Automated Transport Planning using Agents

International congress on Freight Transport Automation and Multimodality, Delft, 23 & 24 May 2002

Authors

Leon Aronson, Roman van der Krogt, Cees Witteveen and Jonne Zutt
Delft University of Technology,
Faculty of Information Technology and Systems
{L.D.Aronson | R.P.J.vanderKrogt | C.Witteveen | J.Zutt}@its.tudelft.nl
P.O. Box 2600 AJ, Delft, The Netherlands
Phone L.D. Aronson +31 (015) 2784433

Contents

Abstract

Modern transportation problems are highly dynamic and time critical. A planning system for transportation problems therefore must include efficient and flexible incident management and replanning strategies.
In this paper we present a general agent-based framework for highly dynamic order-based transportation planning problems. The basis of our framework is a distribution of responsibilities between a strategic planner and several tactical planners (one for each vehicle). The *strategic planner* is responsible for task allocation of orders and tries to maximise the amount of time that can be used for time-shifting parts of the plan, thus trying to obtain a flexible schedule in which new orders can easily be fit in. The *tactical planners* are responsible for individual route planning, execution of the orders and handling of small incidents. If incidents have consequences for other tactical planners or result in a failure to deliver an order, a tactical planner reports this failure back to the strategic planner.
In computing routes the tactical planners make use of a *smart infrastructure*. Roads and crossings will provide information about the traffic of other vehicles and these infrastructure agents compute time windows for passing sections on the route. This information is used by a route-planning heuristic (D**) to determine a feasible route. Finally, this route then is announced to the infrastructure in order to make reservations for route sections and prevent collisions.

1 Introduction

The transportation problem we will analyse can be described as follows. There is a *static infrastructure*, represented by locations, roads between them, global travel times, and distances on the roads. There is also a number of *transport agents*. These transport agents are capable of moving cargo units. Cargo units, of which a number are available, can contain freight.[1] There are also customers 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 in advance. Associated with an order there is a specification of its *time windows*, its *yield* if it is executed within the time windows and its *penalties* if it is rejected or we fail to execute it within the time windows specified.
Our problem now consists of planning the orders and assigning them to the transport agents in such a way that in executing this plan we minimise the total cost, which is dependent of time, distance and the penalties assigned. Also, we must be able to react on incidents to minimise their impact on the total performance.

More specifically, our problem includes the following features:

- bobtailing: moving a transport agent without a cargo unit;
- empty rides: moving an empty cargo unit;
- drop-and-pick: cargo units can be dropped off at a location to (un)load it;
- incidents: roads can be congested or jammed, transport agents can break down;
- time-windows: pickup and delivery time windows can be specified by the customer;
- penalties: whenever pickup or delivery falls outside the specified time window, a constant penalty is applied;
- cost function: the cost can depend on time, distance, or both in an arbitrary way; both transport agents and cargo units may induce costs; and
- home locations: each transport agent and cargo unit can have one or more home locations; staying at a home location has reduced time cost.

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.

Even much simpler problems than the one described above are intractable, i.e., they are not optimally solvable in an efficient way. The use of time windows and the 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 ones. As motivated

[1] Sometimes, cargo units are physically separable from a transport agent, as in the case of tractor/trailer combinations, and sometimes they are physically the same object, as in the case of AGVs. In the latter case the plans for the transport agent and the cargo unit must be equivalent.

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

1.1 Applications

There is a large variety of transport problems fitting into our model. All these problems are order-based, have pickup and delivery specifications, and consider transportation means that can function autonomously. We have explicitly accounted for the following three problems.

- *Taxi planning*: customers specify a pickup time and location, then a delivery location. The taxi company has a number of taxis available. The goal is to minimise both the costs and the customer waiting time.
- *AGV planning* ([Duinkerken, 2000], [Duinkerken, 2001]): Automatic Guided Vehicles drive on a terminal to move containers from a ship to several stacks and vice versa. Besides loading and unloading ships quickly, the use of routes by AGVs must satisfy some restrictions, so the route planning must be explicitly done by the AGV.
- *Intermodal transport pre- and end-haulage* ([NEA, 1999], [Konings, 2002]): containers arrive at and depart from a terminal, e.g. by train. The transportation of these containers from and to their actual pickup and delivery locations is provided by tractor/trailer combinations. If the (un)loading of a trailer takes a long time, the tractor can be disconnected to perform other transports during (un)loading.

Furthermore, our model can be easily adapted to planning problems considering ships and aeroplanes.

1.2 Approach

We model the described planning problem by using a hierarchical agent-based model. On the highest level, we have a strategic planning agent[2] 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 tactical planning and its execution. A transport agent / tactical planner may be a truck driver or a computer program running on an AGV. Each tactical planner is capable to see only that part of the plan it is allocated to by the strategic planner.

The strategic planner is responsible for assigning orders to transport agents such that the whole is efficient and reliable. This has the advantage that it does not have to bother with planning details.
Each tactical 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 strategic planner, such that the strategic planner can adequately adjust plans when necessary. Since

[2] The strategic planner can be implemented by either a human or an automated process.

tactical planners operate on an infrastructure, it may have means to communicate with the infrastructure, e.g. for AGVs to claim parts of a terminal, e.g. for trucks to obtain traffic information.

2 Model

Using this approach, we now present our model in more detail (see Figure 1). Orders, placed by customers, are received by the strategic planner. They contain information about the freight that has to be moved, when and where it should picked up, how much time the loading takes, and similar information about the delivery. The strategic 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.

The strategic planner, however, does not have to literally copy the order data into an order assignment. It has some freedom in specifying the assignments. For example, the strategic planner might restrict the time window of an order such that the next order (assigned to the same agent) can be served on time. It is even possible that strategic planner creates an order assignment by itself, for example to move a transport agent without freight to a place where the next order has to be picked up.

The tactical planners receive order assignments and try to execute them. This means finding a route from the pickup location to the delivery location. If the assignment is feasible and nothing special happens, this suffices to carry out the order assignment. If an order assignment has been carried out successfully, this is reported back to the strategic planner by sending a status report. However, if an incident occurs, the tactical planners have to check whether it affects any of their order assignments, and if so, if those order assignments can still be carried out as specified. If this is no longer the case, tactical planners should look for an alternate route. Only if a tactical planner cannot find a way to meet the requirements of the strategic planner, it sends a status report to the strategic planner containing information about the problem that has occurred and the estimated influence it has on the order assignment. Upon

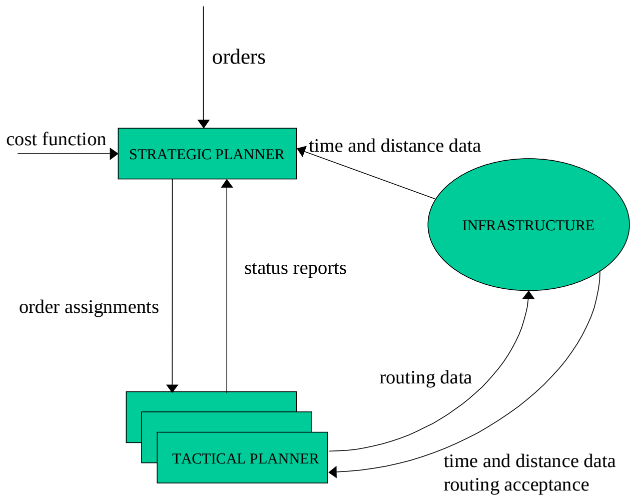


Figure 1. Model layout

reception of such a status report, the strategic planner decides whether the order assignment must still be carried out with some changes, or that a re-plan is necessary, for example by shifting an order to another transport agent.

To plan its route, the tactical planner needs information from the infrastructure. This includes static information about roads, distances, and travel times, but also might include dynamic information, like traffic information. The specific type of dynamic information depends on the application. The communication between tactical planners and infrastructure for the AGV case will be described in the section about tactical planners. In the AGV case, communication is also needed from the tactical planners to the infrastructure. This is needed because the use of parts the AGV terminal is subject to some restrictions. Thus, this infrastructure plays a role in enforcing its rules. This will also be described in the section about tactical planners.

We will now dedicate two sections to planning in more detail: one to strategic planning and one to tactical planning.

3 Strategic Planning

According to our model, the strategic planner assigns the different orders to the different transport agents. It plans on a high level, i.e., the planner does not consider details such as the exact route a vehicle takes. Rather, it focuses on distributing the orders efficiently. It is the responsibility of the transport agents to calculate their precise routes. A strategic planner could be designed using a general-purpose planner and replanner, such as [De Weerdt, 2001] and [Van der Krogt, 2001]. 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.
We will first describe the internals and the communication with the rest of the system. Hereafter we detail the manner in which the strategic planner handles the problem. Finally, an example is presented.

3.1 Internals

Figure 1 shows the embedding of the strategic planner in the planning system: the inputs are orders, order updates, and status reports from tactical planners, and the output is a flow of order assignments (either new, or changed assignments).
For each transport agent, the strategic 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 the associated time windows. Empty rides that do not correspond to orders are not included in this list; they are implicitly there in case the delivery location of one order does not match the pickup location of the next. The reason for this is that only the orders are important during the planning.

The *slack* between two consecutive locations that a transport agent is 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 two 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 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. Thus, the amount of slack in a plan is an important factor in determining the trade-off between efficiency and robustness in plans.

3.1.1 Orders

As described in the model, an order is a tuple (*f, s, T1, d, T2, p*) that describes a freight *f* that is to be picked up at a source location *s*, within a certain time window *T1*, and to be delivered at a destination *d*, also within a certain time window *T2*. *p* is the penalty for not executing the other within the given time windows. Note that the penalty assigned to an order can also be seen as the *priority* that the particular order has. Furthermore, we assume that it is known how much time it will take to load or unload the freight *f*.
Upon receipt of an order, the planner first checks the plans of the tactical planners to find out if it can insert the order in between two existing assignments. If this is the case, the planner inserts the order into one 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.

3.1.2 Order assignments

The generated plan 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. In contrast to the internal representation of the plan, empty rides are included in this list. That is, 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 strategic and tactical planner.

3.1.3 Order changes

The strategic planner also receives order changes. These originate from either the customer or a tactical planner detecting an incident it is not able to resolve.
Order changes arriving from customers can either be changes within orders (such as a new pickup time window or delivery location) or complete cancellations 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 tactical planners are communicated through *status messages* and covered in the next section.

3.1.4 Status reports

A tactical planner reports on its progress whenever it finishes (part of) an order or if it detects incidents it cannot handle on its own. These messages are called *status reports*. If the agent reports normal progress, these messages contain the precise time at which pickup or delivery took place. The strategic 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 an estimate of *(a)* the place the agent is now, *(b)* the kind of incident and *(c)* an estimate of the delay the agent expects for his order to be delivered. This information is used by the strategic planner to decide what to do with this order and the other orders assigned to the agent. For example, if an order is delayed such that an agent can still carry out its next assignment in time, the strategic planner might decide to just let the order be delayed instead of trying to reschedule the order to another agent.

3.2 The planning algorithm

As stated above, the strategic planner is responsible for selecting which agent has to carry out which order. To reach an optimal solution, it would require the planner to consider every possible distribution of orders over agents[3] 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 guidelines (called *heuristics*).

[3] The reason is that this planning problem is a highly intractable (PSPACE-complete) problem

This heuristic function consists of two parts:

- *Agent selection*
  Each of the individual transport agent plans (order sequences) is inspected to look for a place where the new order can be inserted without disturbing the existing sequence. If such places exist, this heuristic selects an agent to carry out the new order.
- *Redistribution of orders*
  If a new order arrives and the first heuristic cannot find suitable places to fit the new order a redistribution heuristic is used. The existing orders are redistributed in order to create space (if possible) for the new order. Next, the first heuristic is used to assign the order to a suitable agent.

We briefly discuss both heuristics and how they are used.

3.2.1 Selecting an Agent

Slack is an important trade-off in plan quality, which can be used as a heuristic. As we've seen, the trade-off is large slack and robustness or small slack and efficiency. At the extremes, we can create two heuristics:

- *Maximise slack*: if there are a fixed number of transport vehicles available, we can create robust plans by choosing the vehicle that has the biggest slack remaining when the current order is included. For example, suppose that there is an order to be picked up at time 10 and that we have two vehicles *A* and *B* available. Suppose *A* can be at the pickup point at time 5 and *B* can be there at time 7. Thus we choose *A*. If there would be some problem when *A* drives to the pickup location, it has more time to solve the problem than if we would have chosen *B*. Also, if new orders arrive, this strategy maximises the probability that an order can be inserted between two existing orders.
- *Minimise slack*: if we would like to use as few transport agents as necessary, the slack should be minimised. The less slack, the less time is "wasted" between two orders. This scheme minimises the number of transport agents, because an agent that does not carry out any orders yet, has per definition a bigger slack that an agent that can insert the order in its current plan. For example, suppose that there is an order to be picked up at time 10 and that we have two vehicles *A* and *B* available. Suppose *A* can be at the pickup point at time 5 and *B* can be there at time 7. Thus we choose *B*. Thus the orders of *B* are closer together so it spends less time waiting to execute the next order.

3.2.2 Rescheduling orders to fit new ones

In case there is no transport agent to assign an order to using the first heuristic, the existing orders have to be rescheduled in order to make room for the new order, 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 *bin-packing* problem.
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 $s(u)$, 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. 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 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]
Similar to these bin-packing heuristics, we will first arrange the orders and then try to allocate them to the different agents, as if these orders just arrived (all existing schedules are thus cleared first). The orders will be sorted in decreasing 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, we examine the orders that cannot be fulfilled, to see if there is an order with lower priority included in the plan that can be removed to make room for the order with 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.

Remark
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 strategic planner has to determine which planned actions are affected. These actions are then removed from the plans of the agents and the system tries to add them again as described above, taking into account the new circumstances that led to the incident. Hence, these actions are simply conceived as new orders that are to be executed[5].

3.2.3 Pseudo-code

To clarify the strategic planner, we present some functions of the algorithm discussed above in pseudo-code (For brevity, we do not include any messages that are sent to the tactical planners).
The `select-agent` function describes a maximise-slack heuristic:

```
select-agent(group of agents A, order o)
    for all agents in A
        calculate the slack for order o
    if there is no agent with positive slack
        return failure
    else
        assign o to the agent with the largest
        slack
```

For each agent, the slack value is computed and then the order is assigned to the agent with the largest slack value. (A minimise-slack heuristic would assign the order to the agent having the smallest positive slack.)

[4] See [Garey, 1979] for a more detailed analysis of these heuristics. More recent results and references can be found in the compendium of NP optimization problems [Compendium NP].
[5] For actions that are being executed at the time the incident occurs, the planner also has to decide whether or not to assign them to another agent. This decision is made based on the penalties of the orders involved. The order with the lowest penalty is not executed or executed with delay

The `reorder` function takes a group of agents and a set of orders and then sorts these orders in decreasing size as described previously:

```
reorder(group of agents A, set of orders O)
    sort the orders in decreasing size
    clear all plans of the agents
    for all orders o in O
        if select-agent(A,o) fails
            add o to the list of failures
    for all orders that could not be assigned
        if an order o' exists in a plan that has
        lower priority and can be replaced by
        this order
            replace o' by o
```

It then tries to assign all orders to an agent, keeping track of which orders could not be executed. After all orders have been assigned, we search for orders that are not executed currently and can replace an order with lower priority. If so, then we exchange the order with lower priority for the one with higher priority.

`Add-order` uses the above two functions to add an order to the plans:

```
Add-order(group of agents A, order o)
    if select-agent(A,o) fails
        reorder(A, all orders)
```

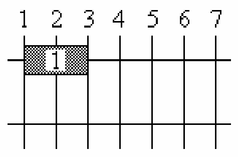
If it is not possible to add an order, it tries to shuffle the orders in order to make room for all of them.

3.3 Example

We present a small example that demonstrating the operation of the strategic planner. Consider the next four orders that are to be scheduled for two transport agents starting in *A*, operating between three cities *A*, *B*, and *C*, with distances as shown below.

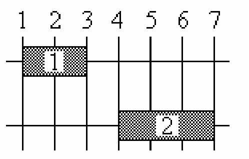
1. $A \rightarrow B$, pickup between [1,3], delivery between [2,5], priority 1.
2. $B \rightarrow C$, pickup between [4,5], delivery at [7], priority 1.
3. $B \rightarrow A$, pickup at [5], delivery between [7,9], priority 2.
4. $A \rightarrow C$, pickup between [2,3], delivery between [6,7], priority 1.

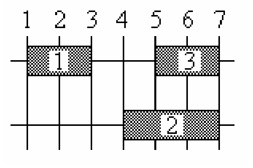
Planning now proceeds as follows, assuming a slack-maximising heuristic:



First, order 1 is scheduled. This situation is depicted at the left. The two horizontal lines depict the schedules for each agent, and as shown, agent 1 is going to pickup order 1 at time step 1, and will deliver it at time step 3.

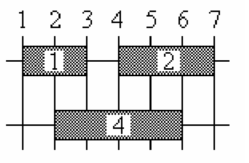
Secondly, order 2 is scheduled. This order is rewarded to agent 2, as it has the biggest slack for this order (as it does not have an order assigned to it yet). A Slack-minimising heuristic would have awarded this order to agent 1.

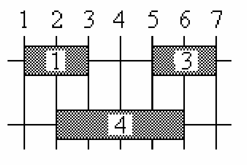




The third order is given to agent 1. This is the only agent capable of executing the order, thus making it an easy choice. Note that order 3 may be delivered up until time step 9. This schedule thus shows the 'optimistic' schedule.

Next, the fourth order is tried. However, this order does not fit into any of the schedules and thus requires replanning. The order in which the orders are given to the planner is determined by size and priority, and is as follows: 4, 2, 1, then 3. After the first three have been scheduled, the division of orders over agents is as given beside.





For each of the remaining orders, we check if it can replace an order of lower priority. This is the case: order 3 has higher priority than order 2, and thus is added to the schedule in stead.

4 Tactical Planning

To each transport agent, a tactical planner is assigned (see Figure 1). This tactical planner receives the orders assigned to the transport agent by the strategic planner and has several responsibilities. First, it has to compute a *route* for the transport agent that corresponds to the given order. If the transport agent traverses this route, the freight can be picked up and delivered correctly without violating the time-windows. Second, tactical planners respond to *incidents*. Incidents are events that cannot be anticipated in advance (i.e. they cannot be taken into account by the strategic layer). Examples of incidents are traffic jams, vehicle breakdowns, road maintenance, etc.

4.1 Routing algorithm

Once the order the transport agent has to deal with is known, this agent has to compute a route from its current position via the source (pickup) location of the order to its destination (delivery) location.
To compute a route, given some transportation order, we use the D**-algorithm, a variant of the D*-algorithm (see [Stentz94, Stentz95]). D* in turn has to be considered as a dynamic variant of the well-known best-first search A*-algorithm.

Remark
The A* algorithm is called a best-first search algorithm since it keeps track of a list of partial solutions to the route finding problem and it tries to expand the most promising alternative first. To determine which alternative is considered best, A* maintains an *estimate* of the final value of each partial solution. This value is the sum of the costs made so far (in the partial solution) and an estimate of the additional costs needed (to complete the partial solution) using a heuristic function. Obviously, not any function can be used; in order to be useful the heuristic function must always *underestimate* the costs of the complete solution – containing the partial solution – to produce optimal results. If a heuristic function satisfies this property, it is called *admissible.* The more accurate the heuristic function is, the more effective the search.

The D* algorithm is a *dynamic* variant of A* in the sense that road costs might change during computation. Using such dynamic cost values it is possible to deal with more practical situations. For example, we can create a traffic-aware routing algorithm, which will automatically spread the traffic loads in order to minimize traffic jams. Like the A* algorithm, the D* algorithm maintains a list of partial[6] paths – called the 'open' list – that is sorted by costs. Each time, the first of these partial paths is removed from this list and further examined. This partial path is considered the best alternative. Furthermore, there is a focusing heuristic that also affects the sorting of the 'open' list. This function directs (focuses) the search towards the goal location. A good example of such a function is the function that computes the Euclidean distance between the current location of the transport agent and the goal location. It can easily be shown that this function is optimistic (under-estimating the costs) and therefore admissible.

[6] A partial path is a simple path starting at a transport agent, but not (yet) ending in the goal location.

Our D** algorithm is an extension of the D* algorithm, adding the following features:

- *distributed algorithm:*
  transport agents do not have a complete map of the infrastructure[7], the only requirement is that they can communicate with all locations;
- *sensitivity for cost changes:*
  we only replan if, for a certain road, traversal costs changed significantly. This extension is needed to control the computation time. In order to use the D* algorithm effectively, one must be able to limit the number of replanning;
- *hard time-windows:*
  we are not interested in costs of a route that fails to meet time constraints, as opposed to soft time-windows where penalties are assigned for violation of the time-windows. In case there is no possible solution that satisfies all constraints, this is reported to the strategic planner (together with an indication of the seriousness of the problem).
- *alternative routes:*
  apart from the optimal route we also store several alternatives in order to react quickly to certain events. In some cases, there is no time to compute an alternative route. Therefore, we store several alternative route that the transport agent can adopt quickly.
- *traffic-aware cost function:*
  the costs of a road also depend on the plans of other transport agents. This cost function will be discussed in more detail later;
- *multi-agent:*
  we plan for multiple agents, whereas the original algorithm was used for single robot's path planning.

Equation 1 displays the cost function of the D** algorithm which we use. This cost function can briefly be explained as follows. The function *height(t)* denotes an estimated traffic load at time t. Each road stores a time-window T_a for each agent a denoting agent a traverses the road within time-window T_a. If a time-window is large, the chance that the agent traverses the road at a certain moment in time is smaller, so we divide by the length of the time-window $|T_a|$. Function *load(T)* is the average load within time-window T.
The cost function *cost(a,s,d,T)* then computes the cost for agent a to travel the road from s to d somewhere within time-window T. The first term of this function is the distance divided by the speed (minimum of the speed of the agent and the maximum speed allowed at the road) and the second term is related to the traffic load. Constant c denotes the capacity of the road; its value determines the influence of the traffic load. Figure 2 shows three examples of a cost function with varying capacities.

[7] The D* algorithm can also be used on partial (even empty) maps of the infrastructure, however, the D* algorithm the map stored by the robot is an incrementally increasing (partial) map.

$$height(t) = \sum_{a \in A} \begin{cases} 0 & t \notin T_a \\ \frac{1}{|T_a|} & t \in T_a \end{cases}$$

$$load(T) = \frac{\int_{T_1}^{T_2} height(t)dt}{|T|}$$

$$cost(a, s, d, T) = \frac{\delta_{sd}}{\min(speed_a, speed_{sd})} + e^{\frac{load(T)}{c}}$$

Equation 1: Traffic-aware cost function.

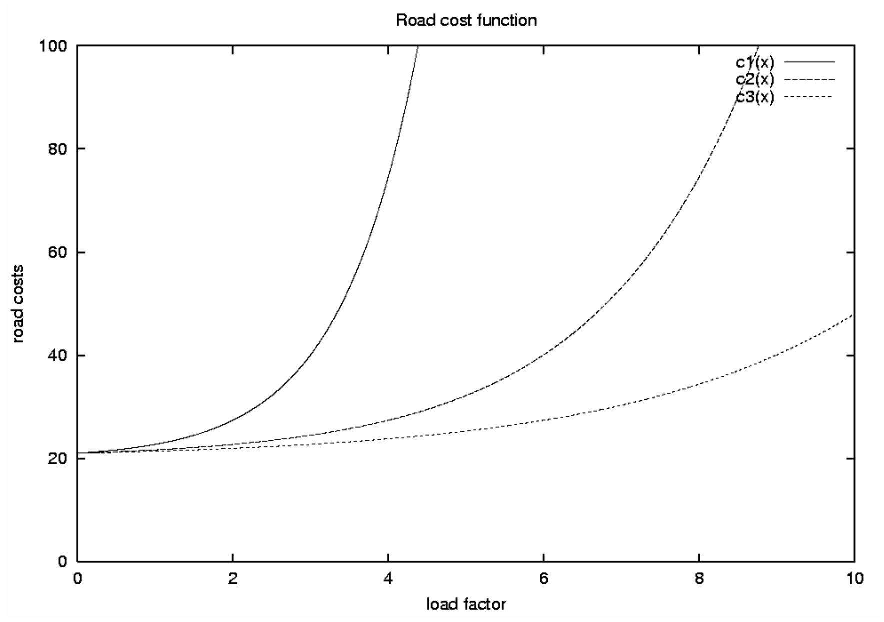


Figure 2: Three example cost functions.

4.2 Incidents

There are many different types of incidents. In this section the incidents relevant to the tactical planner are listed.
Usually, incidents usually require replanning. The D** algorithm, however, tries to minimise the amount of replanning needed by locally analysing the effect of the incident (using a focusing heuristic).

All incidents mentioned include a time at which they originate (at which they are known) and a time interval within which they have effect.

Roadblock
Roadblocks affect the maximum speed attribute of a road. For example, if a road normally has its maximum speed attribute set to 27.8 m/s (100 km/h), in case of traffic jam this might become 11.1 m/s (40 km/h) or in case of a collision it might change to 0 m/s.

If a road block incident occurs the D** algorithm makes sure that all transport agents for which their time-window to visit this road and the time-window of the incident overlap are notified of the incident. These transport agents replan if the change is considered significant.

Note that as soon as the roadblock incident is no longer present, all transport agents have to be notified (each transport agent could now prefer to visit this road) to ensure optimality of the computed route.

Transport agent damage
This incident affects the speed of the transport agent. In many cases, faulty transmitters and receivers lead to communication problems and in such cases the transport agents are only allowed to travel with limited speed. Other vehicle damage might limit its speed physically or vehicle breakdown will even set it to zero.
In such a case, the tactical planner related to the affected transport agent replans; if it is the case that other transport agents are also affected (e.g. in AGV terminals where AGVs cannot pass each other) this incident automatically raises a roadblock incident.

Order change / cancellation
If certain order attributes (e.g. delivery location, delivery time-window, freight) change, the strategic layer will be notified. Possibly, these changes result in replanning. If the order is already in execution – for example, the order is pickup-up and a transport agent is driving to its destination location – this incident must also be reported to a tactical planner. A tactical planner may have to decommit its route and replan.

4.3 Example

To illustrate our tactical planner, this section provides two figures created automatically by our tactical planner and visualised using *aiSee*'s graph visualisation tool[8]. The graph consists of locations (the boxes) and roads (the edges). The roads are labelled with $[\$cost, A_1, A_2, A_3, ..., A_n]$, where *$cost* denotes the costs – not taking traffic loads into account – and $A_1, A_2, \ldots, A_n$ are the identifiers of the transport agents that planned to traverse the particular road.

In this example, 10 transport agents (numbered from 0 to 10) all reside in location A. All are directed by the strategic layer to drive to location C. The tactical planner uses the D** algorithm to determine a traffic-aware optimal route for the transport agent (in the order of their indices). The result is shown in Figure 3. We observe that the first seven transport agents use the route [A, AR, B, BL, D] and the other three the

[8] See http://www.AbsInt.com, AbsInt Angewandte Informatik GmbH; originally developed at Saarland University.

route [A, AL, B, BL, D]. The latter have a different route in the beginning, because the previous transport agents crowded the road from A to AR. In the upper part of the figure, between location B and D, this does not occur, because the amount of traffic over the cheapest route does not weigh against the extra costs of the next best route (of course this depends on the capacities of the roads and the weighing constants used in the cost function).

After this planning we introduce a roadblock incident at the road from location AR to location B. The D** algorithm automatically replans for transport agents 0 to 6. As illustrated in Figure 4, all transport agents now prefer to travel via location AL instead of location AR.

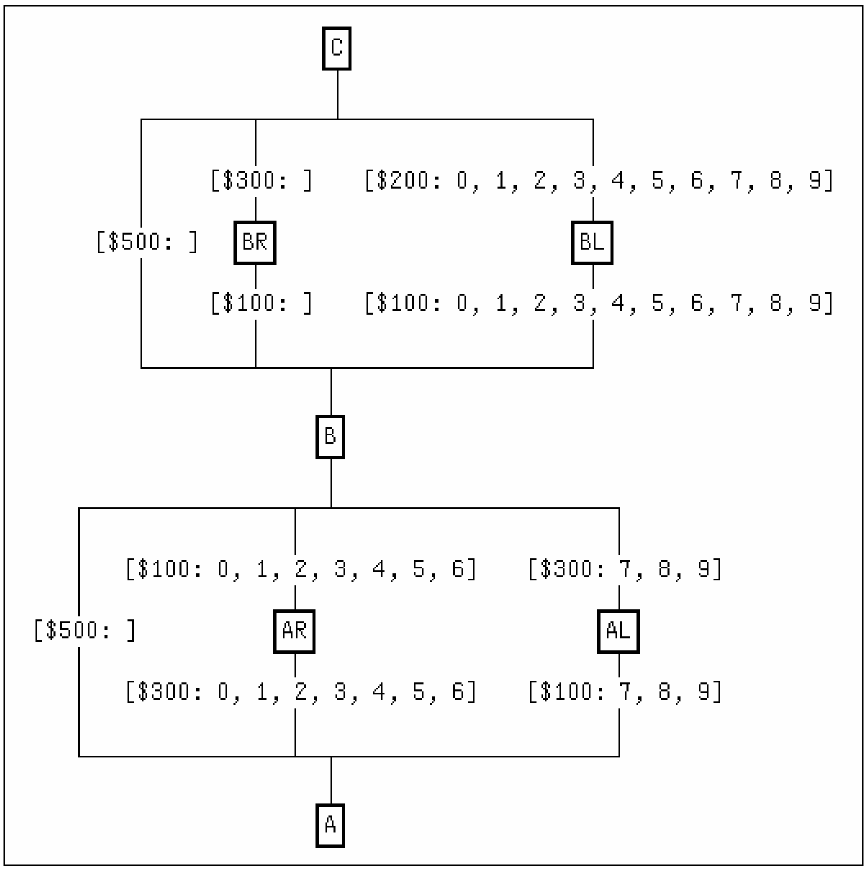


Figure 3: after planning.

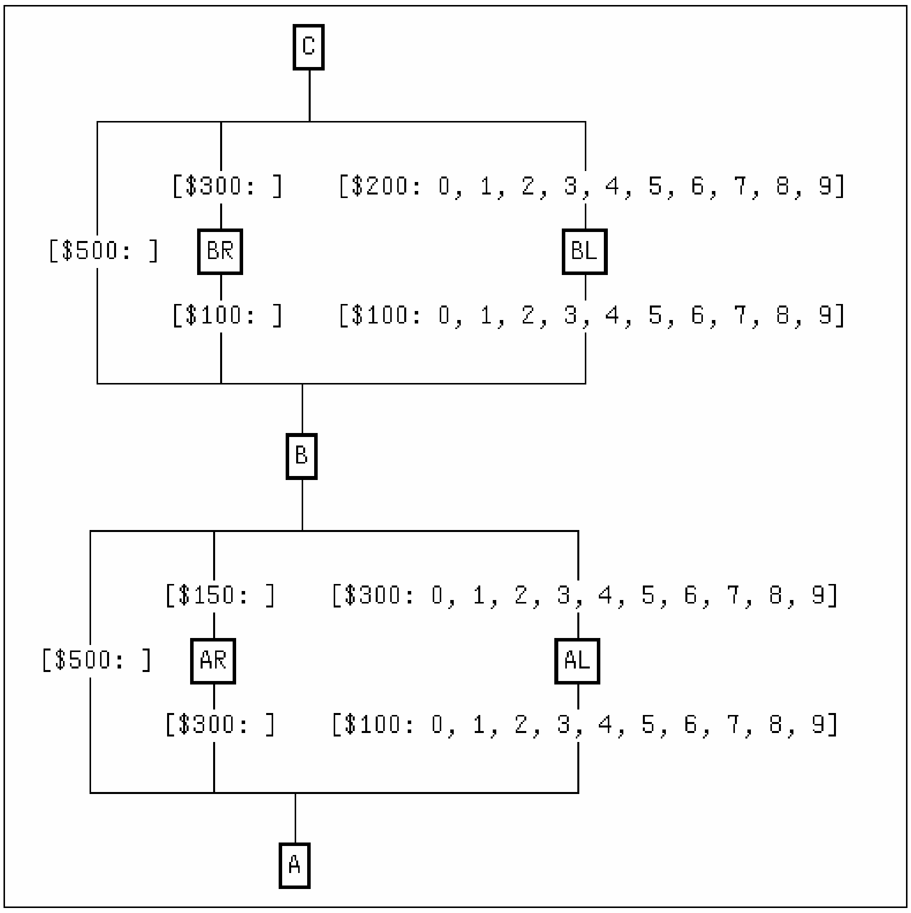


Figure 4: after replanning.

5 Conclusions

In this paper we have outlined an architecture and some algorithms of a flexible and efficient distributed (re)planner for freight transport capable to operate in a highly dynamic world.
Distinguishing features of our planner are

- a layered architecture distinguishing a strategical and a tactical planner
- the integration of planning and replanning (dealing with incidents)

The algorithms proposed are extensions of well-known heuristic approaches to both planning and scheduling as well as route-finding problems, integrated in one framework.
An implementation of the planner is expected to be finished soon and will be tested in a number of artificial and real-life dynamic planning environments.

Future work will pertain to the introduction of various extensions of the current framework. First of all, the hierarchical approach we followed in distinguishing a strategical and a tactical planning layer needs to be refined by distinguishing intermediate layers, e.g. tactical planners that are able to plan for a (small) group of transport agents and a distributed number of strategic planners.
Secondly, our planning framework assumes a fully cooperative planning environment. It would be interesting to extend the current framework by including forms of *competition* at both layers. By distinguishing more than one strategic planner, we can easily introduce auction mechanisms to model various forms of competition between strategical planners representing transportation organisations.
The same auction mechanisms, however, also can be included to model more refined forms of interaction between tactical and strategical planners.

References

Compendium of NP Optimisation Problems,

De Weerdt, M.M., H. Tonino, and C. Witteveen. (2001) Co-operative heuristic multi-agent planning, in: *Proceedings of the 13th Belgium-Netherlands Conference on AI,* Amsterdam, pp. 275-282.

Duinkerken, M.B., J.A. Ottjes (2000) A simulation model for automated container terminals. *Proceedings of the Business and Industry Simulation Symposium (ASTC 1999).* April 2000. Washington D.C. [SCS]. ISBN 1-56555-199-0

Duinkerken, M.B., J.J.M. Evers, J.A. Ottjes (2001) A Simulation Model for Integrating Quay Transport and Stacking Policies on Automated Container Terminals *Proceedings of the 15th European Simulation Multiconference (ESM2001),* Prague [SCS]. ISBN 1-56555-225-3.

Garey, M.R., D.S. Johnson (1979) *A Guide to NP-Completeness*, W.H. Freeman, New York, pp. 124-128.

Konings, R., E. Kreutzberger, L.D. Aronson (2002) Improving pre- and end-haulage in intermodal transport: development of an efficient planner, in: *International FTAM Congress 2002*, Delft.

NEA, Inrets, BCI et al. (1999) IMPREND: Improvement of pre- and end haulage from terminals, *research commisioned by the European Commision DG VII Brussels.*

Stentz, A. (1994) Optimal and Efficient Path Planning for Partially-Known Environments, in: *Proceedings of the International Conference on Robotics and Automation,* San Diego.

Stentz, A. (1995) The Focussed D* Algorithm for Real-Time Replanning, in: *Proceedings of the 14th International Joint Conference on AI,* Montreal.

Van der Krogt, R.P.J., A. Bos, C. Witteveen (2002) Replanning in a Resource-Based Framework, in: *Multi-Agent-Systems and Applications II, LNAI 2322*, Springer Verlag, Berlin, pp. 148-158.