and its applications.

From a high-level perspective, ACO is a population-based algorithm, where artificial pheromone trails are used to coordinate a population of simple agents, called (artificial) ants. Characteristic features of ACO are that (i) the artificial ants *construct* solutions to the problem under concern, (ii) the communication among the ants is *indirect* via the artificial pheromone trails, (iii) *pheromones evaporate* over time, and (iv) additional procedures may be applied by the ants to perform centralized actions that can not be carried out by single ants from their local perspective.

The first ACO algorithm, called Ant System (AS) [3, 7, 8], was initially applied to the Traveling Salesman Problem (TSP). It gave encouraging results, yet its performance was not competitive with state-of-the-art algorithms. Therefore, one important focus of research on ACO has been the introduction of algorithmic improvements over AS to achieve much better performance. These improved algorithms include elitist Ant System [3], Ant Colony System (ACS) [6], $\mathcal{MAX}$–$\mathcal{MIN}$-Ant-System ($\mathcal{MM}$AS) [19, 20], the rank-based version of Ant System [1], and Best-Worst Ant System [2]. Typically, these algorithms have been tested again on the TSP [17] and have shown significantly better performance that AS in several applications.

Only few other successful ant algorithms were proposed for tackling $\mathcal{NP}$-hard optimization problems that do not follow the rules of the ACO metaheuristic. These exceptions include Hybrid Ant System (HAS) proposed by Gambardella, Taillard, and Dorigo [11] and Fast Ant System (FANT) by Taillard and Gambardella [22], which were both applied to the Quadratic Assignment Problem. Both, HAS and FANT are not ACO algorithms because they differ in at least one rule from those of the ACO metaheuristic.

HAS uses pheromone trails to guide a solution modification process but not to construct solutions. So far, HAS was applied only to the quadratic assignment problem [11] and it is not obvious, whether the use of pheromone trails in guiding direct solution modifications has any advantage over the use of pheromone trails in a solution construction process as in ACO algorithms. FANT differs mainly in the pheromone management process and the avoidance of an explicit pheromone evaporation from ACO: FANT uses a pheromone management mechanism that is based on integer numbers to represent pheromone trails and occasionally applies a re-initialization of the pheromone trails instead of pheromone evaporation to diversify the search. One side effect of FANT is that computationally simpler operations like additions instead of multiplications are used in the algorithm.

In this article, we experimentally investigate the influence on the performance of the rules in which HAS and FANT differ from typical ACO algorithms using the TSP as an example application. To investigate the influence of using pheromone trails for solution modifications, we adapt an available implementation of $\mathcal{MM}$AS, one of the currently best performing ACO algorithms [19, 18, 20]. In a second set of experiments we implemented FANT for the TSP. Finally, we combine the modifications introduced by HAS and FANT into a new ant algorithm that appears to be competitive with the currently best performing ACO algorithms when applied to the TSP.

The article is structured as follows. Section 2 introduces ACO, the TSP and one of the currently best performing ACO algorithm for TSPs, $\mathcal{MM}$AS. In Section 3 we introduce HAS and FANT and we present computational results in Section 4. We end with some concluding remarks in Section 5.

```
procedure Ant Colony Optimization
    Init pheromones, calculate heuristic
    while(termination condition not met) do
        p = ConstructSolutions(pheromones, heuristic)
        p = LocalSearch(p)    % optional
        GlobalUpdateTrails(p)
    end
end Ant Colony Optimization
```

Figure 1: Algorithmic skeleton of ant colony optimization algorithms for static combinatorial optimization problems.

2 Ant colony optimization

2.1 ACO algorithms

Ant Colony Optimization (ACO) [5, 4] is a population-based approach that was inspired by the behavior of real ant colonies, in particular, by their foraging behavior. One of the main ideas underlying this approach is the indirect communication among the individuals of a colony of agents, called *(artificial) ants*, based on an analogy with pheromone trails that real ants use for communication (pheromones are an odorous, chemical substance). The (artificial) pheromone trails are a kind of distributed numeric information that is modified by the ants to reflect their accumulated experience while solving a particular problem.

In fact, ants in ACO implement stochastic construction procedures that are biased by pheromone trails and heuristic information (for example, distances between cities in the TSP case) on the problem being solved. The solutions obtained by the ants may then be improved by applying some local search procedure. When applied to the TSP and other $\mathcal{NP}$-hard *static* combinatorial problems (*Static combinatorial problems* are those in which all relevant problem data are available before the start of the algorithm. This is in contrast to dynamic problems, where problem data may change during the solution process.), ACO algorithms follow the high-level procedure given in Figure 1. After the initialization of the pheromone trails and some parameters, a main loop is repeated until some termination condition, like a maximum limit on the number of solution constructions or the maximum CPU-time is met: First, the ants construct solutions; in a following, optional phase the generated solutions may be improved by a local search, and finally the pheromone trails are updated. It should be noted that the ACO metaheuristic [5, 4] is more general than the algorithmic scheme given here. For example, the algorithmic scheme of Figure 1 does not capture the application of ACO algorithms to network routing problems, which is an inherently dynamic problem.

2.2 The traveling salesman problem

The traveling salesman problem (TSP) plays an important role in Ant Colony Optimization for many reasons: (*i*) it is a problem to which ACO algorithms are easily applied, (*ii*) it is an $\mathcal{NP}$-hard 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.

The TSP can be represented by a complete, weighted directed graph $G = (\mathcal{V}, \mathcal{A}, d)$ with $\mathcal{V}$ being the set of nodes (the cities), $\mathcal{A}$ being the set of arcs, and $d : \mathcal{A} \mapsto \mathbb{N}$ being a weight function associating a positive, integer weight $d(i,j)$ to every arc (i,j), which corresponds to the distance between cities i and j. The goal is to find a shortest closed path that visits every city exactly once, that is, a Hamiltonian cycle (often called *tour* in the TSP context). For symmetric TSPs, the distance between every pair of cities is independent of the direction of traversing the cities, that is, $d(i,j) = d(j,i)$ for every pair of nodes. All TSP instances used in the empirical studies presented in this article are symmetric and are taken from the TSPLIB benchmark library which is accessible at `http://www.iwr.uni-heidelberg.de/iwr/ comopt/soft/TSPLIB95`. These instances have been used in many experimental studies on the TSP and, in part, stem from practical applications of the TSP.

2.3 $\mathcal{MAX}$–$\mathcal{MIN}$-Ant-System

$\mathcal{MAX}$–$\mathcal{MIN}$-Ant-System ($\mathcal{MM}$AS) [21, 19, 20] is one of the currently best performing ACO algorithms. $\mathcal{MM}$AS is a direct successor of Ant System (AS), the first ACO algorithm; it differs from AS mainly in the pheromone update, the use of local search algorithms to improve the solutions constructed by the ants, and the use of explicit limits on the range of feasible values of the pheromones.

$\mathcal{MM}$AS constructs solutions in exactly the same way as AS: Initially, each ant is put on some randomly chosen city. At each construction step, ant k applies a probabilistic action choice rule: when being at city i, ant k chooses to go to a yet unvisited city j at the tth construction step with a probability of

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{if } j \in \mathcal{N}_i^k \tag{1}$$

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; if $j \notin \mathcal{N}_i^k$, we have $p_{ij}^k(t) = 0$.

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

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{2}$$

where $\Delta\tau_{ij}^{best} = 1/L^{best}$ if arc $(i,j) \in T^{best}$ and zero otherwise. T^{best} is either the *iteration-best* solution T^{ib}, or the *best-so-far* solution T^{bsf} and L^{best} is the corresponding tour length. Experimental results have shown that the best performance is obtained by gradually increasing the frequency of choosing T^{bsf} for the trail update [16, 20].

In $\mathcal{MM}$AS lower and upper limits on the possible pheromone strengths τ_{min} and τ_{max} on any arc are imposed to avoid search stagnation. Hence, for all arcs (i, j) we have $\tau_{ij} \in [\tau_{min}, \tau_{max}]$. Experimental results [16, 20] 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. The pheromone trails in $\mathcal{MM}$AS are initialized to their upper pheromone trail limits. Doing so together with low value for the pheromone evaporation ρ leads to an increased exploration of tours at the start of the algorithms because the relative differences between the pheromone trail strengths are less pronounced. We refer to [16, 20] for a detailed experimental analysis of $\mathcal{MAX}$–$\mathcal{MIN}$-Ant-System.

3 Hybrid Ant System and Fast Ant System

In this section we describe the two ant algorithms that inspired us to reconsider specific design choices of Ant System, Hybrid Ant System (HAS) [11] and Fast Ant System (FANT) [22]. The features of these two algorithms we consider are (i) the exploitation of pheromone trails for *solution modification* instead of solution construction (taken from HAS) and (ii) a different pheromone management scheme (taken from FANT). While both algorithms were initially applied to the quadratic assignment problem, here their description is geared towards the TSP.

3.1 Solution modification

In HAS, the pheromone trail based modification of an ant's current solution is applied by each ant k. It consists in repeating R times a random exchange move, for example, in the TSP case a two-exchange move. First, an index r is randomly chosen from $1, \ldots, n$. Then, a second index $s \neq r, |s - r| > 1$ is chosen and a 2-exchange move involving the cities at the indices $r, r+1, s$, and $s+1$ is performed. The second index s is chosen in the following way. With probability $1-q$, where q is a parameter, it is chosen to maximize the value of $\tau_{sr} + \tau_{(r+1)(s+1)}$ (exploitation), with a probability of q the second index s is randomly chosen with a probability of

$$\frac{\tau_{sr} + \tau_{(r+1)(s+1)}}{\sum_{i \neq r}(\tau_{si} + \tau_{(r+1)(i+1)})} \tag{3}$$

After a solution has been modified, local search is applied to the so generated solution. In HAS additional features of search intensification and diversification are added, we refer to [11] for a description of the techniques used.

HAS is similar in spirit to iterated local search algorithms [14] in which a local search algorithm is repeatedly applied to perturbations of the current solution. When applied to the TSP, ILS algorithms use a particular 4-exchange move that was shown to be particularly successful. In this solution perturbation a current tour is cut at four randomly chosen edges into four sub-tours $s_1 - s_2 - s_3 - s_4$

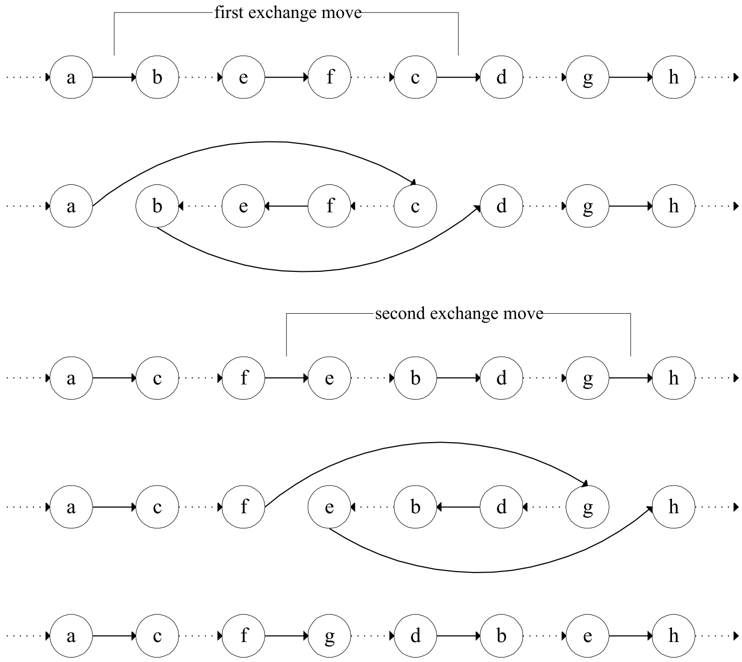

Figure 2: Solution modification for TSPs as used by HAS.

contained in the solution in the given order. These sub-tours are then reconnected in the order $s_4 - s_3 - s_2 - s_1$ to yield a new starting solution for the local search. Note that (i) this corresponds to a 4-exchange move that cannot be undone by two independent 2-exchange moves and that (ii) in ILS algorithms typically random perturbations are applied. Because of the similarity of the solution modification in HAS to this 4-exchange move, in the following we will always apply two of the above described 2-exchange moves simultaneously, which allows for easier comparisons of the effect of the solution modification to this perturbation. The use of pheromones in the perturbation process introduces a bias into the perturbation process and allows to guide the perturbation towards better solutions.

3.2 FANT-type pheromone manipulation

In FANT, solutions are constructed using the action choice rule given in Equation 1, yet without using heuristic information. FANT differs from the other approaches presented so far in two main aspects. The first aspect is that the algorithm makes use of only one single ant, i.e., no population is used. The use of only one single ant allows the algorithm to find good solutions fast. However, using only one single ant appears to be merely a specific parameter setting and not an essential feature of the algorithm: Experiments with FANT showed that on the long run significantly better solution quality is obtained when using a colony size larger than one. Therefore, in the experiments indicated below, we use a population of ants as it is typically done in ACO algorithms. The second aspect, which is the main focus of this research, concerns the management of the pheromone trail matrices. One first difference of the pheromone management in FANT to ACO algorithms is that FANT uses only integer values for the pheromone trails. A second difference is that FANT does not use pheromone evaporation after each iteration, but that it rather uses occasional re-initialization of the pheromone trails to avoid search stagnation.

Pheromone trails in FANT are modified as follows: Initially, all pheromone trails are set to one. After each iteration pheromone is added to the current pheromone trail matrix. For the pheromone trail update two parameters, ξ and ξ^*, are used. ξ represents the amount of pheromone added to each arc of the iteration-best solution and ξ^* is the amount of pheromone added to each arc of the best-so-far solution T^{bsf}. ξ^* is assigned a fixed value, the value of ξ is modified during the algorithm's run. In two occasions the pheromone matrix is updated different from the above described rule: (i) if the best-so-far solution has been improved, ξ and all the pheromone trails are reinitialized to one, resulting in an intensification of the search around the best-so-far solution; (ii) if the constructed solution corresponds to T^{bsf}, ξ is increased by one and all pheromone trails are reinitialized to ξ, resulting in a diversification of the search.

4 Experimental results

We have tested the algorithms on a set of symmetric TSP instances from TSPLIB. The experiments were run on a 500 MHz Celeron processor with 256 MB of RAM, the maximally allowed computation time was 600 seconds for instances with less than 500 cities (the results are based on 25 independent trials in this case), and 1200 seconds for the others (10 independent trials). In the following all implementations use either for the solution construction or the solution modification nearest neighbor lists of length 20. All solutions are improved by a first-improvement 3-opt local search algorithm that uses nearest neighbor lists of length 40, a fixed radius nearest neighbor search, and don't look bits. We refer to [12] for details on these speed-up techniques.

4.1 $\mathcal{MAX}$–$\mathcal{MIN}$-Ant-System

For a baseline comparison for the ant algorithms under investigation to state-of-the-art ACO algorithms, we give computational results obtained with $\mathcal{MM}$AS in Table 1. The results are based on a re-implementation of the $\mathcal{MM}$AS algorithm described in [20] using the same parameter settings: We use $m = 25$ ants, $\alpha = 1$, $\beta = 2$, $\rho = 0.2$, $\tau_{max} = 1/(\rho \cdot L^{bsf})$, $\tau_{min} = \tau_{max}/2n$. The frequency f^{bsf} of using T^{bsf} in the pheromone trail update (f^{bsf} indicates that every f^{bsf} iterations T^{bsf} is allowed to deposit pheromone) is set as follows (if T^{bsf} is not used, pheromone is added to the arcs of T^{ib}): In the first 25 iterations only T^{ib} is used to update the

Table 1: Computational results with $\mathcal{MM}$AS. Given are for each instance the known optimal solution, the best solution found by $\mathcal{MM}$AS, the average solution quality found (avg.) and the average time to find the best solution in each trial.

instance	opt	best	avg.	avg.time
lin318	42029	42029	42029	16
pcb442	50778	50778	50781	142
rat783	8806	8806	8806	433
pcb1173	56892	56892	56897	416
d1291	50801	50801	50809	571

trails; we set f^{bsf} 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 $t > 250$. While $\mathcal{MM}$AS solves the three smallest instances in short computation time in almost every test run (instances `lin318` and `rat783` are solved to optimality in every single test run), on the two largest instances not every test run could identify the known optimal solutions.

4.2 Solution modification

An algorithmic outline of the HAS-algorithm is given in Figure 3. After initializing the ants with random initial tours, the algorithms iterates through a main loop, where, starting from *ant*, first this tour is modified to yield ant'. Next, ant' is improved by a local search algorithm, resulting in a locally optimal tour ant''. Finally, the tour that is kept in the population is the solution ant'', if ant'' is shorter than *ant*, otherwise *ant* is kept in the population. Each single modification consists actually of two independently chosen 2-exchange moves; the effect is that each single modification cuts and re-introduces exactly 4 arcs. Since one single modification may not be enough, we varied the number of modifications for each ant systematically between 1 and 10 in steps of 1. Regarding the pheromone trail manipulation, the same procedure as followed by $\mathcal{MM}$AS is used including the pheromone trail limits and the corresponding parameter settings, and the value of q is set to one (this gave best results according to some preliminary experiments). Because of the pheromone update procedure being taken from $\mathcal{MM}$AS, we call the resulting HAS algorithm in the following MMAS-HAS.

Computational results with this first variant are given on the left side of Table 2. Compared to $\mathcal{MM}$AS, the performance of this variant is poor. But why is this the case? In fact, ILS algorithms based on only one single random solution perturbation by a particular 4-exchange move are among the best performing algorithms for the TSP. We conjectured that the poor performance of solution modifications in ant algorithms may be due to the type of modification introduced. We differentiate between three types of modifications.

Type 1: one 2-exchange move is fully within one other 2-exchange move.

procedure *HAS for TSP*
  Init pheromones
  foreach *ant* **do**
    ant = RandomTour
  while (termination condition not met) **do**
    foreach *ant* **do**
      ant' = Modification(ant, pheromones)
      ant'' = LocalSearch(ant')
      ant = Acceptance(ant, ant'')
    end
    GlobalUpdatePheromoneTrails
    if (converged)
      Reinitialize pheromones
    end
  end
end *HAS for TSP*

Figure 3: Algorithmic skeleton for HAS.

Table 2: *Left side*: Computational results for MMAS-HAS; any modifications are allowed. (See Table 1 for a description of the entries.) *Right side*: Distribution of the frequencies of the three modification types in the basic version of MMAS-HAS. (see text for details).

instance	opt	best	avg.	avg.time
lin318	42029	42029	42100	53
pcb442	50778	50778	50899	137
rat783	8806	8840	8858	793
pcb1173	56892	57363	57510	991
d1291	50801	50885	51019	706

type 1	type 2	type 3
6.8%	0.4%	92.8%
12.6%	1.0%	86.4%
8.8%	0.6%	90.6%
6.4%	0.4%	93.2%
3.7%	0.1%	96.2%

Type 2: the two 2-exchange moves overlap.

Type 3: the two 2-exchange moves are completely independent of each other

Figure 4 gives a graphical illustration of these three types of modifications. An unfortunate case is certainly a type 3 modification, because it can directly be undone in the local search. This is also, depending on the neighborhood examination scheme, true for the type 1 modification; the only case where undoing modification seems more difficult is the type 2 modification.

The right side of Table 2 gives the frequencies of the single modifications for a run of our algorithm. In fact, the rather useless type 3 modification occurs most frequently, for all instances in more than 86% of the cases! Based on this observation we implemented further variants, where the type of solution modification was restricted. The left side of Table 3 gives the computational results if the modification is restricted to type 1 or type 2; the frequencies of the occurrence of the single

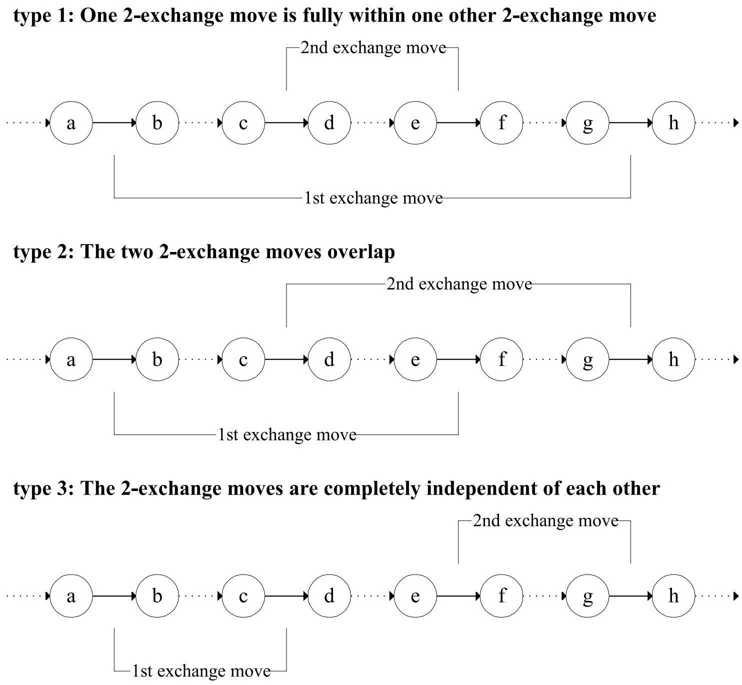

Figure 4: The three types of solution modifications occurring in the HAS application to the TSP.

modification types are given on the right side of Table 3. Although the computational results improved considerably over the first version, we tested a third one with only type 2 modifications; the results are given in Table 4 on the left side. In fact, this variant obtains the best computational results, but still the performance is significantly worse than for $\mathcal{MM}$AS.

A disadvantage of using type 2 modifications only is that finding such a modification is not a trivial task and rather time intensive. To avoid searching for applicable type 2 moves, we may relax the condition that the two exchange moves that compose one single modification move have to be found using the pheromone trails. We do so by considering a modification, where the second 2-exchange move is chosen randomly without using pheromone trails but in such a way that the two 2-exchange moves overlap (type 2a modification). Computational results with this variant are given in Table 4 on the right side. Surprisingly a strong improvement in performance could be obtained. Certainly, due to the faster modification process more solutions can be generated, but this advantage is rather small and does not explain the large improvement of the computational results. Hence, the only reason for this fact may be the increased randomness of the solution modifications.

To test this conjecture, in a final experiment we run MMAS-HAS in a version where all pheromone trails are the same, that is, the solution modification is done

Table 3: *Left side*: Computational results for MMAS-HAS; only type 1 and 2 modifications are allowed. (See Table 1 for a description of the entries.) *Right side*: Distribution of the frequencies of the two modification types in this restricted version of MMAS-HAS.

instance	opt	best	avg.	avg.time	type 1	type 2
lin318	42029	42029	42066	52	93.9%	6.1%
pcb442	50778	50778	50849	121	91.7%	8.3%
rat783	8806	8820	8836	744	93.1%	6.9%
pcb1173	56892	57175	57351	991	94.3%	5.7%
d1291	50801	50909	50955	697	97.5%	2.5%

Table 4: *Left side*: Computational results for MMAS-HAS; only type 2 modifications are allowed. *Right side*: Computational results for MMAS-HAS if only type 2a modifications are allowed (second 2-exchange move does not consider pheromone trails). (See Table 1 for a description of the entries.)

instance	opt	best	avg.	avg.time	best	avg.	avg.time
lin318	42029	42029	42032	55	42029	42029	19
pcb442	50778	50778	50792	175	50778	50792	68
rat783	8806	8807	8820	694	8806	8807	508
pcb1173	56892	57150	57289	894	56892	56910	746
d1291	50801	50843	50885	392	50801	50826	619

in a completely random way according to uniform distributions. In fact, such an algorithm corresponds to the parallel execution of m ILS runs with random modifications as applied in MMAS-HAS.

The results with this algorithm in Table 5 are discouraging regarding the guidance of the modifications by pheromones: The algorithm with random modifications showed even slightly better performance! One interpretation of this results is that the solution modification is biased too strongly by the pheromone trails and that it should be somewhat more random. In fact, a mix of pheromone trails based solution modification and random modifications may be a good compromise.

Table 5: Computational results for MMAS-HAS algorithm with random modifications instead of modifications guided by pheromone trails, which, in fact, corresponds to parallel independent ILS runs. (See Table 1 for a description of the entries.)

instance	opt	best	avg.	avg.time
lin318	42029	42029	42029	33
pcb442	50778	50778	50779	120
rat783	8806	8806	8807	681
pcb1173	56892	56893	56900	784
d1291	50801	50801	50814	529

Table 6: *Left side*: FANT algorithm with parameter settings $\xi = 1, \xi^* = 4$. *Right side*: FANT algorithm with parameter settings $\xi = 1, \xi^* = 4$ and additional restart after 200 iterations without improvement.(See Table 1 for a description of the entries.)

instance	opt	best	avg.	avg.time	best	avg.	avg.time
lin318	42029	42029	42029	15	42029	42034	23
pcb442	50778	50798	50876	88	50778	50805	42
rat783	8806	8819	8851	145	8806	8812	619
pcb1173	56892	57312	57410	425	57018	57068	1107
d1291	50801	50820	50895	337	50803	50849	859

4.3 FANT-type pheromone manipulation

When applying FANT, we run experiments with the parameter settings originally proposed for FANT with the only exception that we use a population of 25 ants. The results with that scheme are given in Table 6 on the left side. As already the case in the first implementation of MMAS-HAS, the computational results are rather deceptive, especially for the larger TSP instances. A closer examination of the algorithm showed that this effect is on the one side due to a quick convergence of the algorithm to suboptimal solutions when ξ is low compared to ξ^*. On the other side, if ξ is too large compared to ξ^* (we tested a setting of $\xi = 1$ and $\xi^* = 2$), we found that the algorithm cannot focus to promising search space regions and fails to identify very good solutions (the results are not shown here). To avoid early convergence when ξ is low compared to ξ^*, we added a restart mechanism to FANT and, in fact, improved results were identified as can be seen in Table 6 on the right side. Nevertheless, the computational results obtained with FANT were still significantly worse than those of $\mathcal{MM}$AS.

4.4 Combining HAS and FANT ideas

In one final experiment we combined the idea of modifying solutions guided by pheromone trails with the FANT-type pheromone manipulation. This may be a good idea, because apparently in MMAS-HAS the guidance given by the pheromone trails was too strong, while in FANT alone it appears to be too weak to efficiently solve the TSP. In fact, the computational results in Table 7 confirm this conjecture: the computational results with this combination are much better than either with MMAS-HAS or with FANT, and also slightly better than with the MMAS-HAS-variant using random solution modifications (see Tables 4 on the right side and 5) and in the same range as the computational result obtained with $\mathcal{MM}$AS.

5 Conclusions

In this article we have re-considered design choices for the implementation of ant algorithms. In particular, we (i) used pheromones to guide solution modifications and (ii) applied a procedure for the pheromone management inspired by the FANT algorithm.

Table 7: Computational results for HAS-variant with FANT-type pheromone update, parameters $\xi = 1, \xi^* = 2$.

instance	opt	best	avg.	avg.time
lin318	42029	42029	42029	18
pcb442	50778	50778	50778	68
rat783	8806	8806	8806	468
pcb1173	56892	56892	56897	909
d1291	50801	50801	50816	541

Initially, we expected that the idea of using solution modifications based on pheromone trails may achieve very high performance, because ILS algorithms, which often use pure random solution modifications, are currently among the best available algorithms for the TSP. However, the initial results were rather disappointing; at least we could identify some reasons for the poor performance: without restrictions on the type of modifications, very often such modifications are introduced that can be undone directly by a local search and the guidance of the solution modifications by the pheromone trails seems to lead to early stagnation behavior of the search. In fact, with increased "randomness" of the solution modification and less guidance by pheromone trails the computational results improved.

Regarding the pheromone management scheme applied in FANT, it seemed that the algorithm was, on one side, stuck too fast in sup-optimal solutions; but once the diversification level obtained by modifying the parameter settings is high, the algorithm looses the ability to focus on a specific search space region with high quality solutions.

Finally, by combining the two ideas of solution modifications and FANT-like pheromone management, we could obtain a new ant algorithm that was able to match the performance of current state-of-the-art ACO algorithms.

From the ACO perspective, the different design choices we investigated and the, in part, rather poor computational results indicate that it seems to be rather difficult to further improve significantly the performance of state-of-the-art ACO algorithms. The results also show that by modifying some details of ACO algorithms and introducing new techniques not necessarily improvements over already existing algorithms are obtained. Therefore, it has to be carefully studied, whether newly introduced techniques are really necessary and helpful. In fact, we may see the existing ACO algorithms as particular instantiations of the available components in a "design space", which contains the set of possible ACO algorithm instantiations. If we take this point of view, we can conclude that the performance of the best available ACO algorithms may be improved somewhat, but it seems that they correspond to a high quality local optimum in this design space.

Acknowledgments

This work was partially supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-

CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] 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(1):25–38, 1999.

[2] O. Cordon, I. Fernández de Viana, F. Herrera, and L. Moreno. A new ACO model integrating evolutionary computation concepts: The best-worst Ant System. In M. Dorigo, M. Middendorff, and T. Stützle, editors, *Proceedings of Ants'2000*, pages 22–29, Brussels, Belgium, 2000.

[3] M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992. pp. 140.

[4] 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*, pages 11–32. McGraw Hill, London, UK, 1999.

[5] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[6] 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.

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

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

[9] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications and advances. In *Metaheuristics Handbook*. Kluwer Academic Publishers, 2002. In press.

[10] L. M. Gambardella and M. Dorigo. Ant Colony System hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.

[11] L. M. Gambardella, È. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176, 1999.

[12] 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, Chichester, 1997.

[13] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358–369, 1999.

[14] 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.

[15] R. Michel and M. Middendorf. An ACO algorithm for the shortest supersequence problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 51–61. McGraw Hill, London, UK, 1999.

[16] T. Stützle. *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications.* Infix, Sankt Augustin, Germany, 1999.

[17] T. Stützle and M. Dorigo. ACO algorithms for the traveling salesman problem. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, pages 163–183. John Wiley & Sons, Chichester, UK, 1999.

[18] T. Stützle and H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for combinatorial optimization problems. In S. Voß, S. Martello, I. H. Osman, and C. Roucairol, editors, *Metaheuristcs — Advances and Trends in Local Search Paradigms for Optimization*, pages 313–329, Dordrecht, 1999. Kluwer Academic Publishers.

[19] 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 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 309–314. IEEE Press, Piscataway, NJ, 1997.

[20] T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

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

[22] É. D. Taillard and L. M. Gambardella. Adaptive memories for the quadratic assignment problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.