Tactical Planning using Agents

Roman van der Krogt [a] Leon Aronson [a] Nico Roos [b]
Cees Witteveen [a] Jonne Zut [a]

[a]Delft University of Technology, Faculty of Information Technology and Systems, P.O. Box 5031, 2600 GA Delft
[b]Universiteit Maastricht, Infonomics/ Institute for Knowledge and Agent Technology, P.O. Box 616, 6200 MD Maastricht

Abstract

Modern transportation problems are highly dynamic and time critical. A planning system for transportation problems therefore must include efficient and flexible planning and replanning strategies. In this paper we introduce a general agent-based framework for highly dynamic order-based transportation planning problems where a tactical planner and several tactical planners (one for each transport agent) are distinguished. In particular we discuss the role of the tactical planner responsible for dynamic task allocation of orders and we present some preliminary results of task allocation heuristics in such dynamic environments.

1 Introduction

The transportation problem we will discuss consists of a number of (transportation) *agents* capable of moving cargo units containing freight. Customers are placing transportation *orders*. Some of these orders may be known far in advance, some of them have to be executed almost immediately. Customers are free to retract or modify their orders in advance. Associated with an order o there is a specification of its time windows[1], its profit if it is executed within the time windows and its penalties if it is rejected or the agents fail to execute it within the time windows specified. There is also a static infrastructure, represented by locations, roads between them, global travel times, and distances on the roads. Our problem consists of assigning orders placed by the customers to the transport agents in such a way that the total cost, which is dependent of time, distance and the penalties assigned, is minimised. [2] Even simpler problems than this one are easily shown to be intractable, i.e., they are not optimally solvable in an efficient way. The use of time windows and its dynamic character make this problem even harder to solve. Therefore, any practical solution must be based on clever heuristics, i.e., problem solving methods that produce acceptable solutions but not necessarily the optimal

[1]There is a time window for pickup and a time window for delivery of the order

[2]The problem description can be easily extended to include non-constant penalties, dynamic infrastructure, variable number of transport agents and cargo units, and variable capacities. However, since planning is complicated enough as it is, we have chosen to exclude these aspects for now.

ones. We also use a *distributed approach*, so each agent can use a heuristic suited for its responsibilities.

We model this problem using a hierarchical agent-based approach (see Figure 1). On the highest level, we have a *tactical planning* agent responsible for taking care of incoming orders and allocating them to individual transport agents. On the lower level, we have the *transport agents* taking care for the operational planning and its execution. Such a transportation agent / operational planner may be a truck driver or a computer program running on an AGV. Each operational planner is capable to see only that part of the plan it is allocated to by the tactical planner.

Orders, placed by customers, are received by the tactical planner and contain information about the freight that has to be moved, when and where it should picked up, how much time the loading takes, similar information about the delivery and the penalty that is incurred when the order is not executed (in time). The tactical planner uses this information, together with its knowledge of the infrastructure and the cost function, to compute a plan. Such a plan consists of feasible order assignments to individual agents, i.e., for each transport agent A a sequence of orders that normally A would be able to execute. Each order assignment contains the same type of information as orders, extended with the transport agent and the cargo unit involved. The order assignment is communicated to the corresponding transport agent[3] .

Each operational planner is responsible for executing the orders assigned to it using the infrastructure and handling incidents. It is also responsible for communicating any (expected) deviations from the orders as they are assigned to the tactical planner, such that the tactical planner can adequately adjust plans when necessary. Since operational planners operate on an infrastructure, they may have means to communicate with the infrastructure, e.g. for AGVs to claim parts of a terminal, or for trucks to obtain traffic information (see [1] for more details). In this paper we will concentrate on the details of the tactical planner. We start by giving a detailed look at the tactical planner and possible heuristics. Then we describe the experiments that were performed to test these heuristics and present the results. Finally, we will conclude.

2 The tactical planner

The tactical planner focuses on efficient distribution of the orders. It could be realized using a general-purpose planner and replanner, such as [3] and [6]. However, due to the specific nature of the planning task, and the environment in which this module will be embedded, we have chosen for a special-purpose planning module.

[3]The tactical planner, however, does not have to literally copy the order data into an order assignment. For example, it might restrict the time window of an order such that the next order (assigned to the same agent) can be served on time. It might even create additional orders by itself, for example to move a transport agent without freight to the pick up place of ther next order.

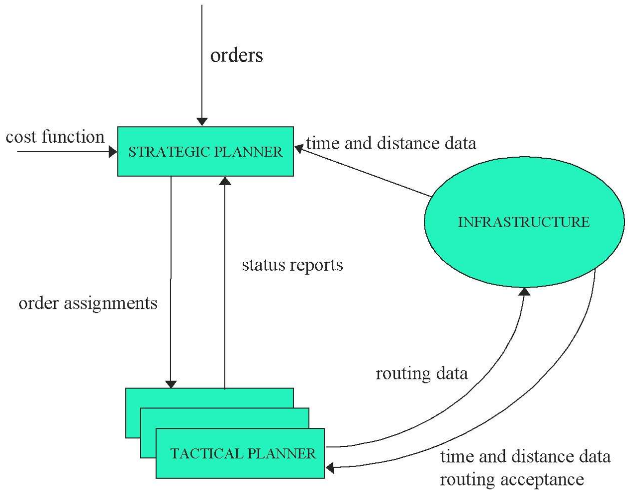

Figure 1: Model layout

We will first describe the internals and the communication with the rest of the system. Hereafter we detail some heuristics used by the tactical planner.

For each transport agent, the tactical planner stores a list of the orders that are currently assigned to that agent. This is a simple list of pickup and delivery locations, along with their associated time windows.[4] The *slack* between two consecutive locations a transport agent has to visit is the amount of time that can be wasted during the travel, while the agent would be still on time at the next location. For example, if the distance between locations A and B is 3 and an agent should visit A at time 1 and B at time 6, the slack would be 2. This slack is an important factor in tactical plans: if the average slack between two actions in a plan is small, the plan is efficient. However, a large average slack makes the plan more robust for disruptions: there is more time for recovering errors. Hence, the amount of slack determines the trade-off between efficiency and robustness in plans.

An order is a tuple $o = (f, s, t_1, d, t_2, l, u, p)$ describing a freight f that is to be picked up at a source location s, within a time window t_1, and to be delivered at destination d within a time window t_2. Also included are the times l to load and u to unload, as is the penalty p for not executing the order o within the given time windows. Note that the penalty assigned to an order can also be conceived as the priority that the particular order has. Upon receipt of an order, the planner first checks the plans of the operational planners to find out if it can insert the order in between two existing assignments. If so, the planner inserts the order into one

[4]Empty rides between different orders are not included in this list. They are however sent to the operational agents.

of the plans that have room for the order. If not, the planner has to shuffle orders in order to try to come up with a schedule in which all orders are included.

The plan generated is communicated to the agents by means of order assignments. Each message tells the agent that it has to move from one location to another, and which cargo-unit and freight are to be transferred (if any). In contrast to the internal representation of the plan, empty rides are included in this list, i.e., whenever the destination location of one order differs from the pickup location of the subsequent order, an additional order assignment is communicated that tells the agent to go from one place to another. This allows for a standard form of communications between the tactical and operational planner.

The tactical planner also receives *order changes*. These originate from either the customer or an operational planner detecting an incident that cannot be resolved within the given timewindows. Changes arriving from customers can either be changes within orders (such as a new pickup time window or delivery location) or complete cancellation of orders. In the latter case, the order is simply removed from the plan. In the former case, we have to examine the implications of the change on the plan.

Order changes that are due to contingencies detected by operational planners are communicated through *status messages*. Such messages originate from a operational planner reporting on its progress whenever it finishes (part of) an order or when it detects incidents it cannot handle on its own. If the agent reports normal progress, these messages contain the precise time at which pickup or delivery took place. The tactical planner uses these status messages to further refine its plans, such that the plans confirm with the situation reported. In case of incidents, these messages contain *(a)* the current position of the agent, *(b)* the type of incident and *(c)* an estimate of the delay the agent expects for his order to be delivered. This information is used by the tactical planner to decide what to do with this order and the other orders assigned to that agent. For example, if an order is delayed such that an agent can still carry out its next assignment in time, the tactical planner might decide to just let the order be delayed instead of trying to reschedule the order to another agent.

2.1 The planning algorithm

The tactical planner is responsible for selecting which agent has to carry out which order. To reach an optimal solution this would require the planner to consider every possible distribution of orders over agents. This would then have to be computed over and over again for each new order arriving in the system, causing an exponential blow-up of computation time. Therefore, we use a heuristic function that determines which agent should carry out an order, by using some simple heuristics. This heuristic order assignment function consists of two parts: agent selection and order redistribution

Agent selection

Each of the individual transport agent order sequences is inspected to look for a place where the new order can be inserted without affecting the existing ordering. If such places exist, this heuristic selects an agent to carry out the new order. Since slack is an important trade-off in plan quality, we can distinguish two selection heuristics based on slack:

1. Maximise slack: if there are a fixed number of transport agents, robust plans can be created by choosing an agent having the largest slack remaining after the current order is included in its plan. If new orders arrive this strategy maximises the probability that an order can be inserted between two existing orders.

2. Minimise slack: if we would like to use as few transport agents as necessary, an agent with minimal slack remaining should be chosen. The less slack, the less time is "wasted" between two orders. This scheme minimises the number of transport agents needed to complete the orders, since orders are only assigned to an agent without existing orders when the other agents cannot carry it out.

Of course, there are multiple other strategies. To compare the slack based heuristics, we will also test with a distance based heuristic, that tries to minimize the travel distance.

Redistribution of orders

In case there is no transport agent to assign an order to using the agent selection heuristics, existing orders have to be rescheduled in order to make room for the new assignment, if possible. The rescheduling heuristic first removes all scheduled orders that are not being executed yet and then schedules them again over the agents. This time, however, the orders will be ordered first, according to heuristics for the well-known *bin-packing problem*.[5] Note that this problem closely resembles the problem of dividing orders over agents, using as few agents as possible, making heuristics for bin-packing suitable to use. With respect to the bin-packing heuristics it is well-known that the order in which the items are examined is important for the quality of the solution. For example, by sorting the items by size in decreasing order, the quality of the solution can be increased from about 70% off the optimal to about 22% off for a number of well-known heuristics to this problem (e.g. First Fit and Best Fit) [4, 2]. Similar to these bin-packing heuristics, we will first rearrange the orders and then try to allocate them to the different agents, as if these orders just arrived (all existing orders that are not (about to) being executed are thus removed from the schedules first). The orders will be sorted

[5]The bin packing problem is the following problem: given a number of bins of size S and capacity C and a number of items $u_1, \ldots, u_m$ of size $s(u) \in Z^+$, find a distribution of the items over the bins, such that no bin contains more items that it can hold and the least number of bins is used.

in decreasing size (determined by the load and unload times and the distance between pickup and destination locations) and then by priority. Therefore, orders taking a long time to execute will be assigned to agents first, allowing the smaller ones to "fill up the gaps". In case not all orders can be assigned, the orders that cannot be fulfilled are examined separately, to see if there is an order with lower priority included in the plan that can be removed to make room for an order not included but with a higher priority. If so, we replace the order with lower priority by the one with the higher priority, thus trying to optimize the schedule for higher priority orders

The main task of the planning algorithm described above is to find a suitable plan whenever a new order arrives. This includes dealing with *incidents*: In case a transport agent reports a failure, the tactical planner has to determine which planned actions are affected. These actions then are then removed from the plans of the agents and the system tries to add them again as described above. Hence, these actions are simply conceived as new orders the that to be executed.

3 Experiments

We've performed a number of experiments to test the different heuristics that can be used to assign the orders to agents. Thus, given a set of orders, arriving one by one (possible during execution of the plans), how should the orders be assigned to the different agents such that the most orders as possible are scheduled, while keeping the cost of the plans (measured in distance) as low as possible. A environment in which some orders fail to execute properly (causing the operational agents to report errors) is not considered here. The aspects that we are interested in are the percentage of orders that the planner can successfully include in the plans (given that these orders can in fact all be accommodated) and the distance that is covered by the plans.

The time in which plans are generated is of course also important. However, the heuristics all perform more or less the same calculations: find the spot where the order should be executed and calculate (using a simple function) the heuristic value of that spot. Therefore, we will not discuss the execution times here as they are not interesting enough.[6] Before giving the results of the experiments, we first discuss the setup.

The experiments were conducted as follows: first we randomly generated plans for the agents, based on some given parameters. These parameters include the number of agents, the ranges for load and unload times and penalties, the length of the plans and their density. This density determines the amount of time in the plan that the agent spends doing nothing. The higher the density, the more difficult it is to find a schedule. The process of generating plans this way is illustrated in Figure 2. Here, the horizontal line denotes the density level that should be achieved. Steps 1 to 4 show how randomly generated orders are added tightly to the plan, to just below the density level. (The first order that is generated and

[6]Indeed, different heuristics took about the same amount of time (typically, a few seconds to about a minute for the longest experiments) for the same problems.

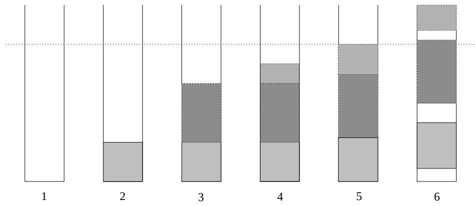

Figure 2: Schematic description of the randomization process

violates the density level, stops this process.) Then, we increase random load and unload times, until the plan has the desired density. Finally, the spare time in the plan is (randomly) distributed over the whole plan. Note that the problem instances that we generate this way do not have to be optimal (in any sense), since they are randomly generated.

After generating a plan, we used the different heuristics to compute a schedule. This was done for the following heuristics:

- *[Cooperative]* Orders were assigned to agents based on a slack maximizing and minimizing heuristic and one based on distance. In case an order could not be assigned, a reordering heuristic was used that ordered the orders based on their length and priority. Orders were then reassigned by the same heuristic as was used before to select agents.

- *[Compatitive]* Orders were assigned to the agent with the highest bid (based on distance (cost) and reward). When orders can not be assigned, agents try to exchange orders, but only if this decreases the cost of the plan for both agents.

Once an agent was determined, the adjustments of the time windows within one schedule was performed by Ilog Solver, a contraint programming engine [5].

3.1 Cooperating agents

The problems were generated with the following parameters: 1, 3 or 5 agents, plan densities: $10, 20, \ldots, 70, 75, 80, 85, 90, 95, 100$ per cent, (un)load times in the range $[0, 5]$ and penalties in the range $[0, 10]$. For each set of parameters, we averaged the results over 25 generated plans.

We will illustrate our findings by two graphs. In Figure 3 we show the performance of the distance based heuristic, for problems of 3 agents. For each plan density, it shows how many orders the problem consisted of, and how many were succesfully scheduled. This figure is representative for all heuristics and for all numbers of agents tested: the curves are all almost equal. Thus, the heuristics do not differ much in the percentage of orders they can schedule. There is, however, a difference in the average length of the orders. As illustrated in Figure 4 (which shows results for 5 agents), the distance based and slack maximizing heuristic always perform better than the randomly generated plan that was the input to the planners. The slack minimizing performed not as well, and generated slightly

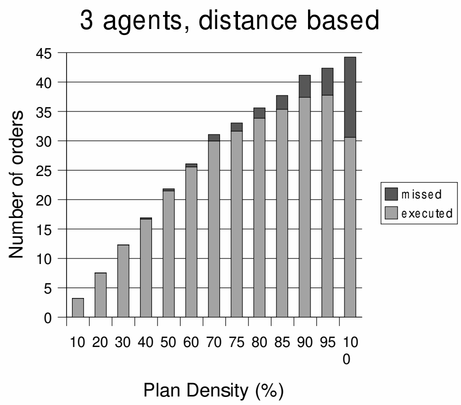

Figure 3: Number of orders succesfully scheduled for different plan densities.

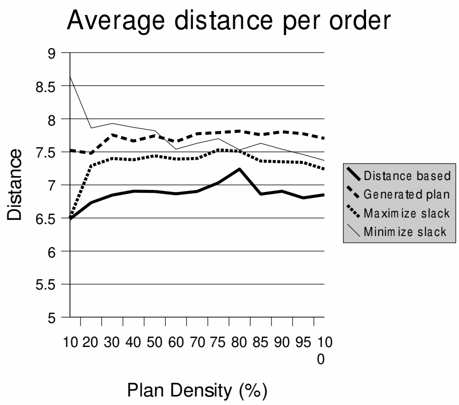

Figure 4: Number of orders succesfully scheduled for different plan densities.

longer plans. This is explained by noticing that travelling greater distances reduces the slack in a plan, since there is less time remaining to complete the order.

3.2 Compatitive agents

To test the heuristics for competing agents, we used the exact same plans as for the cooperating agents. Figure X compares the average cooperative performance and the performance of the competing agents. As we can see, the compatitive agents are outperformed as a group. This is due to two reasons: (1) some orders are not rewarding enough for an individual agent to execute and (2) fewer orders are exchanged between agents, as they only exchange if there is an immediate benefit for both agents.

4 Conclusions

We have sketched how a multi-level tactic can be used for scheduling transportation orders. A high-level tactical planner determines which order is executed by which agent, and lower-level operational planners carry out their orders. The tactical layer is described in more detail, and some possible heuristics for dividing the orders over the agents were developed. Using these heuristics, we have conducted experiments to evaluate the different strategies. It turns out that the heuristic has no great impact on the percentage of orders that can be scheduled. There is a difference in the travel distance, however. The distance based heuristic performed better (as could be expected), whereas the slack minimizing heuristic was the worst performer. When comparing competing agents with cooperating agents, the cooperating group was shown to outperform the others (again, as expected).

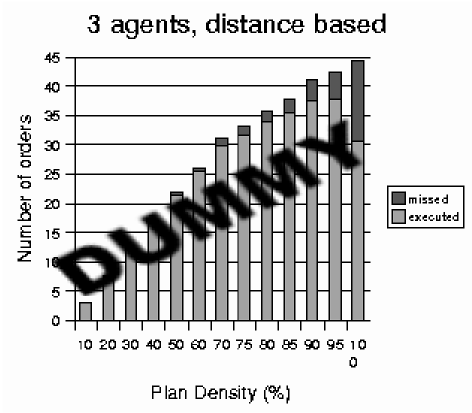

Figure 5: Number of orders succesfully scheduled for different plan densities.

Future work includes refining the strategies for incident handling and testing those. Also, a simulator is currently being developed which will be able to graphically show the performance of a tactical agent working together with several operational agents. The simulater will then be used to assess the validity of our technique.

References

[1] L.D. Aronson, R.P.J. van der Krogt, C. Witteveen, and J. Zutt. Automated transport planning using agents. In *Proceedings of the International Congress on Freight Transport Automation and Multimodality: Organisational and Technological Innovations*, 2002.

[2] P. Crescenzi and V. Kann. A compendium of np optimization problems. http://www.nada.kth.se/~viggo/problemlist/compendium.html.

[3] M.M. de Weerdt, A. Bos, H. Tonino, and C. Witteveen. Multi-agent cooperation in a planning framework. In *Proceedings of the Twelfth Belgium-Netherlands Artificial Intelligence Conference (BNAIC-00)*, pages 53–60, 2000.

[4] M.R. Garey and D.S. Johnson. *Computers and intractability – a guide to the theory of NP-completeness.* W.H. Freeman and company, New York, 1979.

[5] ILOG. *Ilog Solver 5.1 User's Manual and Reference Manual.*

[6] C. Witteveen R.P.J. van der Krogt, A. Bos. Replanning in a resource-based framework. In H. Krautwurmová V. Mařík, O. Štěpánková and M. Luck, editors, *Multi-Agent Systems and Application II (LNAI Volume 2322)*, pages 148–158. Springer Verlag, 2002.