Working Paper Series

Ant Colony Optimization applied to the Pickup and Delivery Problem

K. Doerner
R.F. Hartl
M. Reimann

Working Paper No. 76
November 2000

Working Paper Series

November 2000

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').

Ant Colony Optimization applied to the Pickup and Delivery Problem

K. Doerner R.F. Hartl M. Reimann
Department of Management Science, University of Vienna, Austria.

November 9, 2000

Abstract

In this paper we propose an ACO algorithm to optimize the total costs associated with the pickup and delivery of full truckloads under time window constraints in a hub network. We perform a thorough technical analysis of the ACO by comparing different pheromone decoding schemes, different visibility information and various population sizes. Furthermore we propose a post-optimization technique to improve the solutions. Our results show that appropriate data structures significantly improve the solution quality.

1 Introduction

Ant Colony Optimization (ACO), a new meta-heuristic recently introduced by Dorigo et al. (c.f. Dorigo et al. (1999a), Dorigo et al. (1999b)), finds its roots in work done on the Ant System in the early nineties, (see e. g. Colorni et al. (1991), Dorigo (1992), Dorigo et al. (1996)). It is inspired by the behaviour of real ants. Ants, when searching for food, mark the traversed paths with a pheromone quantity, which depends on the quality of the food source. Other ants observe these pheromone trails and are attracted to follow them, thus reinforcing the paths. Gradually, paths leading to rich food sources will be used more frequently, while other paths, leading to remote food sources will not be used any more.

The behavioural mechanism described above has been used to solve various hard combinatorial optimization problems (cf. e.g. Bullnheimer et al. (1999a), Bullnheimer et al.(1999b), Costa and Hertz (1997), Dorigo and Gambardella (1997), Stützle and Dorigo (1999)) by simulating with artificial ants searching the solution space instead of real ants searching their environment. In addition to that the objective values correspond to the quality of the discovered food and an adaptive memory is the equivalent of the pheromone trails. To guide their search through the set of feasible solutions, the artificial ants are furthermore equiped with a local heuristic function, the so-called visibility. A convergence proof for a generalized Ant System Algorithm is provided in Gutjahr (2000).

In our paper we use the Ant Colony Optimization approach to solve a special case of the pickup and delivery problem with time window constraints. A comprehensive overview of the class of pickup and delivery problems can be found in Savelsbergh and Sol (1995). In our case only full truckloads have to be considered and each truck can be used repeatedly in the planning period. The objective function is to minimize the total costs associated with the utilization of the fleet and the vehicle movements needed to satisfy all customers.

The paper is organized as follows. Section 2 deals with the description of the problem and the presentation of our model. In section 3 we provide the technical details of our proposed ACO algo-

rithm and its implementation. Our numerical analysis is presented in section 4. We conclude with some final remarks and an outlook on opportunities for future research.

2 Description of the problem and model formulation

2.1 Problem description

Consider a problem, where customers place orders with a logistics service provider, requiring shipments between two locations. In general, due to small shipment sizes, loads are not transported directly but via distribution centers. Thus, shipments occur between the pickup location of an order and the closest distribution center, between distribution centers and between a distribution center and the delivery location of an order. Furthermore, each customer demands specified times for the collection and the delivery of the order. The objective of the service provider is to minimize total costs associated with the satisfaction of all demands.

The distribution process described above basically consists of three stages. The first stage, the transportation of goods from the customers' locations to the distribution centers is usually done on vehicle round-trips using small trucks. The same applies to the third stage, the transportation between distribution centers and customer delivery locations. Such problems have been treated very recently, e.g. in Irnich (2000).

In this paper we look at the second stage, i.e. the transportation processes between the distribution centers. At the distribution centers orders, requiring drop-off at the same destination, are consolidated to full truckloads. Therefore, trucks moving between any two distribution centers are fully loaded and thus go directly from their source to their destination. Clearly the time windows at the customers lead to corresponding time windows at the two distribution centers associated with an order.

The primary goal at this stage is to minimize total costs associated with the satisfaction of all truckload movements. These costs consist of two cost factors. The first one is associated with the fleet size required to perform the deliveries, while the second one covers the costs associated with the actual movements, loaded as well as empty, by the utilized fleet.

The algorithms described below solve this problem subject to the following assumptions:

1. All orders are known in advance.
2. All orders are consolidated to full truckloads.
3. Time windows for each order have to be respected strictly.
4. A tour must not exceed a given time-span.
5. Each truck is assigned to a specific depot, where it has to return to after each tour.

The exact formulation for our problem can be found in Doerner et al. (2000). An exact solution procedure for a similar problem can be found in Desrosiers et al. (1988), where a situation with full truckloads is considered. However, they do not deal with time window constraints. Apart from that, their algorithm can only solve very small problems. As our aim is to solve problems of larger real world applications we chose to develop an appropriate heuristic.

2.2 Graph-based model formulation

Let $J = \{1, ..., n\}$ denote the set of orders, and $D = \{1, ..., m\}$ denote the set of distribution centers, which are the home depots of the trucks. Then, the problem considered in this paper can be represented

by a weighted directed graph $G = \{V, A, d\}$, where $V = \{v_1, v_2, v_3, ..., v_n, v_{n+1}, ..., v_{n+m}\}$ is a set of vertices. The vertices v_1 to v_n denote the orders 1 to n (we will refer to this subset of V as V_o), the vertices v_{n+1} to v_{n+m} denote the depots 1 to m (we will refer to this subset of V as V_d). $A = \{(v_i, v_j); i \neq j, V \times V \setminus V_d \times V_d\}$ is a set of arcs. An arc (v_i, v_j) represents the empty vehicle movement required between orders i and j, and is weighted with the non-negative distance d_{ij}. Figure 1 below shows the graph for a problem with 4 orders and 2 depots.

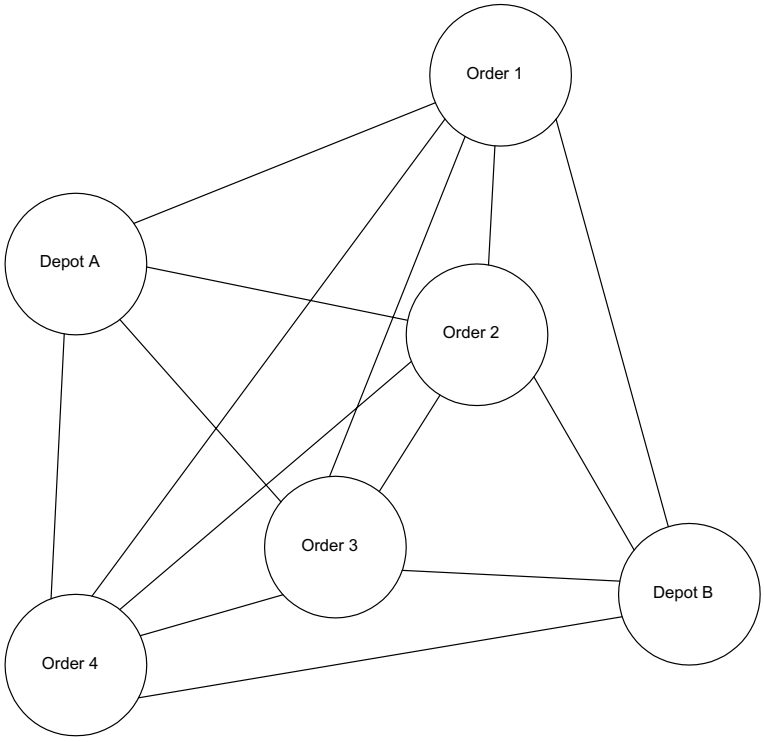

Figure 1: Graphical representation (2 depots, 4 orders)

Based on this graph our optimization problem can be stated as follows. Subject to the constraint, that all order nodes $v_i \in V_o$ are visited exactly once and each cycle contains exactly one depot node $v_j \in V_d$, the problem of finding the minimal fleet size corresponds to finding the minimum number of cycles in the graph, while the problem of minimizing total vehicle movements corresponds to finding a number of cycles, such that the total length of all cycles is minimal.

The approach followed in this paper is to solve these objectives simultaneously rather than sequentially.

3 Heuristic solution procedures

Let us now describe how we implemented the ACO algorithm for the problem at hand. First, we briefly describe the two basic ACO phases, namely the construction of a feasible solution (in subsection 3.1) and the trail update (in subsection 3.2). This will be followed by a post-optimization approach which aims to improve the starting depots for each truck. Finally, we will introduce a detailed description of the algorithmic procedure.

3.1 Construction of a feasible solution

The construction of a feasible solution in our ACO algorithm is done as follows:
Starting at time $t = 0$ a truck is sequentially filled with orders. After each order assignment the time t is updated, it is set to the actual delivery time of the last assigned order. The order assignment is continued until the end of the planning horizon T is reached, or no more order assignment is feasible. At this point another vehicle is brought into use, t is reset to $t = 0$ and the order assignment is continued. This procedure is repeated until all orders are assigned.

For the selection of orders that have not yet been assigned to trucks, two aspects are taken into account: how promising the choice of that order is in general, and how good the choice of that order was in previous iterations of the algorithm. The former information is the visibility, mentioned in section 3.1.1, the latter is stored in the pheromone information, mentioned in section 3.1.2.

3.1.1 Visibility

The visibility information is stored in a matrix $\eta(t)$. Each matrix element $\eta_{ij}(t)$ is positive, if and only if the assignment of order j after order i is feasible. An assignment of order j is feasible, if the order can be scheduled on the current vehicle without violation of its time window. Hence, it is clear that η depends on the time. Note that in each iteration only the row associated with the order assigned in the previous iteration has to be evaluated. The actual value of the visibility of order j depends on the priority rule incorporated in the algorithm.

It is quite obvious that the choice of the priority rule substantially influences the solution quality. However, this influence also depends on the problem characteristics. Thus, we compared different priority rules for our numerical analysis. The first one is a combination of two measures which we will now briefly discuss.

- EDD (earliest due date) - this measure assigns priority to orders according to their latest finishing time (LFT), such that the order with the earliest LFT has highest priority. Thus, this measure favours orders which are relatively due, regardless of their origin and destination.
- EPST (earliest possible starting time) - using this measure orders are assigned priorities according to their EPST. This EPST depends on three criteria: the finishing time (FT) of the last order assigned, the empty vehicle movements required to reach the pickup location of the considered order (DIST) and the considered order's earliest starting time (EST). Given these data, the EPST of an order j is calculated as $EPST_j(i,t) = max[FT(i) + DIST(i,j), EST(j)]$, where order i is the previously assigned order. Highest priority is given to the order with the earliest EPST. Note, that the empty vehicle movements depend on the fact, whether or not the considered order can be performed on the current round trip without violating the maximum tour length constraints. If both customers can be served on one roundtrip without violating the maximum tour length constraint, this distance is simply the distance between the two locations. If the customers can not be served on the same roundtrip, the vehicle has to return to its home depot before visiting the considered customer.

The second priority rule is solely based on the distance traveled to get from the delivery location of the last customer assigned (i) to the pickup location of customer j, $DIST(i,j)$.

The two priority rules used are:

$$\eta_{ij}(t) = \begin{cases} e^{-4\cdot(EDD_j + 2\cdot EPST_j(i,t))} & \text{if the assignment of order } j \text{ is feasible} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in J, \quad (1)$$

and

$$\eta_{ij}(t) = \begin{cases} e^{-16 \cdot DIST(i,j)} & \text{if the assignment of order } j \text{ is feasible} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in J. \tag{2}$$

In what follows, we will refer to the priority rule given in (1) as *Time windows rule (TW rule)*, and to the priority rule given in (2) as *Distance rule* . We have also investigated several other priority rules, which combine the measures presented above. However, the ones presented in this paper turned out to be best.

3.1.2 Pheromone information

The pheromone information can be decoded in two different ways. Below we will describe both decoding approaches, which differ in the decoding of the information concerning the home depots of the vehicles.

The value τ_{ij}, which is associated with each arc (v_i, v_j), represents the current pheromone information. In both decoding schemes for $i \leq n$, $j \leq n$, the value τ_{ij} represents the pheromone information of assigning order j immediately after order i.

In the first decoding approach the value τ_{ij}, $i \leq n$, $j \geq n+1$ represents the pheromone information to begin a new tour with another truck in depot j provided that order i is the last order on the current vehicle.

On the contrary, in the second decoding approach, the pheromone information for choosing a certain home depot for a vehicle is stored in an extra array and is independent from the last orders assigned to any vehicle and the home depots of all other vehicles.

Finally, for both approaches τ_{ij}, $i \geq n+1$, $j \leq n$ represents the pheromone information for order j being the first order starting from depot i.

From now on we will refer to the first approach as *Pheromone matrix 1* and to the second approach as *Pheromone matrix 2*.

3.1.3 Decision rule

Given the visibility and pheromone information described above, and $\Omega_i(t) = \{j \in J \cup D : \exists \eta_{ij}(t) > 0\}$, order or depot j is selected to be visited immediately after order or depot i according to a random-proportional rule that can be stated as follows:

$$\mathcal{P}_{ij}(t) = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{h \in \Omega} [\tau_{ih}]^\alpha [\eta_{ih}(t)]^\beta} & \text{if } j \in \Omega_i(t) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This probability distribution is biased by the parameters α and β that determine the relative influence of the trails and the visibility, respectively.

3.2 Trail update

After an artificial ant has constructed a feasible solution, the pheromone trails are updated. We use a pheromone update procedure, where only a number of the best ants, ranked according to solution quality, contribute to the pheromone trails. Such a procedure was proposed in Bullnheimer et al. (1999a) and Bullnheimer et al. (1999b). The update rule is as follows:

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + \sum_{\lambda=1}^{\Lambda} \Delta\tau_{ij}^{\lambda} \qquad \forall v_i, v_j \in V \tag{4}$$

where ρ is the trail persistence (with $0 \leq \rho \leq 1$). Only the Λ best ants update the pheromone information. If an arc (v_i, v_j) was used by the λ-th best ant the pheromone trail is increased by a quantity $\Delta\tau_{ij}^{\lambda}$.

This update quantity can be represented as

$$\Delta\tau_{ij}^{\lambda} = \begin{cases} 1 - \frac{\lambda - 1}{\Lambda} & \text{if } 1 \leq \lambda \leq \Lambda \\ \\ 0 & \text{otherwise} \end{cases}$$

3.3 Post-Optimization

The procedure described above in section 3.1.1 assigns each truck a home depot before orders are assigned. The actual choice of an order is only partly influenced by the depot assignment. Thus, it is not clear if the set of orders finally served by a truck could not be satisfied from another depot with lower vehicle movement costs. To overcome this drawback we implemented a post-optimization technique which seeks to improve a solution by finding the optimal depot for each truck given the orders assigned. For each truck all possible depot assignments are evaluated and the one yielding lowest costs is chosen. Note that the trucks can be optimized independently of each other, as order re-assignment is not allowed. Neither the sets of orders assigned to the trucks nor the sequence of orders on the trucks can be changed.

As this procedure is rather time-consuming we chose the following approach. In each iteration the solutions of the ants are sorted with respect to their relative fitness. At this point the optimization procedure is performed only for the Λ best ants which also are chosen for the pheromone update. After the post-optimization the solutions are re-sorted, as their objective values might have changed, and the pheromone update is performed.

3.4 The Ant Colony Algorithm

In the initialization phase, Γ ants are generated, and each depot is assigned the same number of ants. Then the two basic phases - construction of tours and trail update - are executed for a given number of iterations. To improve the solution quality a post optimization procedure will be applied. The proposed ant system can be described by the algorithm given in Table 1.

4 Numerical analysis

In this section we will present extensive numerical studies performed to test the proposed heuristics. We will start our discussion with a description of the benchmark problems and the parameter settings used for our simulations. After that we will provide a technical analysis of our ACO algorithm. There we will show the impact of the pheromone information on the solution quality, discuss possible improvements through post-optimization and analyze the two priority rules incorporated in the problem-specific heuristic information. We will also discuss the tradeoff between computational effort and solution quality and provide a large set of benchmark problems with solutions obtained by our algorithms.

Table 1: The ACO algorithm

```
procedure AntColonyOptimization {
  Initialization of the Ant System;
  set a number of ants on each depot;
  for i := 1 to maxIterations {
    for Ant := 1 to Γ {
      while not all orders are assigned {
        initialize a new truck;
        t = 0;
        select a home base for the truck;
        while ∃ η_ij(t) > 0        ∀i, j ∈ J {
          select an order using formula (3);
          update t;
        }
      }
      compute the number of trucks required;
      evaluate the objective function;
    }
    for λ := 1 to Λ {
      improve the solution using the post optimization procedure
      (see subsection 3.3);
    }
    compute the number of trucks required;
    evaluate the objective function;
    update the pheromone trails using formula (4);
  }
}
```

4.1 Parameter settings

In order to analyze the performance of our heuristics we tested them on a number of randomly generated problems, which are all based on the following input data:

- 8 distribution centers,
- 512 orders,
- a maximum tour length of 2 periods and
- a planning horizon of 8 periods.

For each distribution center we chose x- and y-coordinates randomly from the interval (0,100]. All our test problems have the geographical setting in common, in order to obtain comparable results.

For each order the following data were generated: pickup and delivery location, both chosen from the set of distribution centers, and time windows, i.e. an earliest possible pickup time and a latest possible delivery time.

For our technical analysis we generated 5 order constellations (denoted A, B, C, D and E). For each order constellation we generated one problem with tight time windows, with an average length of 2 periods, and one problem with wide time windows, with an average length of 7 periods, by drawing the values from a binomial distribution. In the remainder of this paper we will refer to these problems as A2, A7, B2, ... , E2, E7.

The parameter settings for the ACO algorithm given in Table 2 were chosen in accordance with known, good parameter settings from literature (see e.g. Bullnheimer et al. (1999a), Bullnheimer et

Table 2: Parameter settings for the ACO algorithm

Parameter	value
α	1
β	1
ρ	0.5
τ_0	0.1
Λ	20
Γ	80
$maxIterations$	30

al. (1999b)). While we did vary these parameters within reasonable ranges, extensive testing was not performed and fine-tuning of these parameters might yield some more improvements.

As stated above, the objective in our problem is to minimize total costs which consist of fixed costs for the vehicle fleet and variable costs depending on the total distance traveled. For our numerical analysis we chose the following objective function:

$$TC = 20 \cdot FS + 1 \cdot d_{tot},$$

where TC are the total costs, FS is the fleet size and d_{tot} is the total distance traveled by all vehicles. Each vehicle utilized costs 20 units for the planning horizon, while the costs for a 'moving' truck are 1 unit per day. For an interpretation of these cost factors see Doerner et al. (2000).

4.2 Technical analysis of the ACO algorithm

In this subsection we will focus on the comparison of different versions of the Ant Algorithm proposed in this paper. First we will discuss the value of the pheromone information and the importance of choosing an appropriate data structure for the pheromone information. After that we will examine the influence of the post-optimization technique on the solution quality. This will be followed by a comparison of the results associated with the two priority rules. Finally, we will show the relationship between computational effort and solution quality and provide solutions for an extended set of benchmark problems.

4.2.1 The impact of the pheromone information on the solution quality

An important feature of ant algorithms is the fact that good solutions from the past influence decisions when new solutions are built. Without this memory from the past an unguided stochastic search would be performed. Thus, an interesting question is how strongly the pheromone information influences the solution quality. To answer this question we compare the performance of our algorithm with and without memory.

Figure 2 shows the evolution of the solution quality for the problem A7. Clearly similar figures could be obtained for all other problems. From Figure 2 it is obvious that the influence of the pheromone information significantly improves the solution quality. First, the solutions found with the ACO are much better than the solution of the unguided stochastic search. Second, the ACO converges, while the stochastic search oscillates. We also found that the pheromone information has a balanced positive influence on both the fleet size and the total vehicle movements. Fleet sizes as well as empty vehicle movements can be significantly reduced through the use of the memory.

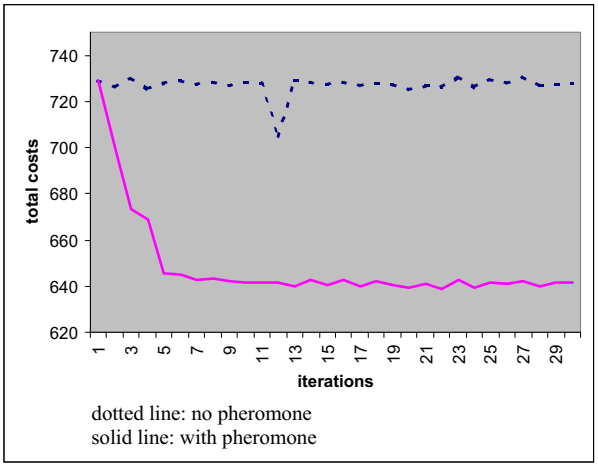

Figure 2: Convergence of the ACO algorithms (*TW rule*, no post-optimization, *Pheromone matrix 1*)

Now that we showed that the use of an ant colony approach is justified for the problem at hand, we will turn to the more interesting question how different pheromone update schemes affect the behaviour of the algorithm. To that end we can compare the two decoding schemes proposed in the last section. The following Table 3 shows the percentage deviations between the solutions associated with the different decoding schemes for various parameter settings.

Table 3: Percentage deviation of the solution associated with *Pheromone matrix 1* from the solution associated with *Pheromone matrix 2*

Setting	Deviation
no post-optimization/ narrow time windows	-1.7332
no post-optimization/ wide time windows	-0.13464
with post-optimization/ narrow time windows	-0.24366
with post-optimization/ wide time windows	-0.05142

From Table 3 we can see that *Pheromone matrix 1* outperforms *Pheromone matrix 2* for any setting. Thus, for all the numerical studies discussed below we used only *Pheromone matrix 1*. However, the size of the effect depends whether or not the post-optimization technique is used. In the next subsection we will take a close look at the performance of the post-optimization technique.

4.2.2 The impact of the post-optimization technique on the solution quality

The post-optimization method proposed in subsection 3.3 was used to a posteriori improve the base depots for all utilized vehicles. The aim of this procedure was to further improve the vehicle movements required to satisfy all customer orders. Thus, in Table 4 empty vehicle movements obtained for the cases with and without post-optimization are reported.

The average improvements through post-optimization range between 4.582 % and 16.288 % and depend on the choice of the priority rule incorporated in the heuristic information. For one problem instance we even found a smaller fleet size through the use of the post-optimization procedure. As we applied our post-optimization only to promising solutions, the computational effort was rather small.

Table 4: Percentage improvement of the solution quality through the use of a post-optimization technique.

Setting	empty vehicle movements without post-optimization	empty vehicle movements with post-optimization	improvement through post-optimization in %
TW rule/ narrow time windows	43.672014	36.8875243	15.535
TW rule/ wide time windows	26.7206219	22.3683514	16.288
Distance rule/ narrow time windows	35.640652	32.6206068	8.474
Distance rule/ wide time windows	16.9914388	16.212823	4.582

Thus, these results justify the use of the post-optimization and for all our further studies we will report only results for runs where the post-optimization was performed.

In the next subsection, we will discuss the two priority rules proposed in section 3.1.1.

4.2.3 The impact of two different priority rules on the solution quality

The analysis in this section is based on an extended set of benchmark problems. This set of test-problems consists of the problems used for the technical analysis of the Ant-System above, and some new problems generated using the same mechanism as described in section 4.1. For each order constellation A-E we additionally generated problems with an average length of customer time windows of 1,3,4,5,6 and 8 periods. Thus, we altogether have 40 Testproblems. The results in this section are based on averages over the five order constellations.

As outlined in section 3.1.1 we used two different priority rules to guide the ants through the search space. Now, we want to show the differences in the solution quality associated with the two rules. Let us first look at the total vehicle movements. The results in Figure 3 show that *Distance rule* outperforms *TW rule* in all cases with respect to the vehicle movements.

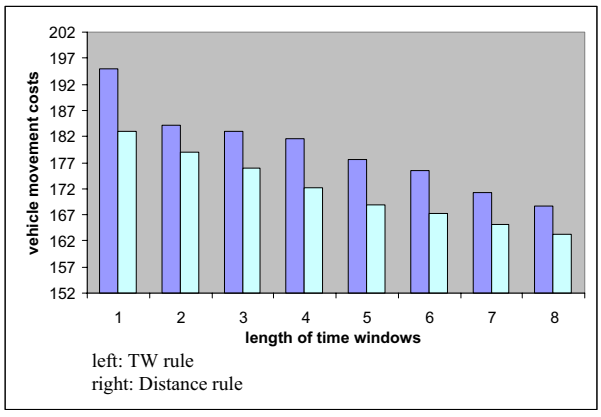

Figure 3: Total vehicle movements associated with the two priority rules

This result can be understood, if one considers that *TW rule* aims to assign as many orders as possible on a truck, while *Distance rule* always aims to find the best successor for the order previously assigned to a vehicle. Based on these characteristics, one would expect that for the minimization of

fleet sizes the performance of the two priority rules should be the opposite way. Indeed, we see from Figure 4, that *TW rule* outperforms *Distance rule* for all cases with time windows. Only the problems with average time window lengths 8 - note that this means that time windows do not restrict the solution space - *Distance rule* finds smaller fleet sizes.

Given the weight of the fixed costs in our objective function, this leads to the results shown in Figure 5. Here the total costs for both Priority rules and all problems are depicted. From this figure we see that *TW rule* finds generally better solutions than *Distance rule*. As was true for the fleet size, *Distance rule* is favourable only for problems without time-window restrictions.

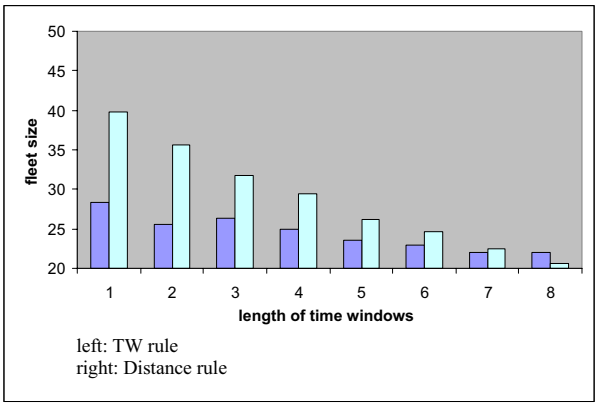

Figure 4: Fleet sizes associated with the two priority rules

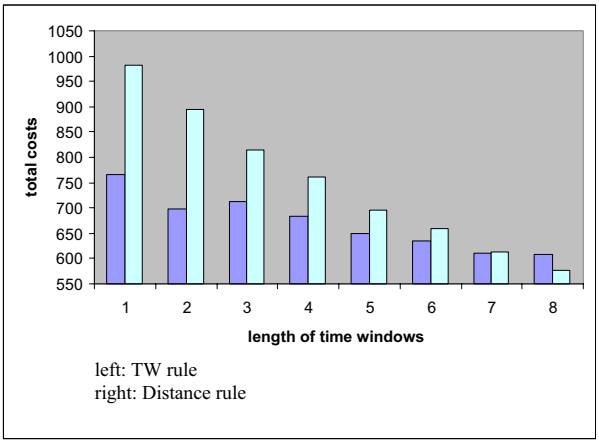

Figure 5: Total costs associated with the two priority rules

However, the results show that the differences between the two priority rules are largest if time windows are very tight, and monotonically decrease with increasing length of the time windows. Thus, a priori it is not clear which priority rule to use.

4.2.4 Computational effort and solution quality

For all our simulations in the last section, we used populations of 80 ants. While this was chosen to find good solutions as fast as possible, we will now show how computational effort as well as solution quality increase with the number of solutions generated.

In an ACO the two different ways of increasing the number of solutions are to increase the number of iterations and to increase the population size. We found that increasing the number of iterations, to a level exceeding 30 iterations, has only minor influence on the solution quality. Thus all our results are reported for runs of 30 iterations.

With respect to population size we compared values of 40, 80, 160, 320, 640 and 960 ants. Our results are shown in Table 5.

Table 5: Influence of the population size on the solution quality

Problem	Population size					
instance	40	80	160	320	640	960
A1	811.687	784.646	779.136	755.852	756.313	752.056
A2	734.895	727.383	705.701	700.803	683.290	681.048
A3	731.740	706.303	702.503	684.989	677.280	661.610
A4	708.936	683.079	664.301	658.152	653.714	633.373
A5	686.862	661.514	657.671	635.128	632.779	614.634
A6	666.249	639.901	635.178	613.269	610.927	609.739
A7	629.011	615.679	605.336	606.549	587.405	585.841
A8	586.524	585.021	584.285	563.051	562.917	562.478

For the sake of comparability, the results shown in Table 5 are those for problem class A. However, similar results were achieved for the other problems. From the table we see that increasing population size yields significantly better results. We also see that problems with medium length time windows seem to be the hardest ones to solve as for those problems a population size of 960 ants could save one vehicle as compared to the results with population size 640. We also see that the difference between the objective values associated with population sizes 640 and 320, respectively is small. In order to be able to evaluate these findings with respect to the computational effort needed, we show the trade-off between population size and computational effort in Table 6.

Table 6: Trade-off between population size and computational effort

Population size	40	80	160	320	640	960
Computation time in CPU minutes	33	68	131	254	502	772
Avg. Total Costs	694.488	675.441	666.764	652.224	645.578	637.597

The results reported in Table 6 are average values over the problems A1-A8 for each population size. We see that the computational effort increases linearly. Generally, we see that the difference in the solution quality between a population size of 40 ants and 960 ants equals approximately 60 cost units, while the computational effort for 960 ants is 24 times larger than the one associated with 40 ants. An improvement of 30 cost units can be achieved with only 4 times the computational effort associated with 40 ants. Thus, 160 ants seem to guarantee a good balance between solution quality and computational effort.

We will finally provide all our results for the extended set of benchmark problems in order to enable researchers to compare their solutions with the solutions found using the ACO proposed in this paper.

Table 7: Empty vehicle movements and fleet size (in brackets): Our best solutions compared to lower bounds

Average length of time windows	Problem class A	B	C	D	E
1	204.646 (29)	200.293 (27)	197.816 (29)	191.699 (28)	191.938 (29)
2	187.383 (27)	186.849 (25)	186.793 (26)	186.011 (25)	184.969 (25)
3	186.303 (26)	180.619 (26)	186.863 (27)	182.479 (27)	182.690 (26)
4	183.079 (25)	179.578 (25)	182.661 (25)	184.507 (25)	183.249 (25)
5	181.514 (24)	176.644 (24)	179.088 (24)	178.963 (23)	174.224 (23)
6	179.901 (23)	175.199 (23)	174.238 (23)	174.763 (23)	175.256 (23)
7	175.679 (22)	164.782 (22)	163.877 (22)	167.082 (22)	172.049 (22)
8	165.021 (21)	161.697 (20)	162.659 (20)	163.778 (21)	164.650 (21)
Lower bounds	159.203 (19)	156.207 (19)	155.096 (19)	157.820 (19)	159.157 (19)
Loaded vehicle movement costs	153.22	149.33	150.58	148.30	146.14

In Table 7 we provide for each problem class A-E information about the compulsory loaded vehicle movements, a lower bound for the total vehicle movements, a lower bound for the fleet size as well as the empty vehicle movements and fleet sizes associated with the best solutions found with our ACO using populations of 80 ants. The lower bound on the total vehicle movements was generated by solving a network flow problem, which is formulated in Gronalt et al. (2000). The lower bound on the fleet size is calculated by dividing the lower bound on the total vehicle movements by the maximum time one truck can operate, which is 8.5. This number comes from the length of the planning horizon, which for our simulations was 8, and the largest distance between any two distribution centers, which was 0.5 in our simulations. The latest possible due time for any order is the end of the planning horizon. Thus, in the worst case, a truck reaches its final delivery location at time 8. Because of the largest distance of 0.5 periods, the latest possible time to return to the home depot is 8.5.

We see that the lower bounds are independent from the time windows, as these constraints were relaxed in order to obtain the lower bounds. Due to this relaxation it is obvious, that these lower bounds are bound to be rather poor for problems with tight time window constraints, while they are good for problems with wide time window constraints.

5 Conclusions and future research

In this paper we have dealt with a technical analysis of an ACO algorithm applied to a special case of the pickup and delivery problem with time window constraints. Our objective function was to minimize total costs, i.e. to optimize both the fleet size and empty vehicle movements. We proposed two different data structures for the pheromone information. Our results showed that the solution quality associated with the two approaches was differing significantly.

Furthermore we compared two different visibility functions. We tested our algorithms for a wide range of parameter constellations and found that neither of the two priority rules incorporated in the visibilty information was superior with respect to all problem instances.

Finally we also proposed a post-optimization technique which improved our solutions with respect to empty vehicle movements by approximately 11 %.

Together these results imply the following findings:

1. In order to find the appropriate data structure for the pheromone information a solid understanding of the problem in general is necessary.

2. To obtain good solutions the user has to choose the appropriate priority rule. Thus, he needs thorough knowledge of the problem characteristics and the implications of the actual parameter constellation for the choice of the priority rule.

Both findings show that sound problem specific knowledge is necessary to implement an ACO which finds high quality solutions. However, the first finding is concerned with an implementation issue, which has to be dealt with once for a special class of problems. On the contrary, the second finding is related to the actual use of the ACO for a given problem instance. While for some problems it may be easy for the user to determine the structure of the problem and to choose the appropriate priority rule to be incorporated in the visibility information, it would be preferable to make this choice independent from the actual user.

Thus, our aim for future research is to find and implement concepts which allow to endogeneously choose an appropriate priority rule.

Acknowledgments

The authors are grateful for financial support from the Austrian Science Foundation (FWF) under grant SFB #010 'Adaptive Information Systems and Modelling in Economics and Management Science' and from the Oesterreichische Nationalbank (OENB) under grant #8630.

References

Bullnheimer, B., R. F. Hartl, Ch. Strauss. 1999. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research* **89**, 319–328.

Bullnheimer, B., R. F. Hartl, Ch. Strauss. 1999. A new rank based version of the Ant System. *Central European Journal of Operations Research* **7**(1), 25–38.

Colorni, A., M. Dorigo, V. Maniezzo. 1991. Distributed Optimization by Ant Colonies. F. Varela, P. Bourgine, eds. *Proc. Europ. Conf. Artificial Life*, Elsevier, Amsterdam.

Costa, D., A. Hertz. 1997. Ants can colour graphs. *Journal of the Operational Research Society* **48**(3), 295–305.

Desrosiers, J., G. Laporte, M. Sauve, F. Soumis, S. Taillefer. 1988. Vehicle routing with full loads. *Computers and Operations Research* **15**(3), 219–226.

Doerner, K. F., M. Gronalt, R. F. Hartl, M. Reimann. 2000. Optimizing Pickup and Delivery Operations in a Hub Network with Ant Systems. POM Working Paper 02/2000, Department of Production and Operations Management, University of Vienna, Vienna, Austria.

Dorigo, M. 1992. Optimization, Learning and Natural Algorithms. Doctoral Dissertation, Politecnico di Milano, Italy.

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

Dorigo, M., L. M. Gambardella. 1997. Ant Colony System: A cooperative learning approach to the Travelling Salesman Problem. *IEEE Transactions on Evolutionary Computation* **1**(1), 53–66.

Dorigo, M., G. Di Caro, G. (1999), The Ant Colony Optimization Meta-Heuristic. D. Corne, M. Dorigo, F. Glover, eds. *New Ideas in Optimization*, Mc Graw-Hill, London.

Dorigo, M., G. Di Caro, L. M. Gambardella. 1999. Ant Algorithms for Discrete Optimization. *Artificial Life* **5**(2), 137-172.

Gronalt, M., R. F. Hartl, M. Reimann. 2000. Time Constrained Pickup and Delivery of Full Truckloads. POM Working Paper 02/2000, Department of Production and Operations Management, University of Vienna, Vienna, Austria.

Gutjahr, W. J. 2000. A graph-based Ant System and its convergence. *Future Generation Computing Systems* **16**, 873–888.

Irnich, St. 2000. A Multi-Depot Pickup and Delivery Problem with a Single Hub and Heterogeneous Vehicles. *European Journal of Operational Research* **122**(2), 310-328.

Savelsbergh, M. W. P., M. Sol. 1995. The general pickup and delivery problem. *Transportation Science* **29**(1), 17–29.

Stützle, T., M. Dorigo. 1999. ACO Algorithms for the Quadratic Assignment Problem. D. Corne, M. Dorigo, F. Glover, eds. *New Ideas in Optimization*, Mc Graw-Hill, London.