

Tactical Planning using Heuristics

Roman van der Krogt^a Leon Aronson^a Nico Roos^b
Cees Witteveen^a Jonne Zutt^a

^aDelft University of Technology, Faculty of Information Technology and Systems, P.O.
Box 5031, 2600 GA Delft

^bUniversiteit Maastricht, IKAT, P.O. Box 616, 6200 MD Maastricht

Abstract

Modern transportation problems are highly dynamic and time critical. A planning system for transportation problems must therefore 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 operational 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. These agents can be cooperative (the scenario which we are primarily interested in), or competitive. 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 also free to retract or modify their orders. Associated with an order *o* there is a specification of its time windows, 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.¹ Finally, there is a static infrastructure, represented by locations, roads between them, global travel times, and distances on the roads.

The question is how to assign 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 minimized.² Even simpler problems than this one are easily shown to be intractable [4]. Therefore, any practical solution must be based on clever heuristics, i.e., problem solving methods that produce acceptable solutions but not necessarily the optimal ones.

We model this problem using a hierarchical agent-based approach (see Figure 1). On the higher level, we have a *tactical planning agent* responsible for handling incoming

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

²The 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.

orders and allocating them to individual transport agents. On the lower level, we have the *transport agents* taking care of the operational planning and its execution. Such a transport agent/operational planner may be a truck driver or a computer program running on an Automated Guided Vehicle (AGV). Each operational planner is only aware of those plans parts assigned to it by the tactical planner.

Orders, placed by customers, are received by the tactical planner and contain information about which freight has to be moved, when and where it should be 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 used by the transport agents, to compute a plan, consisting of order assignments.³

These order assignments constitute a sequence of orders for each agent A , that it would normally be able to execute.⁴ These order assignments are communicated to the corresponding transport agent.

Each operational planner is responsible for executing the orders assigned to it. It is also responsible for communicating any (expected) deviations from the orders when contingencies occur, such that the tactical planner can adequately adjust plans if 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 heuristics for the *tactical planner*. We start by giving a detailed look at the tactical planner and present possible heuristics for it. Next, we describe some initial experiments that have been performed. These experiments were limited to a static environment, where no incidents occur and give an indication of the performance of the developed heuristics. Finally, we present the results of these preliminary experiments and conclusions.

2 The tactical planner

The tactical planner focuses on efficient distribution of the orders. Of course, 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.

For each transport agent, the tactical planner stores a list of the orders that are currently assigned to that agent. This is a simple sequence of pairs of pickup and delivery locations,

³The tactical planner, 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 a later order can be served on time. It might even create additional orders by itself, for example to move a transport agent without freight to the pickup place of the next order.

⁴That is, if no incidents occur.

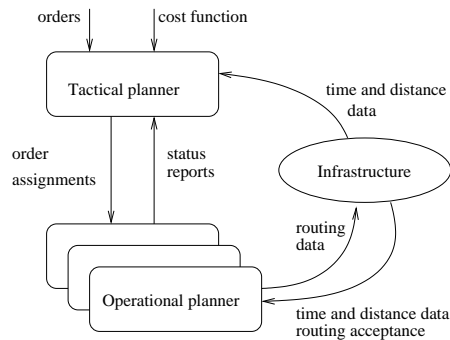


Figure 1: Planner hierarchy

along with their associated time windows.⁵ The *slack* between two consecutive locations is the amount of time that can be wasted during the travel, while still being 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 2.⁶ The slack of an order o_i in a sequence o_{i-1}, o_i, o_{i+1} is the time that can be lost from the moment of delivering o_{i-1} to the moment of pickup up o_{i+1} . 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 $o = (f, s, T_s, w_s, d, T_d, w_d, p)$ describing a freight f that is to be picked up at a source location s , within a time window T_s , and to be delivered at destination d within a time window T_d . Also included are the times w_s to load and w_d to unload the freight, as is the penalty p for not executing the order o within the given time. Upon receipt of an order, the planner first tries to insert the new order in between two existing (already assigned) orders. If this fails, the planner has to reshuffle orders in order to come up with a schedule in which all orders are included.

The generated plan is communicated to the agents by means of order assignment messages. Each such 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). Note that *empty rides* are generated when necessary, 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 time windows. 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 an operational planner are communicated through *status messages*. Such messages report the progress of an operational planner whenever it finishes (part of) an order or when it detects incidents it cannot handle on its own. 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.

⁵Empty rides between orders are not included in this list. They are however sent to the operational agents.

⁶In this paper we will assume that the speed of vehicles equals one distance unit per time unit.

maximizeSlack(order o , agents A)

1. for each agent $a \in A$ calculate the slack for o when it is included, i.e., the time that can be lost when executing o while a is still in time for the next order.
2. assign o to the agent with the greatest slack value.

minimizeSlack(order o , agents A)

1. for each agent $a \in A$ calculate the slack for o when it is included
2. assign o to the agent with the smallest slack value.

Figure 2: Slack-based heuristics

3 The planning heuristics

The tactical planner has to select agents to execute orders. 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. To overcome this problem, we consider two heuristic approaches: one for cooperative agents (the case we are primarily interested in) and one for competitive agents.

3.1 Cooperative agents

The cooperative heuristic iteratively examines each order that arrives in the system. It consists of two phases: *agent selection* and *replanning*:

In phase 1, an agent is selected for assigning an order to. Phase 2 is entered if no such agent can be found in phase 1. In that case a replanning method consisting of a reallocation of orders to agents is performed.

Ad 1: As a new order arrives in the system, each of the individual transport agent plans (possibly partial) is inspected for a place where the new order can be inserted without affecting the existing ordering. If found, the new order is assigned to the corresponding agent. Since slack is an important determinant of trade-off in robustness versus efficiency, we distinguish two selection heuristics based on slack: *slack maximizing* and *slack minimizing*. These heuristics are described in Figure 2.

Ad 2: In case there is no transport agent to assign an order to using an agent selection heuristic, 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*.⁷ Note that this problem closely resembles the problem of distributing orders over agents, using as few agents as possible, making heuristics for bin-packing suitable

⁷The bin packing problem is the following problem: given a number of bins of size S and a number of items u_1, \dots, u_m of size $s(u) \in \mathbb{Z}^+$, 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.

for 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), see [2, 4]. 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 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 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 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 to be executed.

Heuristics for competitive agents

Competitive agents are primarily focused on performing well on their own, and not as much on their joint performance. For each order that arrives, a tactical planner operating in such an environment collects bids from the transport agents and awards the order to the highest bidder (if any). It is possible that for some orders none of the agents makes a bid, because the rewards of the order do not outweigh the costs of executing it. Therefore, one may expect poorer group performance. By letting the agents negotiate about exchanging orders, some of these limitation may be overcome, as they can try to adjust their plans to include a new order.

4 Experiments

To get a first impression of the behavior of the developed heuristics, we focussed on their planning performance, i.e. we simplified the problem as follows: no incidents occur and orders are either scheduled in time or not at all. As cost is largely determined by the distance travelled and the penalties incurred these are the aspects that were measured.

Before presenting 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 the following parameters: the number of agents involved, the ranges for load and unload times and penalties, the length of the plans and their density.⁸ The higher

⁸The density of a plan is the ratio of the time an agent spends on executing orders in a plan versus the total

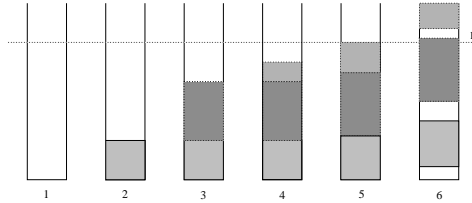


Figure 3: Schematic description of the randomization process

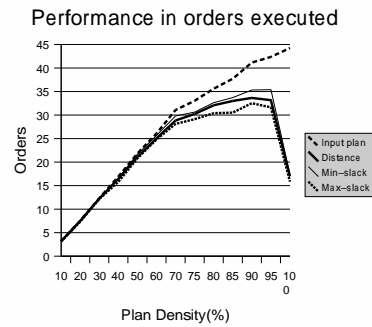


Figure 4: Number of orders successfully scheduled for different plan densities.

the density, the more difficult it is to find a schedule. The process of generating plans this way is illustrated in Figure 3. This visualizes the process of generating a plan for one agent. The horizontal line l 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 violating the density level stops this process.) Then we increase random load and unload times, until the plan has the desired density. Finally, the idle time in the plan is (randomly) distributed over the whole plan. For problems consisting of more than one agent, this process is repeated for each agent. Note that the generated problem instances do not have to be optimal (in any sense).⁹

For each plan that was generated, 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.
- [*Competitive*] Orders were assigned to the agent with the highest bid (based on distance (cost) and reward). When orders could not be assigned, agents tried to exchange orders, but only if this decreases the cost of the plan for both agents.

Once an agent was determined, the calculation of the new schedule was performed by generating the set of constraints for these orders (such as the time windows in which they have to be performed, how much time there should be between orders to allow for travel, etc). This set of constraints was then given to ILOG Solver, a constraint programming engine [5], that solved the problem.

4.1 Cooperating heuristics

The problems instances were generated using the following parameters: agents: 1, 3, or 5; plan densities: 10, 20, ..., 70, 75, 80, 85, 90, 95, 100 %; (un)load times: in the range [0, 5]; time spent in a plan (including idle time).

⁹Frequently, exchanging orders between the agents can improve the quality of the plans.

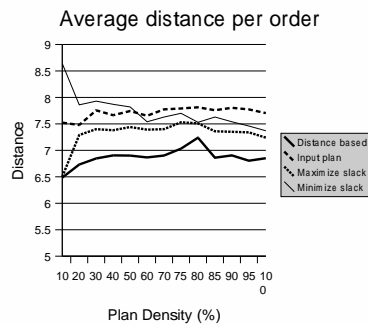


Figure 5: Average distance of executed orders for different plan densities.

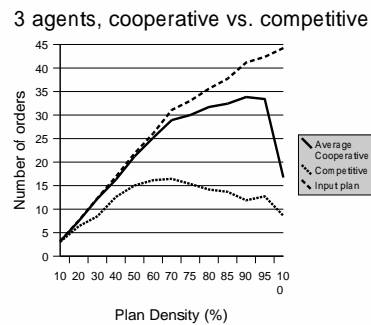


Figure 6: Number of orders successfully scheduled for different plan densities.

and penalties in the range $[0, 10]$. As infrastructure, we used a fully connected graph of 7 locations, with distances between 3 and 6. For each set of parameters, we averaged the results over 25 generated plans.

In Figure 4 we show the performance of the different heuristics for problems of 3 agents. For each plan density, it shows how many orders the problem consisted of, and how many were successfully scheduled by each heuristic.¹⁰ As can be seen in the figure, the heuristics minimally differ in the percentage of orders they can schedule. Contrary to what we expected, the slack-maximizing heuristic did not perform better than the slack-minimizing. However, the main strength of this heuristic lies in dynamic situations. Experiments with such situations will have to show if this heuristic lives up to its expectations. Also, all three heuristics have problems with very dense plans. For these situations, other heuristics may be developed. There is a difference in the average distance that is needed to execute an order. As illustrated in Figure 5 (which shows results for 5 agents), the distance-based and slack-maximizing heuristics always perform better than the randomly generated plan. The slack-minimizing performed not as well, and generated slightly longer plans. This is explained by noticing that traveling greater distances reduces the slack in a plan, since less time remains to complete the order. The execution times of the different heuristics were equal. Execution times did not exceed 15 seconds for problems with a density of 90 % or less. For denser plans, the times went up to a maximum of 60 seconds for some of the problems with a 100 % density.

The first large-scale experiments seem to agree with the above findings. For example, problems with 50 agents and a total of 5000 orders (75% density) can be solved in up to 10 minutes. In these cases, 96.9% of the orders is successfully scheduled.

4.2 Competitive heuristics

Competitive heuristics were tested with the same inputs as the cooperating heuristics. Figure 6 compares the average cooperative performance with the performance of the competing agents. As expected, the competitive agents are outperformed by the cooperative heuristics. This is due to two obvious reasons: (1) some orders are not rewarding enough for an individual agent to execute and (2) fewer orders are exchanged between agents, as

¹⁰Note that the plan densities are not evenly distributed, so the x-axis is not either.

they only exchange if there is an immediate benefit for both agents.

5 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. In contrary to our expectations, it turns out that the evaluated heuristics have a small impact on the percentage of orders that can be scheduled in a static environment. However, the slack-maximizing heuristic was developed with dynamic situations in mind. Experiments with such situations have to show whether our expectations are true under those conditions. Also, all suffer from serious performance loss when the plans are very dense (95-100%). It has to be investigated if there are other heuristics that do perform well in this range. When the distance per order is investigated, the performances do differ. The distance-based heuristic performed better (as could be expected), whereas the slack-minimizing heuristic was the worst performer. When comparing competing agents to cooperating agents, the cooperating group was shown to outperform the others.

Future work includes refining the strategies for incident handling. Further experiments will have to show how the heuristics perform in such a dynamic environment. 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 simulator 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 Weerd, 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.