Utilising an Ant System for a Competitive Real-Life Planning Scenario

Christopher Blöcker[1]
Research and Development
implico GmbH
Hamburg, Germany
Email: christopher.bloecker@implico.com

Sebastian Iwanowski
Department of Computer Science
FH Wedel, University of Applied Sciences
Wedel, Germany
Email: iw@fh-wedel.de

***Abstract*—This paper describes the design of an ant system for a dynamic tour planning scenario for oil and gas delivery. The software has been integrated into an existing planning system and achieved satisfying results in the simulation of real-life scenarios considering spontaneous non-predictable changes of tasks. The notion of such dynamics is more general than in previous approaches. The response time and other complexity measures match the needs of real practice. While other papers already exist describing the functionality and advantages of ant systems and giving some case studies, this paper is the first one referring to an integration into a standard operational SAP system. Thus, this paper shows how to bridge the gap between innovative scientific research and industrial application.**

Keywords-ant system; dynamic tour planning; vehicle routing problem with time windows; greedy strategy; SAP system

I. MOTIVATION

Software for tour planning solutions often suffers from the problem that typical real-life applications continuously violate the original input specifications due to changes of existing tasks, generation of new tasks, malfunction of operational units, and traffic congestion in the underlying route system. All these dynamic events may occur while the software is executing its current task.

Tour planning is an NP complete problem, which makes it infeasible to design an efficient solution satisfying all theoretical needs. Real-life logistics requires a solution for the even more complex vehicle routing problem VRP (cf. [1], [2]) dealing with several vehicles to serve delivery orders meeting pre-defined time windows.

Despite these theoretical obstacles, reasonable heuristics for tour planning already exist achieving remarkable run time results for problem sizes occurring in real problems. Besides classical OR techniques (e.g., cf. [3]), some promising heuristics also use innovative artificial intelligence techniques such as neural networks, genetic algorithms, and ant algorithms (cf. [4], [5], [6], [7]).

However, the benchmarks normally used in the scientific community (cf. [8], [9]) in order to evaluate the different heuristics do not consider problems of the type described above i.e., problems where the input is subject to continuous and unpredictable changes even during execution of the software.

Some of the ant papers cited above (e.g., [6]) do work with dynamic changes explicitly, which is not surprising because ant algorithms are specially suited for that situation as we will also elaborate in the following. But none of them referred to the exact planning tasks we wanted to deal with, which are described in Section II.

The task of this work is an implementation of a tour planning system coping with dynamic changes. The software has to be integrated smoothly into the IDM (Integrated Dispatch Management), which is an implico framework for tour planning of oil and gas delivery linked to an SAP system. Typical scenarios of past applications of IDM serve as evaluation benchmarks.

This paper is organised as follows: Section II presents the problem to be solved in practice. Section III describes the software architecture of the operational system, in which the new solution had to be integrated, and the functionality of the modules existing before. Section IV gives a short description of principles and advantages of ant systems in general. Section V shows how we adapted these general principles for the actual problem. Section VI gives some test results and interpretations. The conclusion in Section VII compares with other approaches.

II. THE ACTUAL PLANNING TASK

Our actual planning problem is the scheduling and routing of oil and gas delivery to a set of customers: The customers specify the requested product and a time window, in which they want to receive the delivery. The possible transportation units are heterogeneous trucks, which are initially located at several truck depots. The products are located at several supply depots differing in availability and price. Not all trucks are eligible to transport all kind of products or fit further needs of all customers.

Among the frequent dynamic events that we want to take special care of are the failure of a vehicle, delays in the delivery procedure, incoming of new orders with high priority, and traffic congestion during the tour. Our target is to provide a substitute schedule shortly after a dynamic event occurs and to minimize the additional costs.

[1]This work was done while the author was working on his Bachelor's degree at FH Wedel.

 ISBN: 978-1-61208-222-6

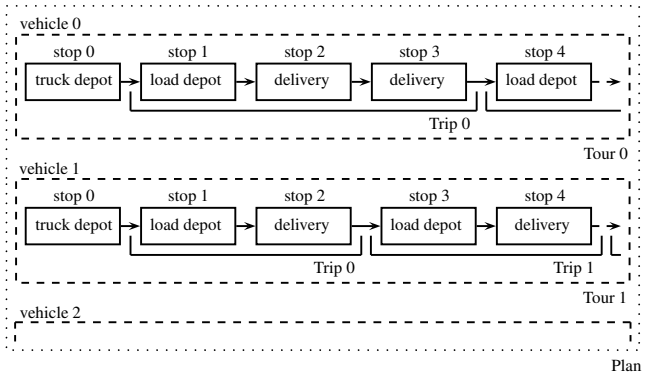

Figure 1. The structure of a delivery plan built from tours, trips and stops.

A schedule in our context consists of a set of several delivery plans each describing the deliveries to be performed within one day in the given planning period. A single delivery plan is composed of a set of several delivery tours, one tour per vehicle. Due to common practice in oil and gas delivery, every tour is defined to begin and end at the corresponding truck's home depot. A tour contains several subsequently ordered trips. Each trip is a sequence of stops. The first stop of a trip is a supply depot followed by several delivery stops. When a product for a subsequent customer is finished, the truck has to reload new supply at a depot starting a new trip. An extract of a sample plan is shown in Fig. 1.

Each customer may reserve time windows, in which he wants to be delivered. Additional attention has to be paid to the reservation of recreation breaks for the drivers prescribed by law. Furthermore, not all different products are eligible to be shipped together on the same trip due to possible chemical reactions or even explosions.

In general, we assume that there is no need to split delivery charges because for each single delivery charge there will be at least one vehicle providing sufficient transport capacity. If this is not the case, we require that the delivery charge is decomposed in appropriate sizes prior to consideration within our planning procedure.

III. Software Architecture of the Operational System

The underlying business framework IDM provides all functionalities a human dispatcher needs. For example, it visualizes current orders to be fulfilled, the current delivery plan on a map and gives easy opportunities for manual integration of new stops and rearranging the current plan. It is based on an SAP system [10], which handles the entire delivery order process. In addition to this base functionality, IDM provides an interface for the integration of various tour optimizers, which may even be operated simultaneously and should ideally make a manual interaction unnecessary. But since the dispatchers are often overcharged with the frequent dynamic changes of the situation in practice, they should at least be supported by an automatic tour optimizer being able to handle specially the dynamic case in a very short response time.

Currently, IDM involves three different optimizers in the context of delivery scheduling (including our ant system), each addressing a different part of the problem.

The first of them, TermiDe, is used to determine whether an incoming request can be served considering the time window and other constraints given by the customer. This is typically done by a phone order several days prior to the actual delivery. If a delivery can be granted, the order is put into a pre-schedule for tour planning, which is the starting point for subsequent optimizations. Since TermiDe is used for telephone sale, it must provide its decision a few seconds after the request. This leads to a feasible but not very good solution.

The second optimizer, IcedG, uses a metaheuristic approach based on tabu search [11], [12] in order to approximate solutions for the static VRP. It is run daily with the purpose to precompute the delivery plan for the following day based on the results of TermiDe. IcedG faces no competitive restrictions regarding the response time, but the quality of the result is required to be very high because it has direct influence on the operational costs of the schedule. Typically, IcedG computes almost the optimal result, but it may need several hours for computation.

The tour optimizer, which is subject of this paper, is called Dyonisys, which is short for **Dy**namic **o**ptimization using a **n**ature-**i**nspired **sys**tem. This optimizer is running during the execution of a delivery plan, i.e., during the whole day. In the morning it gets its input from the schedule, which was previously computed by IcedG. Whenever a notable event occurs, Dyonisys tries to adapt the schedule to the new situation. In fact, Dyonisys can also be configured such that it may further improve the current plan while there is no other work to do.

Most likely Dyonisys will not be able to compute as good results as IcedG would do. The advantage of Dynisys is the following: While IcedG only solves the static VRP and needs several hours to compute an approximate solution, Dyonisys is capable of solving the dynamic VRP and to reduce the response time to a few minutes.

At any time, IDM may ask Dyonisys for the next stop for a particular vehicle and assign a task to it according to the current schedule. Dyonisys then considers the stop the vehicle is currently heading to as granted and estimates the time point when the vehicle will be available for further deliveries when the next delivery has been rolled out.

If the situation happens that a certain delivery cannot be performed at all according to the current schedule due to an unexpected event, this is detected by IDM, which informs Dyonisys. Then, Dyonisys would reply with a substitute schedule, as soon as required. This can be guaranteed,

 ISBN: 978-1-61208-222-6

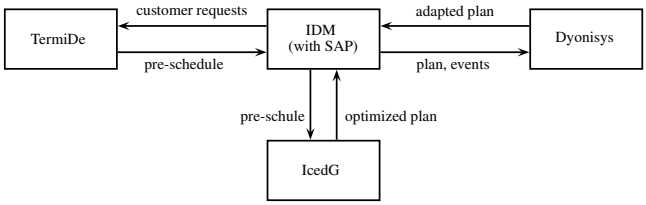

Figure 2. Communication between IDM and its optimizers TermiDe, IcedG and Dyonisys.

because ant systems are anytime algorithms which always can provide a solution. Of course, the quality of the solution depends on the allowed response time.

The communication between the different modules described above is given in Fig. 2.

IV. General Principles of Ant Systems and their Advantages

Ant systems are used to solve optimization problems on graphs. The current optimization problem is defined by a target function depending on edge costs and possibly further information about the graph. Ant systems are specially designed for the scenario that edge costs and the target function vary during operation. They resemble the behavior of natural ants when they seek for food.

Being living animals with a nondeterministic behaviour, ants differ in their strategy from classical optimization methods in the way that there are always some individuals searching a solution on an obviously non-optimal path. An ant colony consists of a large number of individuals, who communicate via pheromones, which are chemical substances dropped on the paths. The intensity of the pheromones biases the behaviour of ants, which leads to a nearly optimal solution for the majority of the individuals. However, the nondeterministic behaviour of the minority enables such a system to react rather quickly to dynamic changes of the environment. This is the main advantage of ant systems.

In the following, we describe the general idea of artificial ants and ant systems.

An (artificial) ant is a software unit, which is continuously generated over time by the ant system. Each ant uses the current data of the graph, considers the current constraints to be solved at the time of its generation, and tries to find a single solution for this problem. The quality of the result influences the modification of pheromones, which are dynamic information chunks placed onto the edges of the graph. The pheromones represent the collected memory of previous ants using the respective edge. Subsequently generated ants are biased by these pheromones for their own construction of a solution.

In general, ant systems use complete graphs for tour planning problems since this will always enable them to complete partial solutions. For our problem, this assumption is reasonable because in practice it is always possible to find a route from one location to any other.

The probability of selecting an edge for tour completion depends on the quality of pheromones put so far as well as on some heuristic value, which is usually derived from the graph's cost function. Usually, this heuristic value is static, but for ant systems even that need not be. The trade-off between the dynamic pheromones and this heuristic value may vary depending on the stage of the process or on the application in general. The continuous generation of ants by the system guarantees that the pheromone value is successively updated to the latest situation in an eager way i.e., prior to a possible request from a user to the ant system. This guarantees a quick response time for any request. However, after the occurrence of a new event, the longer the ant system is running, the better do the pheromones reflect the current situation.

The construction of solutions is carried out in different phases.

First, in the initiation phase, initial pheromones are distributed to all edges in the graph. Normally, all edges get the same pheromone value, but at initialization we could make that also dependent on the cost function [4].

Then, in a loop, construction and coordination phase are executed in turn as long as the ant system is needed. In the following, we denote a construction step followed by a coordination step as one iteration.

In the construction phase, a certain number of ants is generated simultaneously. Each ant has to find a solution. Applied to our VRP, the task of a single ant is to construct an assignment of vehicles to the stops in a certain order such that the tour is feasible for each vehicle, each station is served in the requested time window, and there is always sufficient product supply. At each stop, which is reached by an ant during the construction of a tour, the probability that this ant will use a certain edge leaving this stop is directly proportional to the amount of the pheromone value. Thus, more ants will use the edges baring good pheromone values.

When all ants that were generated in the construction phase have constructed a solution, the construction phase is finished and the coordination phase starts, in which all pheromone values are updated: First, all pheromones are decreased resembling evaporation. This makes the future results more decisive than the past ones. After evaporation all ants increase the pheromones on the edges they actually used for their specific solution. The increase of the pheromones is inversely proportional to the real costs of the associated solution. In total, this makes pheromones of edges belonging to favorable tours increasing and of disfavorable tours decreasing.

Note that the alternating phases of construction and coordination correspond to a discrete simulation of a continuous process which was first described in [4].

 ISBN: 978-1-61208-222-6

If a current solution is requested, the system returns the solution obtained from the current pheromones. This enables an ant system to give a quick answer with a solution corresponding to the current status of the dynamic system and makes it suit well for the problem we have addressed above.

For further information about ant systems, we refer to the standard literature such as [4], [5], [13], [14].

V. How Dyonisys Works in Detail

A request to the ant system is generated every time a vehicle in operation needs the information about the next target. Dyonisys then looks up which delivery is scheduled next, passes this information to IDM and removes it from the set of deliveries that are left to be performed. We have decided to take this approach because it is much more practical to inform a driver about his next target once he needs to know this than to inform the driver continuously about the best schedule found so far.

At the beginning of a day, Dyonisys starts with the schedule produced by IcedG. As long as no dynamic event occurs there is no need in running the ant optimizer because the IcedG schedule achieved near optimality. Once an event occurs, Dyonisys catches up with the current state of the real world situation i.e., it modifies the graph's cost function or adds / removes vehicles or deliveries.

Dyonisys uses the following representation of the scenario: For every delivery, load depot, and truck depot, the underlying graph maintains a separate node. Since IDM already provides a distance matrix between the locations involved, Dyonisys need not perform the actual navigation on street level.

According to the general principle described in the previous section, each ant to be generated will try to find a solution for the total planning task, which is currently in consideration. Each solution is evaluated according to a quality function considering the overall time and the consumption of resources. The solution is also compared with the constraints currently valid. If a constraint is violated, a penalty function is applied decreasing the evaluated quality of the current solution. There are several penalties depending on the different types of constraint violations.

In the initialization phase, Dyonisys creates the ants, which will be used for solution construction and sets the initial pheromones τ_0 to all edges (i, j) from nodes i to j (this is the way it is explained in [4]). For performance reasons, we prefer to reuse the ants rather than to dispose them and create new ones every iteration.

Right after that, the loop of construction and coordination phase is started, and each ant produces a schedule. We chose a greedy strategy for this step: An ant picks one of the vehicles and creates a complete tour before proceeding with the next one until all deliveries are assigned to a tour or there is no capacity left among the vehicles. The benefits of this technique are a good response time and quite satisfactory results.

Note that there always exists the trivial solution where no delivery is carried out. So we will always have an initial solution. However, in nearly all cases we will get a better one, because this trivial solution is associated with very high penalties.

As mentioned before, the ants successively construct a tour starting at the selected vehicle's home depot. At any point, they decide which node of the graph (i.e., which delivery stop respectively, which supply depot) to visit next depending on a value derived from the pheromone level and heuristic value of the incident edges.

If an ant is located at node i, then the probability of visiting node j next is obtained by evaluating term (1), where $\tau_{i,j}$ denotes the pheromone level on the edge (i, j), $\eta_{i,j}$ the heuristic value of the same edge and α, respectively β, are weighting parameters to control the contribution of pheromones and heuristic. $N(i)$ is the set of unvisited neighbours of node i (cf. [4], [5]).

$$p_j = \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum_{j \in N(i)} \tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}} \tag{1}$$

For reasons of efficiency, we only evaluate the numerator of (1) and accumulate the values to obtain the denominator. Instead of normalizing the probabilistic values according to (1) and generating a random value $r \in [0, 1]$, we introduce the new approach to let them unchanged and take a random value $r' \in [0, \sum_j p_j]$. Choosing this implementation we were able to save a lot of runtime without altering the result. Note that if α were set to 0, the ant system would act in a completely deterministic way, and the pheromone values would be ignored.

After that, the coordination phase is started:

First, evaporation takes place. A fraction ρ specifies the amount, by which a pheromone value should evaporate, cf. (2).

$$\tau_{i,j} \leftarrow (1 - \rho)\, \tau_{i,j} \tag{2}$$

Then, each ant increases the pheromone value on the edges it used for constructing its individual solution. According to Dorigo et al. [4], if the cost of a solution of ant a is c_a, then the pheromone increase is $\frac{1}{c_a}$. Dorigo et al. also suggest to apply additional methods such as allowing only the best ants to increase pheromone values at all or to add so-called elite ants, which reinforce the best solutions found so far (cf. [15]).

Since our solution had to be effective also for huge networks which may occur in practice, we had to adapt the pheromone update a little more sophisticated than previous papers suggest:

 ISBN: 978-1-61208-222-6

Since some overall cost values c of our scenarios were rather high, the inverse $\frac{1}{c}$ (which should be the pheromone update value) came close to 0. In these cases, the evaporation rate of pheromone values would outnumber the increase rate of pheromone values, which eventually would lead to a pheromone value nearly 0. Subsequent ants would then use the (static) heuristic function only, but not the dynamic pheromone values, which makes our system less prone for dynamic changes. This is why we use the update value $\frac{c_{best}}{c_a}$ instead of $\frac{1}{c_a}$, where c_{best} denotes the value of the best solution found so far and c_a the costs of the solution computed by and a. Then the update value for ants having found a rather good solution is close to 1 (or even higher if the ant found a better solution than before). This would make the respective edge favourable for subsequent ants. Only when an ant finds a very bad solution, this would still decrease the update value as it did before our modification, but a frequent occurrence of such solutions would indicate that the link really deteriorated. This makes our system reacting on dynamic changes just as desired.

We summarise the algorithmic improvements to the state-of-the-art described in Section IV:

1) The probabilistic values of (1) are not normalized, which saves a lot of time without altering the result.
2) Our evaporation update of the pheromones is done in such a way that pheromone changes are also realised in huge networks.

Considering the operational system, special attention should be taken to handle dynamic events such that the proper execution of the ant system is not too much disturbed by the unexpected input of a new event. For this sake, we make a distinction between events changing the structure of the graph and events leaving the graph as it is.

Since we are only dealing with complete graphs, a cost function change leaves its structure always unmodified. Thus, an information about a traffic congestion would simply result in an update of our cost function and let the ants continue their work. The same argument holds for the opposite case, i.e., when edge costs are reduced.

However, adding or deleting a node would be a change of the graph structure. Such changes need a more sophisticated treatment: The ant system has to be halted, the changes have to be applied, and then the ant system may resume its work.

The following considers the tasks of such structure changing events in more detail:

New customer orders must be added to the set of deliveries and marked as unplanned. Simultaneously, a new node is added to the graph, connected to all other nodes, and pheromones are put onto the new edges. Feasible values for the new pheromones are the initial values for starting graphs or the average of pheromone values computed so far.

A vehicle break down results in the deletion of a node and its adjacent edges. But this requires a scan of all ants and the removal of this vehicle from all partial solutions. We implemented a proper data structure and method such that this is still superior than deleting all ants and starting from the scratch.

In detail, this is achieved by the following:

Note that we only need to consider ants that have already finished constructing a solution or - at least - that have already started constructing a solution. All other ants will be supplied with the new situation before they start and, thus, need not be bothered by the fact that in previous times we had a different situation.

Our data structure allows an easy access to the tour belonging to the removed vehicle, which is possible in constant time because there is a direct correspondance between a vehicle and the tour it is used for.

This is why we wait until all ants having started, before removal was announced, have finished their construction. Then we delete the tour corresponding to the removed vehicle and mark all deliveries from the deleted tour as unplanned, which automatically leads to higher penalties in the evaluation of the corresponding solution. Only then we start the ants for the new situation. This can be expected to happen very fast due to our greedy strategy for tour construction.

VI. Test Results

We tested Dyonisys using typical real world scenarios scanned by IDM in the past. We chose a standard configuration of 100 iterations, an evaporation rate of $\rho = 0.1$, initial pheromones of $\tau_0 = 25$, $\alpha = 1$ and $\beta = 5$ and ran our tests on an Intel Core-i5 M560 with 8GB RAM. In our tests we used different values for the parameters and studied their effect on the runtime of the ant system and the quality of the solutions.

Not surprisingly, we found that the runtime is directly proportional to the amount of ants used for optimization, cf. Table I and Fig. 3.

Interestingly, we did not get a corresponding result for the quality of the solutions. In general, using more ants enlarges the chance of finding a better solution, but the more ants are used, the smaller is the benefit of using an additional ant.

We observed a similar result with respect to the number of iterations: The solutions also get better as more iterations are performed, cf. Table II and Fig. 4. But, if we have already applied a high number of iterations, the benefit of an additional iteration is rather low.

As expected, if the number of iterations is kept constant, Dyonisys is able to find better solutions using more ants and vice versa. Thus, a higher amount of ants used and a higher number of iterations performed have a similar quality enforcing effect.

Summarizing, we conclude that it is generally more practical to perform a higher number of iterations keeping the number of ants not too high. How many iterations and ants should exactly be used depends on the actual scenario.

 ISBN: 978-1-61208-222-6

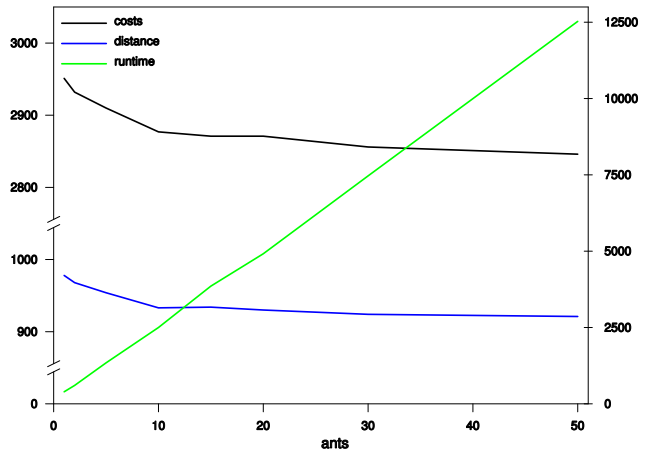

Figure 3. Progression of runtime, total costs and total distance of a schedule depending on the count of ants used for optimization.

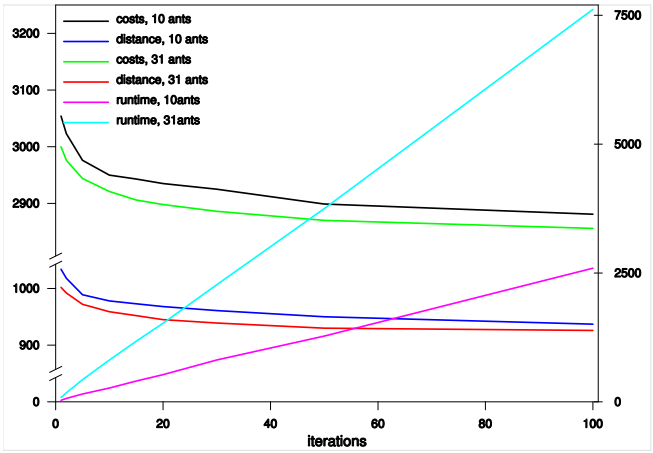

Figure 4. Progression of runtime, total costs and total distance of a schedule depending on the iterations performed by the ant system.

ants	costs	distance	time
1	2.951	978	395
2	2.932	968	604
5	2.910	954	1.343
10	2.877	933	2.501
15	2.871	934	3.856
20	2.871	930	4.918
30	2.856	924	7.468
50	2.846	921	12.529

Table I

	10 ants			31 ants		
iterations	costs	distance	time	costs	distance	time
1	3.054	1.034	31	3.000	1.002	85
2	3.023	1.018	67	2.976	992	177
5	2.976	989	152	2.944	972	429
10	2.950	978	268	2.921	959	810
15	2.943	973	404	2.906	952	1.176
20	2.935	968	527	2.898	945	1.526
30	2.925	961	812	2.886	939	2.274
50	2.899	950	1.276	2.870	930	3.749
100	2.881	937	2.592	2.856	926	7.614

Table II

In most cases, we were able to achieve satisfying results using only 10 ants. In general, there is a trade-off between runtime and quality.

The complete results dependent on the variation of several parameters are elaborated in [16] (in German).

VII. Conclusion

We developed an ant system solving the VRP with time windows for a special application scenario in practice. The ant system was integrated into an operational SAP software environment. The ant heuristics used were simple using a greedy strategy, which resulted in a system with reasonable run time and space consumption. Our intensive testing revealed, which parameters to adjust in order to obtain qualitatively best results in short time. We could thus obtain a setting that fulfils all functional needs.

The improvements to the state-of-the-art are the following:

1) We proposed some algorithmic improvements concerning runtime and applicability for huge networks (cf. Section V). The feasibility is shown in Section VI in examples and in [16] in detail.
2) We took special care for some implementation specific details important to improve the usability in practice. One example is the distinction between different types of events (cf. Section V).
3) Our notion of dynamics is more general than that of other papers. For example, compared to [6], we admit the removal of vehicles and the removal of deliveries from a certain tour.
4) A unique novelty of this paper is the combination with other (standard) techniques for the VRP problem in an integrated software environment (SAP), which makes the novel technique of ant systems ready to be sold in a software product. Previous papers dealing with practical applications showed stand-alone field studies and were not integrated into a software product.

A promising target of improvement would be a replacement of the greedy strategy by a tabu search. Besides this conceptional improvement our focus will be the further product development fulfilling all needs of our customers.

The general message of this work is:

Operational logistics systems baring dynamic behaviour profit from ant technology.

A question that may arise to the reader is: Would this result also hold for other innovative approaches of soft computing?

Our previous experience showed that theoretically successful approaches cannot always be adapted straight forward as this happened to our ant approach:

Before we applied the ant approach we tried to solve our

problem with a neural network approach. In particular, we tried to apply self organizing maps (cf. [17]). In a quick implementation we were able to achieve remarkable results on the standard set of benchmarks for the traveling salesman problem (cf. [18]), which were even better than our results with ant systems.

But, we faced two main problems when we attempted to design a neural network approach to solve our vehicle routing problem as stated above.

First, we found no adequate way of readjusting the learning parameters in case of a dynamic event.

Second and even more severe, in our application the triangle inequality does not hold for all triples of nodes in the graph, because we have to obey toll costs in the traffic network. But, the validity of the triangle inequality is an important prerequesite for readjusting the neurons' positions properly.

We do not claim that a neural network approach would fail for our application in general, but at least our attempts revealed that the development of a successful solution would not be that easy as the one presented in this paper using ant systems. Thus, a further value of this work is that it proved the practical applicability of ant systems in a real world setting which is not self understood as indicated with our search for alternatives. This makes the future development of ant systems for logistics applications more attractive even if other approaches may prove superior in closed world experiments.

REFERENCES

[1] A. Larsen. The Dynamic Vehicle Routing Problem. PhD thesis, University of Denmark, 2000.

[2] P. Toth and D. Vigo. The vehicle routing problem, volume 9. Society for Industrial and Applied Mathematics, 2002.

[3] O. Bräysy and M. Gendreau. *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms*. *Transportation Science*, 39(1):104–118, 2005.

[4] M. Dorigo and T. Stützle. Ant colony optimization. The MIT Press, 2004.

[5] M. Dorigo, M. Birattari, and T. Stützle. *Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique*. *IEEE Comput. Intell. Mag*, 1:28–39, 2006.

[6] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati. Ant Colony System for a Dynamic Vehicle Routing Problem. *J. Comb. Optim*, 10(4):327–343, 2005.

[7] O. Bräysy and M. Gendreau. *Vehicle Routing Problem with Time Windows, Part II: Metaheuristics*. *Transportation Science*, 39(1):119–139, 2005.

[8] M. Gendreau, F. Guertin, J. Potvin, and R. Seguin. *Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries*. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174, June 2006.

[9] A. E. Rizzoli, F. Oliverio, R. Montemanni, and L. M. Gambardella. *Ant Colony Optimisation for vehicle routing problems: from theory to applications*. Technical report, 2004.

[10] SAP online reference. http://www.sap.com/uk/solutions/business-suite/erp/index.epx. [retrieved: 05, 2012].

[11] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[12] P. Toth, and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS J. on Computing*, 15(4):333–346, December 2003.

[13] M. Dorigo and L. M. Gambardella. *Ant Colony System: A cooperative learning approach to the traveling salesman problem*. *IEEE Transactions on Evolutionary Computation*, 1997.

[14] É. D. Taillard. *Ant Systems*. *Kluwer*, 2000:131–144, 1999.

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

[16] C. Blöcker. *Entwurf und Implementierung zur dynamischen Optimierung von Liefertouren mit einem Ameisen-System* (in German). Bachelor's thesis, FH Wedel, 2011. http://www.fh-wedel.de/mitarbeiter/iw/eng/r-d/done/bachelor/ [retrieved: 05, 2012].

[17] T. Kohonen, M. R. Schroeder, and T. S. Huang, editor. Self-Organizing Maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[18] G. Reinelt. *TSPLIB - A Traveling Salesman Problem Library*. *INFORMS Journal on Computing*, 3(4):376–384, 1991.

 ISBN: 978-1-61208-222-6