Working Paper Series

# A New Rank Based Version of the Ant System - A Computational Study

Bernd Bullnheimer
Richard F. Hartl
Christine Strauß

Working Paper No. 1
April 1997

Working Paper Series

April 1997

SFB
'Adaptive Information Systems and Modelling in Economics and Management Science'

Vienna University of Economics
and Business Administration
Augasse 2–6, 1090 Wien, Austria

in cooperation with
University of Vienna
Vienna University of Technology

http://www.wu-wien.ac.at/am

This piece of research was supported by the Austrian Science Foundation (FWF) under grant SFB#010 ('Adaptive Information Systems and Modelling in Economics and Management Science').

# A New Rank Based Version of the Ant System - A Computational Study

Bernd Bullnheimer Richard F. Hartl Christine Strauß

University of Vienna
Department of Management Science
Bruenner Str. 72
A - 1210 Vienna, Austria
email: bernd.bullnheimer@univie.ac.at

**Keywords:** ant system, metaheuristics, traveling salesman problem, combinatorial optimization.

**Abstract**
The ant system is a new meta-heuristic for hard combinatorial optimization problems. It is a population-based approach that uses exploitation of positive feedback as well as greedy search. It was first proposed for tackling the well known Traveling Salesman Problem (TSP), but has been also successfully applied to problems such as quadratic assignment, job-shop scheduling, vehicle routing and graph colouring.

In this paper we introduce a new rank based version of the ant system and present results of a computational study, where we compare the ant system with simulated annealing and a genetic algorithm on several TSP instances. It turns out that our rank based ant system can compete with the other methods in terms of average behavior, and shows even better worst case behavior.

# 1 Introduction

The traveling salesman problem (TSP) is probably the most studied problem of combinatorial optimization. For many years it has been tackled with all kinds of exact methods, TSP-tailored heuristics, as well as so-called *meta-heuristics*, which are particularly used to solve hard combinatorial optimization problems. The ant system is a new member in the class of these meta-heuristics, with more famous members being simulated annealing (cf. e.g. [16]), genetic algorithms (cf. e.g. [13], [15]), tabu search (cf. e.g. [11], [12]), neural networks (cf. e.g. [14], [17]) and evolution strategies (cf. e.g [20]). Most of these methods (except tabu search) have been derived from nature, and the same is true for the ant system, that was developed more recently by Colorni, Dorigo and Maniezzo (cf. [4], [5], [9]).

In this paper we introduce a new version of the ant system, which we call $AS_{rank}$, and show its quality on five TSP instances of different sizes. We also present computational results of a comparison between $AS_{rank}$, the original ant system, simulated annealing and a genetic algorithm. The remainder of the paper is organized as follows: in the next section we present the basic idea and the structure of the ant system algorithm. In Section 3 extensions of the algorithm are shown and $AS_{rank}$ is introduced. Then the computational study is described and results are reported in Section 4. We conclude with general remarks on this work and directions for future research.

# 2 The ant system

## 2.1 Basic concept

The idea of the ant system is based on the following observation. A colony of ants is able to succeed in a task (for instance to find the shortest path between the nest and the food source) whereas a single ant would probably fail, especially as ants are almost blind. It was found that ants leave a trail of pheromone when they move. This pheromone trail can be observed by other ants and motivates them to follow the path, i.e. a randomly moving ant will follow the pheromone trail with high probability. That is the way how the trail is reinforced and more and more ants follow that trail. The following example (cf. [10]) shows how over time, short paths are found through this self-reinforcing process.

Suppose that the ants commute between food source $A$ and nest $E$ with speed one, i.e. they need one time unit for a distance of length one (cf. Figure 1). In every time period 30 ants leave the nest and the food source, respectively.

Now let the route that has been taken by all the ants ($ABDE$) be blocked by an obstacle. Then there are only two alternative routes via $C$ (with a total length of 3) and via $F$ (with a total length of 4), respectively. Suppose that at time $t = 0$ there are 30 ants at point $B$ (and 30 at point $D$). As these ants do not observe any pheromone trail they could follow, they randomly decide (with

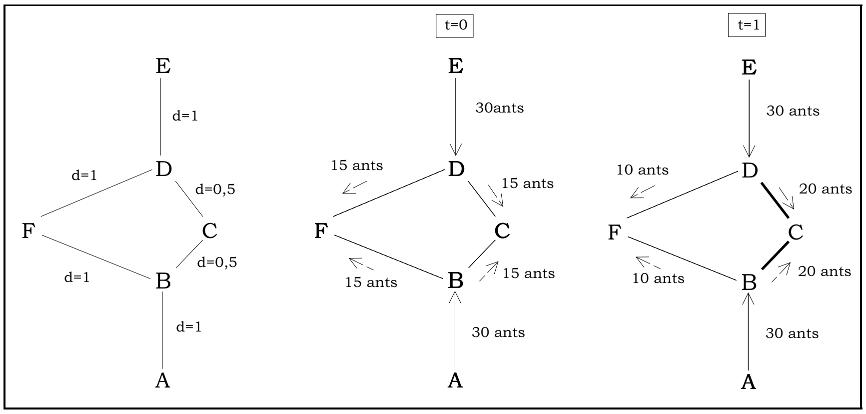


Figure 1: Illustration of example described (*from*: [10])

equal probability) on one of the two alternatives. Therefore 15 ants choose to go to $C$ and the other 15 choose to go to $F$. In $t = 1$, the next 30 ants leaving $B$ (and $D$) can observe existing trails of pheromone. The pheromone on $BCD$ is twice as intensive as the one on $BFD$, because 30 ants traveled on $BCD$ (15 from $B$ to $D$ and 15 vice versa) while only 15 ants traveled the longer way from $B$ to $F$. For that reason now 20 ants take $BCD$ and 10 take $BFD$ (and 20 ants take $DCB$ and 10 take $DFB$ from point $D$). Again there is more pheromone left on the shorter path. This process is repeated in every time period and thereby the pheromone trail is further reinforced. This example describes the transition from random to *adapted* behaviour.

The artificial ants of the ant system behave in a similar way. They differ from their natural counterparts in two aspects. They are not blind, i.e. they have information regarding their environment and they use this information to be *greedy* in addition to being adaptive. And second, they have a memory, which is necessary to ensure that only feasible solutions are generated.

In the next section the ant system algorithm and its application to the traveling salesman problem is described.

## 2.2 Ant system algorithm

Given a $n$-city TSP with distances $d_{ij}$, the artificial ants are distributed to the cities according to some rule. Each ant decides independently on the city to be visited next, until the tour is completed. At the beginning of an iteration, all cities except the one the ant is located in, can be selected. As each ant has to visit each city exactly once, the so far visited cities have to be stored, which is the reason why memory is needed.

The probability that a city is selected is the higher the more intense the trail level leading to this city is[1], and the closer that city is located[2]. The probability that city $j$ is selected to be visited immediately after city $i$ can be written in a formula as follows:

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in \Omega} [\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}} & \text{if } j \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\text{where } \eta_{ij} = \frac{1}{d_{ij}}$$

where

| | |
|---|---|
| $\tau_{ij}$ | intensity of pheromone trail between cities $i$ and $j$ |
| $\alpha$ | parameter to regulate the influence of $\tau_{ij}$ |
| $\eta_{ij}$ | visibility of city $j$ from city $i$ |
| $\beta$ | parameter to regulate the influence of $\eta_{ij}$ |
| $\Omega$ | set of cities, that have not been visited yet |
| $d_{ij}$ | distance between cities $i$ and $j$ |

This selection process is repeated until all ants have completed a tour. In each step of the iteration $t$ the set of cities to be visited is reduced by one city and finally, when only one city is left, this city is selected with probability $p_{ij} = 1$. For each ant the length of the tour generated is calculated and the best tour found so far is updated.

Then the trail levels are updated as follows: on a tour each ant leaves pheromone quantity given by $Q/L_k$, where $Q$ is a constant and $L_k$ is the length of its tour. Therefore on shorter tours there is more pheromone left per unit length. By analogy to nature, part of the pheromone evaporates, i.e. the existing pheromone trails are reduced by a factor $(1-\rho)$ before new pheromone is laid. This is done to avoid early convergence and is regulated by a parameter $\rho$. The updating of the trail level $\tau_{ij}$ can be written in a formula as follows:

[1] The *adapted* behaviour is based on the trail levels and is regulated by a parameter $\alpha$.

[2] The *greedy* behaviour is based on the visibility and is regulated by a parameter $\beta$.

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

$$\text{where } \Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k \text{ and } \Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

where

| | |
|---|---|
| $t$ | iteration counter |
| $\rho \in [0,1]$ | parameter to regulate the reduction of $\tau_{ij}$ |
| $\Delta\tau_{ij}$ | total increase of trail level on edge $(i,j)$ |
| $m$ | number of ants |
| $\Delta\tau_{ij}^k$ | increase of trail level on edge $(i,j)$ caused by ant $k$ |
| $Q$ | quantity of pheromone laid by an ant per tour |
| $L_k$ | tour length of ant $k$ |

On basis of these updated trail levels the next iteration $t+1$ can be started. As a summary, in Figure 2 the structure of the ant system algorithm is given in pseudo code.

```
Initialize
For t = 1 to number of iterations do
    For k = 1 to m do
        Repeat until ant k has completed a tour
            Select the city j to be visited next
            with probability p_ij given by equation (1)
        Calculate the length L_k of the tour generated by ant k
    Update the trail levels τ_ij on all edges according to equation (2)
End
```

Figure 2: Ant system algorithm in PseudoCode

So far nothing has been said about the number of the artificial ants and their distribution in the initialization phase. Other studies (cf. [10]) have shown that the ant system yields very good results, when the number of ants is set equal to the number of cities and each ant starts its tour from another city. Such a configuration implies that the complexity of an iteration is $\mathcal{O}(n^3)$. Therfore the complexity of the algorithm is $\mathcal{O}(T \cdot n^3)$, with $T$ indicating the number of iterations. Concerning the other parameters of the algorithm ($\alpha$, $\beta$, $\rho$ , $Q$) it was found, that if $\alpha$ is too high compared to $\beta$, the algorithm tends to enter stagnation behavior without finding good solutions; if $\alpha$ is too low, the algorithm operates like a repeated construction heuristic and generates good solutions, but can not exploit the positive feedback. The same is true for $\rho$-values close to zero; most of the global information contained in the trail levels "evaporates" immediately and learning does not take place. If $\rho$ is too close to one, there is the danger of early convergence of the algorithm. The parameter

$Q$ measures the influence of the new information (length of the tours) relative to the influence of the initial trail levels $\tau_{ij}(0)$. As long as $Q$ is not too small this parameter is not crucial for the convergence of the algorithm. The setting $\alpha = 1$, $\beta = 5$, $\rho = 0.5$ and $Q = 100$ is suggested to be advantageous in [10].

In the next section we present possible extensions of the algorithm.

# 3 Extensions of the ant system algorithm

## 3.1 Ant system with elitist strategy ($AS_{elite}$)

The quality of the solutions produced by the ant system could be improved using so-called elitist ants (cf. [10]). The idea of the elitist strategy in the context of the ant system is to give extra emphasis to the best path found so far after every iteration. When the trail levels are updated (cf. Formula (2) in Section 2.2), this path is treated as if a certain number of ants, namely the elitist ants, had chosen that path. As it is likely that some edges of that path are part of the optimal solution, the aim is to guide the search in succeeding iterations. The updating of trail levels therefore is done in the following way:

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

$$\text{where } \Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \text{ and } \Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

as before,

$$\text{and } \Delta\tau_{ij}^{*} = \begin{cases} \sigma\frac{Q}{L^{*}} & \text{if edge } (i,j) \text{ is part of the best solution found} \\ 0 & \text{otherwise} \end{cases}$$

where

| | |
|---|---|
| $\Delta\tau_{ij}^{*}$ | increase of trail level on edge $(i,j)$ caused by the elitist ants |
| $\sigma$ | number of elitist ants |
| $L^{*}$ | tour length of best solution found |

Concerning the number of elitist ants we found that the results of the ant system are very good for $\sigma = n = m$, i.e. when there are as many elitist ants used as there are cities in the problem[3].

The concept of elitism can also be found in genetic algorithms. In general, in a genetic algorithm the fittest individual of a generation (best solution of an iteration) will with positive probability not be included in the next generation, if the genetic operators *selection, recombination* and *mutation* are applied. In

[3] In [10] the best results for a 30 city problem were obtained by using eight elitist ants.

that case the genetic information of that individual would be lost. Therefore, the idea of elitism is to preserve the fittest individual of a generation. As a consequence local search aspects (exploitation) become more important while global search aspects (exploration) become less important (cf. eg. [13]). This potential drawback is also valid for the ant system.

On the other hand, in the ant system algorithm the pheromone trails are updated on basis of the tours of all ants (cf. Section 2.2). The level of contribution to this global information depends on the quality of the solution generated. That is another similarity to genetic algorithms, where the probability of being selected for reproduction depends on the fitness of an individual, i.e. better solutions are more likely to contribute to future solutions[4]. A shortcoming of this procedure is the following: if during the evolution the overall solution quality rises and differences between individuals decrease, the differences in selection probabilities decrease, too, and consequently exploitation is not as high as desired. The same is true for the ant system. The effect of emphasizing short paths diminishes when tour lengths come closer, especially when many ants travel on good but sub-optimal paths.

A possible selection scheme to solve this problem of maintaining the so-called selection pressure in genetic algorithms is ranking: first the population is sorted according to fitness and then the probability of being selected depends on the rank of an individual.

In the next section we suggest a new rank-based way of updating the trail levels to overcome the problems metioned above.

### 3.2 Ant system with elitist strategy and ranking ($AS_{rank}$)

The concept of ranking can be applied and extended to the ant system as follows: after all $m$ ants have generated a tour, the ants are sorted by tour length ($L_1 \leq L_2 \leq \ldots \leq L_m$), and the contribution of an ant to the trail level update is weighted according to the rank $\mu$ of the ant. In addition to that, only the $\omega$ best ants are considered. Thus, the danger of over-emphasized pheromone trails caused by many ants using sub-optimal paths can be avoided.

As $\sigma$ is the weight of the trail level contribution of the best tour found so far, it should not be exceeded by any other weight. On the other hand, it seemed reasonable to use "one" as the minimum weight. For that reason we decided to use the weight ($\sigma$ - $\mu$) for the $\mu$-th best ant and to set $\omega = \sigma$ - 1, which implies that the number of ants considered is exceeded by the number of elitist ants by one. In such a combined setting, with elitism and ranking, the new updating of trail levels is done according to the following Formula (4):

[4] A commonly used technique is the so-called roulette wheel selection (cf. [13]).

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau^*_{ij} \tag{4}$$

$$\text{where } \Delta\tau_{ij} = \sum_{\mu=1}^{\sigma-1} \Delta\tau^{\mu}_{ij}$$

$$\text{and } \Delta\tau^{\mu}_{ij} = \begin{cases} (\sigma-\mu)\frac{Q}{L_\mu} & \text{if the } \mu\text{-th best ant travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and } \Delta\tau^*_{ij} = \begin{cases} \sigma\frac{Q}{L^*} & \text{if edge } (i,j) \text{ is part of the best solution found} \\ 0 & \text{otherwise} \end{cases}$$

where

| | |
|---|---|
| $\mu$ | ranking index |
| $\Delta\tau^{\mu}_{ij}$ | increase of trail level on edge $(i,j)$ caused by the $\mu$-th best ant |
| $L_\mu$ | tour length of the $\mu$-th best ant |
| $\Delta\tau^*_{ij}$ | increase of trail level on edge $(i,j)$ caused by the elitist ants |
| $\sigma$ | number of elitist ants |
| $L^*$ | tour length of best solution found |

The presented updating procedure seems to be a good compromise. The results of the following computational study indicate that exploitation (through emphasizing good paths) as well as exploration (through extending the emphasis to several good ants) are considerably high and well balanced. We refrained from changing the setting of the parameters. Their influence on the extended algorithm remains the same and was discussed in Section 2.2.

# 4 Computational Study

To analyze their quality, the three versions of the ant system algorithm presented in this paper, i.e. $AS$, $AS_{elite}$ and $AS_{rank}$, were applied to five different TSP instances: a 30 city instance from literature[5] and four real-life problems from an industrial application with 57, 80, 96 and 132 cities, respectively. For reasons of comparability, we furthermore applied simulated annealing, probably the most classical meta-heuristic, and a genetic algorithm, another population-based method, to the five test problems.

In literature, a wide variety of theoretical as well as applied research on simulated annealing (cf. eg. [1], [18]) and genetic algorithms (cf. eg. [8], [13]) can be found. For that reason we only briefly present the details of the implemented algorithms in the following two sections.

[5]The Oliver30 problem from [21].

## 4.1 Simulated annealing configuration

Simulated annealing was used in two versions. A straightforward one, denoted with $SA$, with a randomly generated starting solution, and an advanced one, denoted with $SA_{nn}$, with a starting solution generated with the nearest neighbor heuristic, beginning with a randomly chosen city. The initial temperature was 70 for $SA$ and 7 for $SA_{nn}$, with a cooling factor of 0.9995 in both cases. Neighborhood solutions were obtained by applying one of the four operators shown in Figure 3, where each operator had equal probability to be selected for application.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | **5** | 8 | 7 | **2** | 3 | 6 | 4 |
| 1 | **2** | 8 | 7 | **5** | 3 | 6 | 4 |

a) exchange of two cities

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | **5** | **8** | **7** | **2** | 3 | 6 | 4 |
| 1 | 3 | 6 | **5** | **8** | **7** | **2** | 4 |

b) shift of a sequence of cities

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | **5** | **8** | **7** | **2** | 3 | 6 | 4 |
| 1 | **2** | **7** | **8** | **5** | 3 | 6 | 4 |

c) inversion of a sequence of cities

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | **5** | **8** | **7** | **2** | 3 | 6 | 4 |
| 1 | **2** | **8** | **5** | **7** | 3 | 6 | 4 |

d) random rearrangement of a sequence of cities

Figure 3: Examples for neighborhood move / mutation

## 4.2 Genetic algorithm configuration

The population size of the genetic algorithm was 10 individuals. The initial population was generated with the nearest neighbor heuristic, beginning with a randomly chosen city for each individual. A generation replacement scheme with an elitist strategy (the best member of each generation always survived) was used, and individuals were selected through rank selection. In cases of mutation ($p_{mut} = 0.85$), the same operators as in simulated annealing were used, in cases of recombination ($p_{rec} = 0.1$), partially mapped crossover and uniform order-based crossover were used with equal probability. In 5% of the cases the individual was copied into the next generation without being changed.

### 4.3 Ant system configuration

In all three ant system versions the number of artificial ants was set equal to the number of cities ($m = n$) and in each city one ant started its tour. Also, the suggested parameter setting $\alpha = 1$, $\beta = 5$, $\rho = 0.5$ and $Q = 100$ (cf. Section 2.2) was used in all versions throughout the study. For $AS_{elite}$ the number of elitist ants was set equal to the number of cities ($\sigma = n$), for $AS_{rank}$ we used $\sigma = 6$ elitist ants and only the $\omega = \sigma$ - 1 = 5 best ants contributed to the pheromone trail update.

The following section comprises a description of the design of the performed computational study and an analysis of the results that were obtained.

### 4.4 Results

The design of the computational study was as follows. For each of the six algorithms described ($SA$, $SA_{nn}$, $GA$, $AS$, $AS_{elite}$ and $AS_{rank}$) 30 runs were performed on each of the five test problems, using a 100 MHz Pentium. The number of iterations respectively generations was set in such a way, that for each problem, the runtime for all methods was equal. Table 1 summarizes the results of the study. The main rows indicate the various problems; they are listed in increasing order of problem size. On top of each main row there is a sub-header including the problem size $n$, the optimum tour length and the allowed runtime. The main columns refer to the method, the average, the best and the worst solution found in 30 runs using that method. Each result appears with its absolute value and its deviation from the optimum.

For the Oliver30 problem the results of all methods were very good, with the classical meta-heuristics (SA and GA) being slightly better (average deviation about 0.15% from the optimum) than the ant system algorithms (about 0.55%), and only the basic AS failing to find the optimal solution (dev. 0.04%).

Fairly the same was true for the 57-city-problem, only the average deviation from the optimum was higher in all cases, and the differences in deviation were smaller. Again, all methods were able to solve the problem optimally, except basic AS, which again had the highest average deviation from the optimum (1.17%).

With increasing problem size, the results achieved by the ants became better compared to the other procedures. For the problem with 80 cities, the optimal solution could still be found by simulated annealing and the improved ant strategies, while basic AS and the genetic algorithm failed at least by more than 0.6%. When the average deviations are compared, the quality of the ant system is more obvious. The classical methods generated tours that exceeded the optimum on average by more than 2.5% whereas for $AS_{elite}$ and $AS_{rank}$ the corresponding value was less than 1%.

The larger problems ($n = 96$ and $n = 132$) could not be solved optimally by any of the methods within the allowed runtime, but the best solutions found were (with deviations of about 1%) still very close to the optimum. The genetic algorithm performed somewhat better than simulated annealing with random

| method | avg. sol. | dev. | best sol. | dev. | worst sol. | dev. |
|---|---|---|---|---|---|---|
| $n = 30,\ L_{opt} = 423.74,\ runtime = 30s$ | | | | | | |
| SA | 424.52 | 0.18% | 423.74 | 0.00% | 447.01 | 5.49% |
| $\text{SA}_{nn}$ | 424.26 | 0.12% | 423.74 | 0.00% | 438.38 | 3.46% |
| GA | 424.42 | 0.16% | 423.74 | 0.00% | 425.27 | 0.36% |
| AS | 426.24 | 0.59% | 423.91 | 0.04% | 431.29 | 1.78% |
| $\text{AS}_{elite}$ | 426.08 | 0.55% | 423.74 | 0.00% | 438.38 | 3.46% |
| $\text{AS}_{rank}$ | 425.72 | 0.47% | 423.74 | 0.00% | 431.29 | 1.78% |
| $n = 57,\ L_{opt} = 920.08,\ runtime = 50s$ | | | | | | |
| SA | 924.62 | 0.49% | 920.08 | 0.00% | 937.31 | 1.87% |
| $\text{SA}_{nn}$ | 925.79 | 0.62% | 920.08 | 0.00% | 930.50 | 1.13% |
| GA | 927.13 | 0.77% | 920.08 | 0.00% | 931.67 | 1.26% |
| AS | 930.86 | 1.17% | 924.20 | 0.45% | 932.85 | 1.39% |
| $\text{AS}_{elite}$ | 928.00 | 0.86% | 920.08 | 0.00% | 934.68 | 1.59% |
| $\text{AS}_{rank}$ | 926.91 | 0.74% | 920.08 | 0.00% | 934.70 | 1.59% |
| $n = 80,\ L_{opt} = 370.97,\ runtime = 60s$ | | | | | | |
| SA | 382.41 | 3.08% | 370.97 | 0.00% | 407.43 | 9.83% |
| $\text{SA}_{nn}$ | 380.49 | 2.57% | 370.97 | 0.00% | 398.01 | 7.29% |
| GA | 380.23 | 2.50% | 373.51 | 0.69% | 393.38 | 6.04% |
| AS | 380.19 | 2.49% | 373.36 | 0.64% | 383.03 | 3.25% |
| $\text{AS}_{elite}$ | 374.44 | 0.94% | 370.97 | 0.00% | 381.85 | 2.93% |
| $\text{AS}_{rank}$ | 373.74 | 0.75% | 370.97 | 0.00% | 376.84 | 1.58% |
| $n = 96,\ L_{opt} = 1041.67,\ runtime = 80s$ | | | | | | |
| SA | 1101.78 | 5.77% | 1049.98 | 0.80% | 1157.24 | 11.09% |
| $\text{SA}_{nn}$ | 1079.89 | 3.67% | 1043.70 | 0.19% | 1110.47 | 6.61% |
| GA | 1074.93 | 3.19% | 1056.67 | 1.44% | 1105.81 | 6.16% |
| AS | 1068.85 | 2.61% | 1053.26 | 1.11% | 1080.66 | 3.74% |
| $\text{AS}_{elite}$ | 1055.14 | 1.29% | 1045.98 | 0.41% | 1067.93 | 2.52% |
| $\text{AS}_{rank}$ | 1055.00 | 1.28% | 1043.70 | 0.19% | 1065.20 | 2.26% |
| $n = 132,\ L_{opt} = 1528.78,\ runtime = 120s$ | | | | | | |
| SA | 1596.09 | 4.40% | 1558.53 | 1.95% | 1663.28 | 8.80% |
| $\text{SA}_{nn}$ | 1577.69 | 3.20% | 1537.23 | 0.55% | 1620.84 | 6.02% |
| GA | 1588.99 | 3.94% | 1543.14 | 0.94% | 1642.49 | 7.44% |
| AS | 1568.02 | 2.57% | 1544.30 | 1.02% | 1587.63 | 3.85% |
| $\text{AS}_{elite}$ | 1558.15 | 1.92% | 1537.73 | 0.59% | 1587.27 | 3.83% |
| $\text{AS}_{rank}$ | 1556.65 | 1.82% | 1533.54 | 0.31% | 1587.67 | 3.85% |

Table 1: Computational results

initial solution, but was itself slightly outperformed by $SA_{nn}$. The results obtained with the basic ant system were of the same quality, with lower average deviations but longer best tours found. The ant system with elitist strategy could not find a better solution than $SA_{nn}$, only through the new ranking procedure a shorter tour for the 132-city-problem could be generated.

In general it can be stated, that the ant system could for all problem instances compete with the classical meta-heuristics regarding speed and quality, and that the ranking improved the performance of the ant system algorithm in every respect. On small problems, the methods working on neighborhoods (simulated annealing and genetic algorithms) are able to search the solution space thoroughly. But for larger problems there seems to be a tendency to search only parts of it and to get caught in local optima, as the deviations of the worst solution found from the optimum were in a range between 6% and 11%. Here lies another advantage of the ant system: in every iteration new solutions are generated on the basis of *learned* data, i.e. the ant system can be seen as a repeated (stochastic) construction heuristic. At the same time, the risk of beginning with a poor initial solution, that is immanent to most meta-heuristics is much less. Especially for large problems with huge search spaces, the possible consequences, i.e. the wasted search effort in less promising areas of the search space, can worsen the results extremely. For that reason the "worst case behavior" of the ant system is (with deviations of less than 4%) very good.

# 5 Conclusion

Many combinatorial optimization problems still require heuristic search approaches, even though today's computers are very powerful. To solve these problems local search procedures such as simulated annealing or tabu search are widely used. The local search itself is based on the concept of neighborhood, and neighborhood solutions. Defining these neighborhoods is a very difficult task for certain problems. Here lies one major advantage of the ant system: instead of altering existing solutions, in each iteration new solutions are generated. Therefore the ant system does not depend on neighborhoods. Compared to genetic algorithms, another population based method, its advantage is that the finding of appropriate crossover operators - still an unsolved problem - is unnecessary. At the same time, positive feedback within the population, one major advantage of population based methods, can still be exploited in the ant system approach.

The computational study in which we compared the algorithm with simulated annealing and a genetic algorithm lead to the following conclusions. In general the ant system can compete with the other two meta-heuristics. For large problems it seems to outperform the other methods regarding average and especially worst case behaviour. Within the ant system algorithms, our new rank based version outperformed the others in any respect.

The study confirms the positive experience that was made by applying the ant system to other problems of combinatorial optimization such as the job shop scheduling problem [6], the quadratic assignment problem [19], the vehicle routing problem [2] and the graph colouring problem [7].

Directions for future research should be the analysis of parameter settings in a problem specific as well as a general framework. Furthermore the ant system algorithm seems to be well suited for parallelization [3] and for the application to other problems in the field of combinatorial optimization.

**Acknowledgements**

The authors would like to thank Marco Dorigo for many useful comments on a previous version of this paper, and Gerhard Waescher and Joerg Heuer for providing four of the five TSP instances that were taken from an industrial application, as well as their optimal solutions.

# References

[1] Aarts, E. and Korst J.: Simulated Annealing and Boltzman Machines. Wiley, Chichester 1989.

[2] Bullnheimer, B., Hartl, R.F. and Strauss, C.: Applying the Ant System to the Vehicle Routing Problem. 2nd International Conference on Metaheuristics (MIC'97), Sophia-Antipolis, France, 1997.

[3] Bullnheimer, B., Kotsis, G. and Strauss, C.: Parallelization Strategies for the Ant System. Conference on High Performance Software for Nonlinear Optimization: Status and Perspectives (HPSNO), Ischia, Italy, 1997.

[4] Colorni, A., Dorigo, M. and Maniezzo, V.: Distributed Optimization by Ant Colonies. In Proceedings of the European Conference on Artificial Life (ECAL'91, Paris, France), eds F. Varela and P. Bourgine. Elsevier Publishing, Amsterdam, 1991, pp. 134-142.

[5] Colorni, A., Dorigo, M. and Maniezzo, V.: An investigation of some properties of an ant algorithm. In Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN II, Brussels, Belgium), eds R. Maenner and B. Manderick. North-Holland, Amsterdam, 1992, pp. 509-520.

[6] Colorni, A., Dorigo, M., Maniezzo, V. and Trubian, M.: Ant system for Job-Shop Scheduling. JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science 34 (1), 1994, 39-53.

[7] Costa, D. and Hertz, A.: Ants can colour graphs. Journal of the Operational Research Society 48, 1997, 295-305.

[8] Davis, L. (ed): Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York 1991.

[9] Dorigo, M.: Optimization, Learning and Natural Algorithms. Ph.D. Thesis, Politecnico di Milano, Italy 1992 (in Italian).

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

[11] Glover, F.: Tabu Search - Part I. ORSA Journal on Computing 1 (3), 1989, 190-206.

[12] Glover, F.: Tabu Search - Part II. ORSA Journal on Computing 2 (1), 1990, 4-32.

[13] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading 1989.

[14] Hopfield, J.J. and Tank, D.W.: Neural Computation of Decisions in Optimization Problems. Biological Cybernetics 52, 1985, 141-152.

[15] Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor 1975.

[16] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: Optimization by Simulated Annealing. Science 220, 1983, 671-680.

[17] Kohonen, T.: Self-Organizing Maps. Springer, Berlin 1995.

[18] Laarhoven, P.J.M. van, and Aarts, E.: Simulated Annealing: Theory and Applications. Kluwer, Dordrecht 1987.

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

[20] Rechenberg, I.: Evolutionstrategie'94. Frommann-Holzboog, Stuttgart 1994.

[21] Whitley, D., Starkweather, T. and Fuquay, D.: Scheduling Problems and Travelling Salesman: the Genetic Edge Recombination Operator. In Proceedings of the Third International Conference on Genetic Algorithms, ed J. Schaffer. Morgan Kaufmann, Los Altos, CA, 1989, pp. 133-140.