Ant Colony Optimization for dynamic Traveling Salesman Problems

Carlos A. Silva and Thomas A. Runkler

Siemens AG, Corporate Technology
Information and Communications, CT IC 4
81730 Munich - Germany
thomas.runkler@siemens.com

Abstract:
This paper addresses the optimization of a dynamic Traveling Salesman Problem using the Ant Colony Optimization algorithm. Ants are social insects with limited skills that live in colonies able to solve complex problems. The intelligence of the global society arises from self organization mechanisms, based on the indirect communication between individuals through pheromones. The routing problem here presented is a typical case that requires a self organization type of algorithm, in order to cope with the problem dynamics. The simulation results show how the ant colony optimization is able to solve the different possible routing cases.

1 Introduction

The social insect metaphor has been increasingly used in the last ten years for solving different types of engineering problems [BDT99]. This approach emphasizes the distribution, communication, flexibility and robustness of simple agents, capable of achieving a form of collective artificial intelligence. The individual agents are not able to solve the problem by themselves, but a solution emerges from the cooperative behavior of all associated individuals. This is particularly interesting at a time where systems are becoming more and more complex and the design of a single control system able to cope with all the objectives is virtually impossible.

Individual insects living in colonies, such as ants, bees or wasps do their own specific task and yet the colony is very organized and it does not require any kind of supervision. This is called *self organization* (SO). Theories of self organization [NP77], originally developed in the context of physics and chemistry can be extended to social insects to explain how the complex behavior emerges from the interaction among individuals that exhibit simple behavior [DGFP91]. The mechanism that allows the SO in insect colonies is the *stimergy*, which consists of the indirect communication between pairs of individuals through the change of some environment property; in the ant colonies case, this property is the pheromone concentration.

The daily problems solved by an insect colony includes finding food, building or extending a nest and feeding the brood. These problems have counterparts in engineering and computer science areas, like routing optimization, data clustering, robotics, etc [BDT99]. *Ant Colony Optimization* (ACO) is an optimization algorithm inspired in the collective foraging behavior of ants to find and exploit the food source that is closest to the nest. The first application was the *Traveling Salesman Problem* (TSP) [DMC96], which can be seen as a direct translation of the food search problem. Soon, the metaphor was extrapolated for different types of routing problems, like the vehicle routing problem [GrTA99].

The problem we have to solve here is a special case of a routing problem, with a common topology to the TSP problem, but with dynamic constraints. This makes the ACO an appealing approach for this problem, since it uses pheromone trails that keep track on the optimization steps. In cases of changes, the algorithm does not need to restart the complete procedure, but only to adapt the results to the new conditions.

2 The Traveling Salesman problem for Cash Machines

Cash Machines (CM) are computer terminals operated by banks from which the clients of any bank can withdraw money. From time to time, a cash logistics company visits the machine and changes the money containers. This routing problem can be modeled as a Traveling Salesman Problem, where the location of each CM is a city and the cash logistics company starts and ends the distribution always at the central bank office.

For the banks in charge of the CM, there are two contradictory situations to avoid: having empty CM or full CM, caused by machine service rates different from the cash replacement rate of the logistics company. In the first case the bank customers are not satisfied with the quality of service, and in the second case the bank loses interests for the stocked money.

After each transaction each machine reports its money stock f_i to the bank. The money stocks are one criterion for the decision which CMs should be recharged. If a machine still has a lot of money, it makes no sense to recharge it at all; if the CM is not empty yet, but will possibly be before the next distribution, it might be considered to be recharged. This decision depends on another criterion: the location of the machines. There is a cost for visiting a machine and maybe this cost is higher than the cost of having a machine empty for one or two days before the next distribution. But if this machine is very near to another machine that is going to be recharged, the cost of recharging this one also is very small. More than the absolute location of the machine, the decision therefore depends on the relative location to other CMs.

The distribution planner has to take into consideration both the money levels of the machines and their relative locations. This routing problem is a special case of the TSP: there are several locations to visit and the travel begins and ends at the same location. However there are two additional constraints:

- If the machine stock is high, i.e. $f_i > 60\%$, the machine is not visited.
- If the machine stock is low, i.e. $f_i \leq 60\%$ and the product of the stock level f_i and

the distance to travel is higher than 20% of the maximum distance to travel, i.e.

$$d_{ij} \times f_i > 0.2 \max_k \{d_{ik}\}, \tag{1}$$

then the machine is not visited. This constraint depends on the planned route, therefore it is a dynamic constraint.

The first constraint can be evaluated before routing, but the second constraint depends on the solution. This is a case where the use of a self organization algorithm like the ACO is most interesting. The ACO is a construction algorithm. The solution is built step by step, and in each step the algorithm can check the constraint for empty machines: in case the distance is smaller than the constraint, the machine is included in the tour, if not, it is excluded. Notice that with other types of planners using e.g. simulated annealing or genetic algorithms this is not possible, since the solutions are constructed in one single step and cannot deal with the dynamics of this problem.

3 Ant colony optimization

The meta heuristic Ant Colony Optimization (ACO) is an optimization algorithm successfully used to solve many NP hard optimization problems introduced in [DMC96]. ACO algorithms are a very interesting approach to find minimum cost paths in graphs especially when the connection costs in the graphs can change over time, i.e. when the problems are dynamic. The artificial ants have been successfully used to solve the (conventional) Traveling Salesman Problem (TSP) [DMC96], as well as other NP hard optimization problems, including applications in quadratic assignment [SH00] or vehicle routing [GrTA99].

The algorithm is based on the fact that ants are always able to find the shortest path between the nest and the food sources, using information of the pheromones previously laid on the ground by other ants in the colony. When an ant is searching for the nearest food source and arrives at several possible trails, it tends to choose the trail with the largest concentration of pheromones τ, with a certain probability p. After choosing the trail, it deposits another pheromone, increasing the concentration of pheromones in this trail. The ants return to the nest using always the same path, depositing another portion of pheromone in the way back. Imagine then, that two ants at the same location choose two different trails at the same time. The pheromone concentration on the shortest way will increase faster than the other: the ant that chooses this way, will deposit more pheromone in a smaller period of time, because it returns earlier. If a whole colony of thousands of ants follows this behavior, soon the concentration of pheromone on the shortest path will be much higher than the concentration in other paths. Then the probability of choosing any other way will be very small, and only very few ants among the colony will fail to follow the shortest path. There is another phenomenon related with the pheromone concentration. Since it is a chemical substance, it tends to evaporate, so the concentration of pheromones vanishes along the time. In this way, the concentration of the less used paths will be much lower than that of the most used ones, not only because the concentration increases on the other paths, but also because its own concentration decreases.

The artificial ants mimic this behavior in a disjunctive graph environment, with nodes and arcs between the nest and the food source. They can be uploaded with more characteristics, e.g. memory and heuristic information of the problem. If the pheromone expresses the *experience* of the colony in the job of finding the shortest path, memory and heuristic information express useful *knowledge* about the problem the ants are solving. In this way, the probability of choosing the next trail is given by:

$$p_{ij}^{k}(t) = \begin{cases} \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta} \,/\, \sum\limits_{k \notin \Gamma}^{m} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta} & \text{if } j \notin \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where τ_{ij} is the pheromone concentration in the path (i, j), η_{ij} is a *heuristic function* and Γ is a *tabu list*. The heuristic function η conducts the search with some valuable information of the problem under optimization. The tabu list Γ is a list that contains all the trails that the ant has already passed and must not be chosen again, acting as the memory of the ant. The parameters α and β measure the relative importance of trail pheromone (experience) and local heuristic (knowledge), respectively.

At each iteration, a new colony of g ants is released. After finding their own way, the ants deposit pheromones on the paths, usually at the end of one tour. A *tour* is a complete route between the nest and the food source and an *iteration* t is a step from i to j done by all the g ants. The update of the pheromone concentration in the trails is given by

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

where $\rho \in [0, 1]$ expresses the pheromone evaporation phenomenon, and $\sum_{k=1}^{g} \Delta\tau_{ij}^{k}$ are pheromones deposited in the trails (i, j) followed by all the g ants after a complete tour, which are defined as

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{f_k} & \text{if the arc } (i, j) \text{ was used by ant } k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where f_k is the value of an evaluation function for each k ant in a minimization problem. With real ants, time acts as a performance index, but artificial ants use all the same time to perform the task, whether they choose a short path or not. In this way, the global update is biased by the solution found by each individual ant. Notice that the time interval taken by the g ants to do a complete tour is $t + m \times g$ iterations. In every N^{th} of the N maximum number of tours, a new *ant colony* is released. The algorithm runs in $O(m \times g \times N) \approx O(N^3)$ time. The general algorithm for the ant colonies is described in Fig. 1.

In our application, which is a particular case of TSP, each node represents the location of a CM. The difference between the TSP and this problem is that not all locations are visited. As explained before, some of the machines are not empty, so it is not always necessary to recharge them. It only pays off a visit if the relation between the emptiness of the CM and the traveling time is smaller than a certain value (1). The heuristic function η in this case is the inverse of travelling time between two machines. In general in the TSP problem, the

```
**procedure** Ant colony algorithm
Set for every pair $(i, j)$: $\tau_{ij} = \tau_{max}$
Place the $g$ ants
**For** $i = 1$ to $N$:
  *Build a complete tour*
    **For** $j = 1$ to $m$
        **For** $k = 1$ to $g$
            Choose the next node using $p_{ij}^k$ in (2)
            Update the **tabu list** $\Gamma$
        **end**
    **end**
    *Analyze solutions*
        **For** $k = 1$ to $g$
            Compute performance index $f_k$
            Update globally $\tau_{ij}(t + m \times g)$ using (3)
        **end**
**end**
```

Figure 1: Ant Colonies Optimization Algorithm

Euclidean distance between two locations d_{ij} is used as heuristic. However, within a city, the traveling time t_{ij} between two machines is more relevant than distance, due to traffic reasons. Therefore, the heuristic function is given by $\eta = (t_{ij} - t_{min})/(t_{max} - t_{min})$, where t_{ij} is the estimated traveling time between location i and location j and $t_{min} = \min t_{ij}$ and $t_{max} = \max t_{ij}$ are the minimum and maximum traveling times considered. In this way, the heuristic matrix η entries are always restricted to the interval $[0, 1]$.

The objective function to minimize, $f^k(t)$, is simply the sum of traveling time between all the visited locations:

$$f^k(t) = \sum_i^S \sum_j^S t_{ij} \tag{5}$$

At the end of each iteration, when all the g ants have visited all the machines the cost function is evaluated and the paths followed by the best ant are updated following the rule in (3).

The algorithm presented here is the ACO implementation called *Max Min Ant System* (MMAS) [SH00]: only the best ant updates the trails in every cycle (see (4)); the pheromone trail is limited to an interval $[\tau_{min}, \tau_{max}]$, in this case to the interval $[0, 1]$. However, unlike the MMAS, the pheromone update mechanism called the *pheromone trail smoothing* is not used here. The set of parameters is tuned using a trial and error approach. The number of ants is $g = 5$ and the number of iterations is $N = 100$. The heuristic function is normalized to the interval $[0, 1]$. The pheromone trails are initialized with the value of τ_{max} and the values $\alpha = 1$ and $\beta = 5$ are used. Since both τ and η are defined in the $[0, 1]$ domain, a small value of α will indicate a higher relative weight to the pheromones trail.

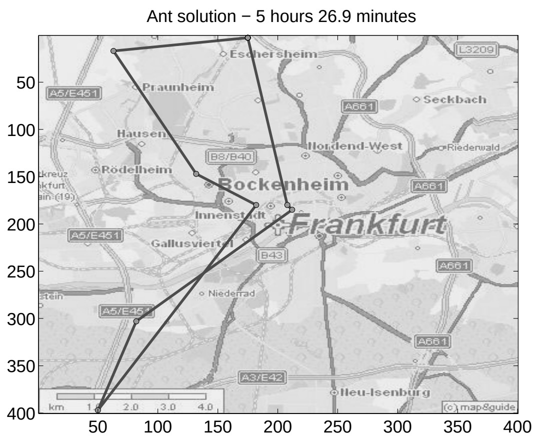


Figure 2: Case 1: All CM are empty

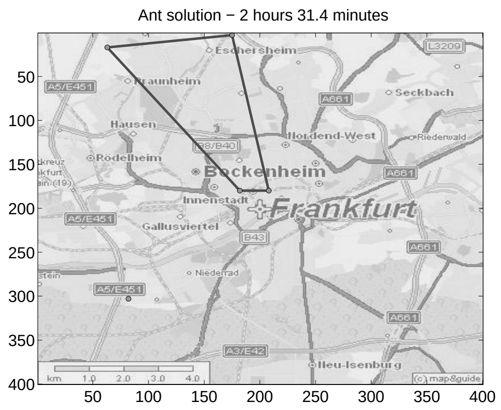
The evaporation coefficient is $\rho = 0.2$.

4 Simulation results

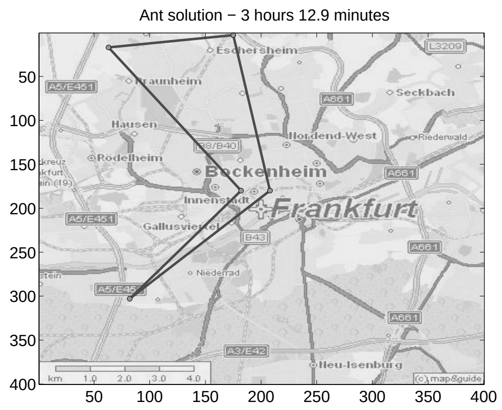
In this section, we simulate the problem described in section 2, where 9 CMs from a specific bank in the city of Frankfurt are maintained by a specific cash logistics company. Once a week, the bank provides the distributer with the information about the money levels of the machines, and the distributer has to decide which machines to visit. As explained, every machine with a money level lower than 60% can be visited by the logistics company, as long as the dynamic constraint regarding stock level and distance is respected (see (1)). The routing planner was developed in Matlab. To show how the algorithm works, we will present three situations: the case where all the machines are completely empty, which turns out to be the generic TSP problem; the case where the money level influences the solution; and finally the case where the solution is mainly biased by the relative location of the CMs.

In the first case, when all the CMs are empty, the routing planner solves a generic TSP problem: all the machines have to be visited and the distribution vehicle starts and ends at the same location. The routing results are presented in Figure 2. The ants have found the path with the shortest traveling time. In the figure, it is obvious that the traveling time is not directly proportional to the Euclidean distance, otherwise, the optimal path could not have crossing arcs.

In the second case the money level is the dominant optimization criterion. In the example depicted in Figure 3, there are five machines to be visited: machine 1, 2, 3 and 4 with $f_1 = f_2 = f_3 = f_4 = 10\%$ and machine 5 with $f_5 = 40\%$. Although the distance between the 3 and 2 machines and 3 and 5 are similar, machine 5 is not included in the trip



(a) $f_1 = f_2 = f_3 = f_4 = 10\%$ *(almost empty) and* $f_5 = 40\%$ *(partially empty)*



(b) $f_5 = 10\%$, *so all machines are recharged*

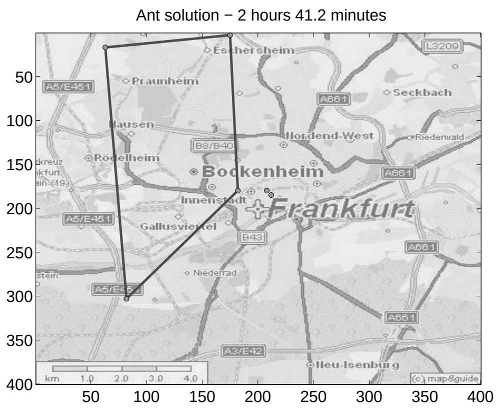
Figure 3: Money level as dominant criterion

(see Fig. 3a). However, if the money level of machine 5 drops to $f_5 = 10\%$, the machine has to be visited and it is included in the trip, as shown in Fig. 3b.

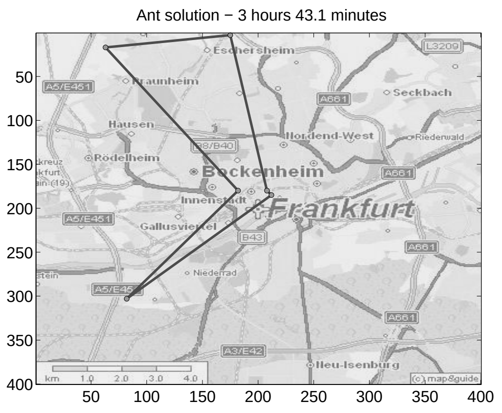
The typical case to be solved by the routing planner is the case where the machines have different money levels f_i, i.e. when the relative location is the dominant criterion. In the example in Figure 4, 6 machines are considered with the respective money levels: $f_1 = 10\%$, $f_2 = 15\%$, $f_3 = 20\%$, $f_5 = 25\%$ and $f_4 = f_6 = 40\%$. Figure 4a shows that the solution is to visit only machines $1, 2, 3$ and 5, since the cost of traveling further to visit machines that are 40% full is higher than not visiting the machines at all. The situation is similar to the one described in the previous case. However, if machine 4 has a stock level of $f_4 = 10\%$ and machine 6 remains with $f_6 = 40\%$, the machine 4 has to be visited. Then, since machine 4 and 6 are so close to each other, the cost of visiting machine 6 becomes sufficiently small, so all the six machines are recharged, as Figure 4b shows.

5 Conclusions

In this paper, we presented an application of the Ant Colony Algorithm to a Traveling Salesman Problem with dynamic constraints. It models the routing problem of a cash logistics company that has to recharge cash machines of a certain bank. The logistics company only visits machines that are empty or when the relation between money level and travel time is sufficiently small. The ACO is an optimization algorithm based on self organization effects of individual agents and it was particularly suited for this application, since one of the optimization constraints depends on how the solution is constructed. The



(a) $f_1 = f_2 = f_3 = f_5 = 10\%$ *(almost empty) and* $f_4 = f_6 = 40$ *(partially empty)*



(b) $f_4 = 10$ *and all the machines are recharged, including* $f_6 = 40$

Figure 4: Relative location as dominant criterion

simulation results show how the ants dynamically find different solutions based on the distances between the machines and the money level, minimizing always the traveling time.

References

[BDT99] Bonabeau, E., Dorigo, M., und Theraulaz, G.: *Swarm Intelligence, From natural to artificial systems*. Oxford University Press. 1999.

[DGFP91] Deneuborg, J.-L., Goss, S., Franks, N. R., und Pasteels, J.-M.: The blind leading the blind: modelling chemically mediated army ant raid paterns. *Journal of Insect Behavior*. 3:356–365. 1991.

[DMC96] Dorigo, M., Maniezzo, V., und Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*. 26(1):29–41. 1996.

[GrTA99] Gambardella, L. M., Éric Taillard, und Agazzi, G.: MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In: Corne, D., Dorigo, M., und Glover, F. (Hrsg.), *New Ideas in Optimization*. S. 63–76. McGraw-Hill. 1999.

[NP77] Nicolis, G. und Prigogine, I.: *Self Organization in Non-Equilibrium Systems*. Wiley and & Sons. 1977.

[SH00] Stützle, T. und Hoos, H.: Max min ant system. *Journal of Future Generation Computer Systems*. 8(16):889–914. 2000.