Using Ant Colony Optimization for Infrastructure Maintenance Scheduling

K. Lukas, A. Borrmann & E. Rank
Chair for Computation in Engineering, Technische Universität München

ABSTRACT: For the optimal planning of maintenance schedules for infrastructural buildings (bridges, tunnels, etc) in urban road systems not only the budget has to be considered but also the impact on traffic to avoid unnecessary traffic jams. In a current research project we develop an optimization tool for this multi-objective problem based on ant colony optimization.
In each iteration, the ants produce several different schedules for the maintenance over the next few years. Each of these schedules is formed by several scenarios of simultaneously closed roads. A parallel maintenance on different buildings can be modeled be introducing teams of ants. The scenarios are evaluated by an external traffic simulator. The quality of the different schedules, assessed by the waiting time created in the system, influences the amount of pheromone deposited on each schedule and therefore the probability that this or a similar schedule is chosen by the ants in the next iteration step.
The building condition also has influence on the probability of choosing a certain schedule: Buildings in bad condition are getting more attractive to be chosen - thus avoiding that only buildings in good condition and therefore with low repair costs are scheduled for maintenance while buildings in bad condition are left to further deterioration.
Additional constraints, e.g. budget constraints, can be introduced by applying a modification of the Elitist Ant strategy that guides the ants away from infeasible schedules.

1 INTRODUCTION

In a current research project we are developing a software tool to assist the planning of maintenance measures for infrastructural buildings (bridges, tunnels, etc).

Traditionally this planning is done by hand. The planner has to consider numerous constraints. Some constraints, as the steadying of the budget flow, which will reduce the administrative effort a great deal, can not be considered with this approach.

Also the impact on traffic by road closings due to maintenance work, whose minimization is a main optimization goal, can only be considered in a very simplified way. The actual benefit of postponing the maintenance at one site to avoid the parallel closure of two given roads can not be evaluated.

The tool we are developing will search a schedule for maintenance work over the next few years that fulfills all given constraints and at the same time keeps the impact on traffic as low as possible.

2 PROBLEM STATEMENT

The first goal of optimizing the maintenance schedule is to minimize the impact on traffic. This can be measured by the waiting time produced by the parallel closure of a set of roads.

In addition to this the schedule is subject to a number of constraints.

The most obvious and also most important task is to schedule the maintenance work of one particular building before this building reaches a state where the security is no longer guaranteed, i. e. the building is likely to collapse.

Besides this security constraint there exist various constraints set by the public and by monetary considerations.

For the public it is important to guarantee free traffic flow, that is, not to block possible detours by setting up a parallel work zone. Also the blocking of arterial roads and urban freeways should be minimized by maintaining buildings on the same main road at the same time.

The monetary objective is to steady the budget flow over the years. Costs can also be decreased by adjusting the maintenance schedule with the maintenance work of third parties also using the building such as public services and public transport organizations.

Additional constraints exist which are known to the planner but are too complicated to be modeled in the planning software. So the planning tool should give as result a set of solutions that satisfy the given constraints for the planner to choose from.

3 APPROACH

3.1 *Ant Colony Optimization*

As for the described problem there exists no algorithm to find an exact solution in reasonable time it has to be solved approximately, e.g. by metaheuristics. We are using ant colony optimization (ACO) to construct the schedules, as ACO has performed very well at similar problems (e.g. Lee 2009).

ACO is a constructive metaheuristic developed by Dorigo (1992). It is based on the behavior of natural ants that are able to find the shortest way between their nest and a source of food by communicating via a chemical substance called pheromone. Ants are depositing pheromone while walking and prefer routes were more pheromone is present. So this positive feedback guides the majority of ants to use only one path after some time (compare e.g. Dorigo & Stützle 2004; Boysen).

The first problem on which ACO was applied was the Traveling Salesman Problem (TSP, Dorigo 1992). Meanwhile ACO has been applied to many other problems, e.g. the bi-criteria TSP (García-Martínez et al. 2004), the Vehicle Routing Problem (Barán & Schaerer 2003; Reimann et al. 2004), the Quadratic Assignment Problem (Stützle & Dorigo 1999), Timetabling Problems (Socha et al. 2002), Construction Site Planning (Ning et al. 2010), and Scheduling Problems (Merkle et al. 2002; Solimanpur et al. 2005; Christodoulou 2010). For an overview see Dorigo & Stützle 2004.

The basic design of the algorithm is shown in Figure 1.

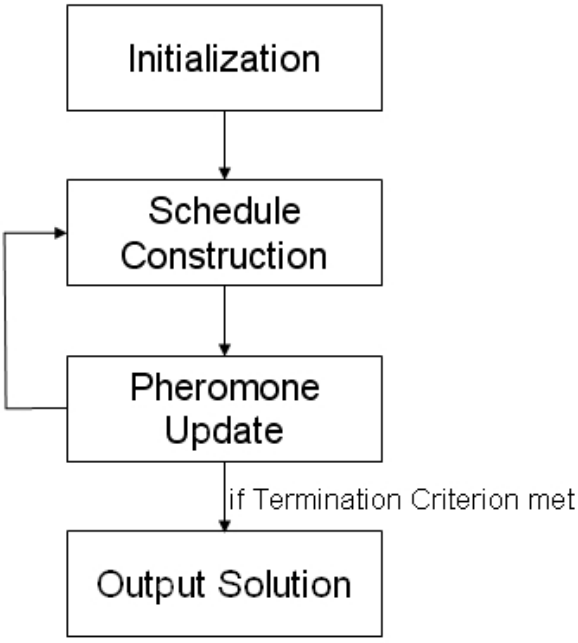

Figure 1. Algorithm

3.2 *Objective function*

Our goal is to minimize the impact on traffic. This can be described by the waiting time produced by establishing a set of working zones that will lead to a closure of some roads.

The waiting time is measured by comparing the travel times in an undisturbed net at rush hour to the scenarios set by the maintenance schedule. For each schedule containing *n* years, *n* scenarios (each with a number of roads closed at the same time) have to be considered. As evaluation value we choose the highest waiting time produced in one of the *n* scenarios of an *n*-year schedule. Another possibility could be to minimize the sum of produced waiting time over all *n* years.

3.3 *Problem model*

To solve the maintenance scheduling problem by ACO we are structuring it as a graph (see Figure 2). To construct a schedule the ants are moving through the graph from left to right. The vertical layers of the graph represent the years. In each year the ants can chose from a number of buildings that have to be maintained, that is the rows of the graph.

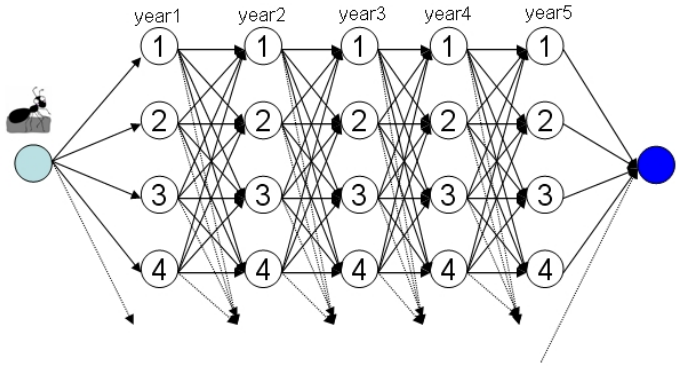

Figure 2. Problem structure

To model the parallel maintenance of several buildings we are not using single ants as in classical ACO but ant teams as proposed by Lee (2009). Those ant teams are sets of ants that together construct a schedule. The choices made by the ants of one team are known by all ants of this team but not by the ants of the other teams. E.g. if building no. 1 has been chosen by one ant of team 1 in the first year, this building no longer has to be considered by the other ants of team 1 in year 1 nor by all ants from this team in the following years but it is still available for the ants of the other teams as well in year 1 and – if not chosen then – in the following years.

Following Dorigo & Stützle (2004) we create as many ant teams as there are buildings in our problem. This helps for a good exploration of the search space.

3.4 *Initialization*

To ensure good performance of the algorithm the pheromone trails have to be initialized to a start value τ_0. This prevents the ants from being influenced to much by the solutions found in the first iteration.

Dorigo & Stützle (2004) propose to set $\tau_0 = m/C^{nn}$ where m is the number of ants (respectively ant teams) and C^{nn} is the cost of a solution constructed randomly or by any other heuristic.

3.5 *Construction of a schedule*

For constructing a good schedule it is of no importance in which sequence the buildings are maintained. Only the year of maintenance of each building and the set of buildings chosen in each single year is of interest. So other than in traditional ACO the pheromone is not deposited on the graph's arches but on the nodes.

An ant standing in column (i-1) chooses the next node j in the next column i with the probability

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} \qquad (1)$$

where τ_{ij} is the amount of pheromone on the node j in column i, η_{ij} is some heuristic information on this node and α and β are parameters for tuning the respective influence of pheromone and heuristic information.

In traditional ACO the heuristic information η_{ij} is obtained by estimating the benefit of the choice of the node j to the objective function. E.g. in the Traveling Salesman Problem η_{ij} is mostly chosen as the inverse of the costs of the arch i-j.

In our maintenance problem we can not estimate the effect of a choice on the objective function as this is a highly complex function depending on all choices made for one year. I.e. the closure only of road A may not have a great impact on traffic flow but if road A is the best detour for road B the impact may increase enormously when there is maintenance done on both roads A and B at the same time.

But we can make use of η to steer the ants away from infeasible solutions: To describe the condition of a building we are using condition indices from 1 to 6 where 1 is the condition of a newly built building and 6 the condition of a building with a high risk of collapse (see also Schießl & Mayer 2006, 2007). This condition index can be used to set η. Buildings with a high condition index should be more attractive for the ants than buildings in a good condition. Thereby η and the amount of pheromone deposited at each iteration τ should be in the same dimension. This biases the ants in the first few iterations away from infeasible solutions with buildings reaching condition index 6 in the period under consideration and so violating the security constraints.

Alternatively one could set deadlines for the latest time for maintenance for all the buildings and construct η depending from the number of years until this deadline is reached.

A first implementation has shown that for different objectives for τ and η the weighting β should be very low to ensure convergence. Convergence here means that after a number of iterations the majority of ant teams will choose the same near-optimal schedule. While for a TSP Dorigo & Stützle (2004) propose $2 \leq \beta \leq 5$ in our implementation the algorithm shows best convergence – and also best results in respect to the objective function – for $\beta = 1$. α as in TSP should be set to 1.

The algorithm introduced here can either be applied to all buildings under one administration or only to buildings that are selected before because they are already in a bad condition. In this second case buildings once chosen need no longer be considered for the rest of the schedule construction; so it is sufficient for each ant team to have a memory where all buildings already in the schedule are stored. The buildings in this memory are not longer available for this ant team for the rest of the schedule.

In the case where all buildings are considered it may be desirable to do maintenance work on the

same building more than once during the period under consideration. Then once maintained buildings have to be still available for choice but their condition index respectively their maintenance deadline and therefore their η has been changed. This change in condition only exists for the one ant team that has scheduled the maintenance of the building, though. So for each team there has to be maintained its own η-matrix that changes during schedule construction and has to be reset after each iteration.

3.6 *Pheromone update*

After all ant teams have constructed full schedules, i.e. each ant of each team has chosen a building for each year of the period under consideration, the pheromone traces on the graph are updated.

First some pheromone evaporates on all nodes to enable the algorithm to forget bad solutions. This is done by decreasing the amount of pheromone by a factor ρ with $0 < \rho \leq 1$. Following Dorigo & Stützle (2004) we choose $\rho = 0.5$.

After this all ant teams deposit pheromone on all the nodes belonging to their respective schedules. The amount of pheromone $\Delta\tau_{ij}^{k}$ deposited by each team k is computed as the inverse square of the highest waiting time produced over the period under consideration in the respective schedule for each node ij that belongs to this schedule. It is 0 for all nodes that are not part of the schedule k. Also ant teams that constructed infeasible solutions, e.g. solutions that violate the security or the budget constraints, are allowed to deposit pheromone to prevent premature convergence and to exploit the information contained in those solutions.

To avoid convergence to such infeasible solutions we are using the Elitist Ant (EA) approach (see also Dorigo & Stützle 2004). In the so called elitist ant the best solution found so far is stored. As long as no better solution is found this elitist ant in each iteration deposits pheromone even if its solution is not part of the set of solutions found in this iteration. The influence of this elitist ant can be weighted by a factor e. So the overall pheromone update is computed as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k} \Delta\tau_{ij}^{k} + e\,\tau_{ij}^{EA} \qquad (2)$$

Dorigo & Stützle propose e to be the same as the number of nodes (buildings). As we have teams of ants, e better is chosen same as the number of ant teams.

Originally EA has been developed to guide the ants away from the greedy solutions, i.e. the solutions with to high influence of the heuristic information η, of the first few iterations. As in our approach η and τ describe different optimization goals the danger of such greedy solutions is not given. By not allowing infeasible solutions to become the solution of the elitist ant the EA approach is a useful tool to get rid of those solutions while still exploiting possible useful information contained in them.

3.7 *Next iteration and termination*

After the pheromone update the next iteration step starts. The memories of the ant teams respectively the changes to the η-matrix are reset. The construction of the schedules now orients on the new values of the pheromone trails.

The algorithm terminates when over a given number of iterations the iteration-best solution does not change.

4 EXPERIMENTAL RESULTS

The algorithm described before has been implemented for a simplified model of the problem to test its performance.

On the one hand simplifications have been made in the deterioration function: For each building was set the same deterioration function where the condition index is only dependent on the age of the building. Maintenance on a building resets its condition index to 1.

Further there is no coupling with a traffic simulator, yet. To model the impact on traffic to each building fictitious values of "waiting time" produced by a work zone at this building were assigned. The "waiting time" per building is set to a value between 100,000 and 400,000 units. At a parallel maintenance of some buildings their respective "waiting time" values are summed up linearly. This of course is far from reality but works well as a simple surrogate model.

For the first test only security and budget constraints were considered. Schedules not fulfilling those constraints are set infeasible.

The test case is a set of 100 buildings with randomly assigned ages. Per year 5 buildings can be maintained, so the worst possible schedule produces a highest waiting time of 2,000,000 units. The period under consideration is 5 years. Without maintenance after the five years 3 buildings reach condition index 6 (i.e. are likely to collapse).

First tests show that the algorithm converges very fast – in about 20 iterations – to a feasible solution. The solutions found produce a highest waiting time of 900,000 to 1,000,000 units. This is far better than a randomly constructed schedule that produces in average a highest waiting time of 1,500,000 units.

5 CONCLUSIONS

Maintenance scheduling is an optimization problem subject to many constraints. With the algorithm

shown in this paper near optimal solutions can be found.

The algorithm is based on Ant Colony Optimization. Modifications have been made to model parallel maintenance work and to guide the search in the first iteration step without knowing the benefit of certain choices to the objective function.

A first implementation could show that the algorithm is able to find good solutions.

A next step will be the coupling with a traffic simulator to get realistic values for the produced waiting time.

Future studies could further improve the performance of the algorithm by testing different settings of the ACO-parameters α, β and ρ. Further adaptations, should make the algorithm fit even better to the needs of building managers.

REFERENCES

Barán, B. & Schaerer, M. 2003. A multiobjective ant colony system for vehicle routing problem with time windows, *Proceedings of the 21st IASTED International Conference Applied Informatics*: 97-102.

Boysen, N. no year. *Ameisenalgorithmen*, Hamburg: http://www.ameisenalgorithmus.de/downloads/ameisenalgorithmen.pdf.

Christodoulou, S. 2010. Scheduling Resource-Constrained projects with Ant Colony Optimization Artificial Agents, *Journal of Computing in Civil Engineering* January/February 2010: 45-55

Dorigo, M. 1992. *Optimization, learning and natural algorithms*. Milan: Politecnico di Milano.

Dorigo, M. & Stützle, T. 2004. *Ant Colony Optimization*. Cambridge: The MIT Press.

García-Martínez, C., Cordón, O. & Herrera, F. 2004, An empirical analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP, *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence*: 61-72.

Lee, H.-Y. 2009. Optimizing schedule for improving the traffic impact of work zone on roads, *Automation in Construction* 18: 1034-1044.

Merkle, D., Middendorf, M. & Schmeck, H. 2002. Ant Colony Optimization for Resource-Constrained Project Scheduling, *IEEE Trans. Evolut. Comput*. 6, 4: 333–346.

Ning, X., Lam, K.-C. & Lam, M. C.-K. 2010, Dynamic construction site layout planning using max-min ant system, *Automation in Construction* 19: 55-65.

Reimann, M., Doerner, K. & Hartl, R. F. 2004, D-Ants: Savings Based Ants divide and conquer the vehicle routing problem, *Computers & Operations Research* 31: 563 – 591.

Schießl, P. & Mayer, T. 2006. Lebensdauermanagement von Stahlbetonbauwerken, *Statusseminar zum Verbundforschungsvorhaben „Nachhaltig Bauen mit Beton“*: 65-74.

Schießl, P. & Mayer, T. 2007. *Forschungsbericht 31006/05, DAfStB-Verbundforschungsvorhaben „Nachhaltig Bauen mit Beton“, Teilprojekt A3 „Wesentliche Bausteine eines Bauwerksmanagementsystems“*. Berlin: Deutscher Ausschuss für Stahlbetonbau (DAfStb), Heft 572.

Socha, K., Knowles, J., Samples, M. 2002. A MAX-MIN Ant System for the University Course Timetabling Problem, *Proceedings of the 3rd International Workshop on Ant Colony Optimization and Swarm Intelligence*: 1-13.

Solimanpur, M., Vrat, P. & Shankar, R. 2005. An ant algorithm fort he single row layout problem in flexible manufacturing systems, *Computers and Operations Research* 32: 583 – 598.

Stützle, T. & Dorigo, M. 1999. ACO algorithms for the Quadratic Assignment Problem, *New Ideas in Optimization*, 33-50.