# Operational Transport Planning with Incidents

## Experiments with TRAPLAS

**TRAIL Research School, Delft, November 2006**


## Authors

**ir. J. Zutt and prof. dr. C. Witteveen**

Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology

# Contents

# Abstract

This paper describes a *resource-based model* for operational pickup and delivery transport planning. The model is well-suited for transport networks with limited capacity, for example, Autonomous Guided Vehicle (AGV) terminals, such as European Container Terminals (ECT) or the underground logistic system (OLS). The operational level abstracts from the exact position of transport resources (i.e. vehicles) within the resources, but nevertheless the model does provide equations that have to guarantee the safety of the system.

After presenting the model several *planning methods* are described that control the software programmed transport resources. The design goal of the planning methods is two-folded: *(i)* to discover which knowledge is required to obtain transportation plans with good performance and *(ii)* to develop robust planning methods that can react to emerging disruptions such as malfunctioning transport and infrastructure resources.

Finally, using a newly developed transport planning simulator called TRAPLAS, experiments are discussed to compare the performance of these planning methods under normal circumstances and in incident cases. The paper finishes by reporting the experimental results, drawing conclusions and considering future work.



# Keywords

Resource-based model, Operational transport planning, Planning methods, Incident management, Robustness, Malfunctioning resources.

# 1 Introducing a resource-based model for operational pickup and delivery transport planning

This paper describes a model for operational pickup and delivery transport planning, designed for systems where the transport network has limited capacities. Planning methods are developed to execute pickup and delivery transportation orders taking into account these limited capacities. How can routing algorithms be constructed that are executed by individual agents and are able to construct efficient joint transportation plans?

In this paper the question *how much knowledge* about the plans of other agents is addressed. Furthermore, the robustness of these planning methods is investigated in case there are *incidents*. Incidents are events that cannot be anticipated in advance and hence require replanning. Examples are communication failure, malfunctioning vehicles or temporary traffic jams that could not have been anticipated. The importance of this research is evident, for example, in Autonomous Guided Vehicles (AGV) systems such as used at European Container Terminals (ECT), where has to be dealt with an ever-growing amount of cargo each day and incidents like malfunctioning vehicle communication hardware. Le-Anh & de Koster (2004) provide an interesting survey on AGV systems.

The model is described in this section, accompanied by a notion for conflicts, i.e. how to deal with these limited capacities. Subsequently, a section is devoted to operational planning methods. These are compared in experiments described in the final section, which shows the performance of the methods in normal circumstances and in incident situations.

## 1.1 Transportation orders

Transportation *orders* (or *tasks*) form the workload for the system. A transportation order $o_j = (f_j, s_j, \tau_j^s, d_j, \tau_j^d) \in O$ is the request to pick up freight $f_j \in F$ with volume $vol(f_j) \in \mathbb{N}$ at a certain source location $s_j$ within a specified time-window $\tau_j^s = [t_{j,1}^s, t_{j,2}^s]$ and to deliver it at a specified destination location $d_j$ within a time-window $\tau_j^d = [t_{j,1}^d, t_{j,2}^d]$. The time required to load and unload freight $f_j \in F$ is $loadingtime(f_j) \in T$ and $unloadingtime(f_j) \in T$ respectively. Time values are continuous, $t_{j,1}^s, t_{j,2}^s, t_{j,1}^d, t_{j,2}^d \in T = \mathbb{R} \cup \{-\infty, \infty\}$, and the time-windows must be meaningful: it is possible to load within the loading time-window, $t_{j,2}^s \geq t_{j,1}^s + loadingtime(f_j)$, it is possible to unload within the unloading time-window, $t_{j,2}^d > t_{j,1}^d + unloadingtime(f_j)$ and (neglecting driving time for the moment) it must be possible to unload after the minimal loading time, $t_{j,2}^d \geq t_{j,1}^s + loadingtime(f_j) + unloadingtime(f_j)$. An infinite time value indicates it does not matter how early ($-\infty$) or how late ($+\infty$) an order is picked up or delivered. Associated with each order $o_j$ there is a reward function $v_j : W \times W \rightarrow \mathbb{R}$, where $W = T \times T$ is the set of time-windows. If the actual pick-up time-window is $\check{\tau}_j^s$ and $\check{\tau}_j^d$ respectively, $v_j(\check{\tau}_j^s, \check{\tau}_j^d)$ is maximised if the order is executed within its time-windows, i.e. $\check{\tau}_j^s \subseteq \tau_j^s \wedge \check{\tau}_j^d \subseteq \tau_j^d$, and will typically be smaller if one or both of the time-windows of the transportation order are violated. Both loading or unloading too early and loading or unloading too late typically decreases the reward the agent receives for executing the transportation order.

## 1.2 Transport network and resources

The *transport network*, or infrastructure, represents how the mobile entities (the *transport resources*) can move around. Following Hatzack & Nebel (2001) we make use of a non-standard graph representation of the transport network using infrastructure resources as nodes. This simplifies, for example, the modelling of intersections. Infrastructure resources represent roads, road segments, crossroads, parking space, etc.

The *transport network* $I = (R, E, cap, dist, maxspd)$ is a quintuple consisting of a set of resources $R$, a directed connectivity relation $E$ (defining which resources are neighbours), a capacity function $cap$, a distance function $dist$ and a maximum allowed speed function $maxspd$.

The set $R = R^{inf} \cup R^{tr}$ of resources is decomposed into the set of *infrastructure* resources $R^{inf}$ and the set of *transport* resources $R^{tr}$. The infrastructure resources represent space that can be occupied by the transport resources. For example, an infrastructure $r \in R^{inf}$ resource can represent a road, a road segment, part of an intersection, a parking space, etcetera. A transport resource $r \in R^{tr}$ represents a mobile entity, e.g. a vehicle that can move around through the transport network. The directed *connectivity* relation $E \subseteq R^{inf} \times R^{inf}$ defines which infrastructure resources a transport resource can traverse to from a given infrastructure resource. The *distance* function $dist : R^{inf} \rightarrow \mathbb{R}_0^+$ gives, for each infrastructure resource $r \in R^{inf}$, the distance $dist(r) \geq 0$ it takes to traverse infrastructure resource $r$.

The remaining capacity and speed functions are overloaded for both types of resources. For all infrastructure resources $r \in R^{inf}$, the *capacity* function $cap : R^{inf} \rightarrow \mathbb{N}$ specifies the number $cap(r)$ of agents that can use resource $r$ simultaneously. In other words, for each infrastructure resource $r \in R^{inf}$, at any point in time $cap(r)$ is the maximum number of agents in resource $r$. The *Speed* function $maxpsd : R^{inf} \rightarrow \mathbb{R}$ specifies that $maxspd(r)$ is the maximum possible driving speed at infrastructure resource $r \in R^{inf}$ irrelevant to which transport resource is traversing resource $r$.

For transport resources, there is a similar *capacity* function $cap : R^{tr} \rightarrow \mathbb{N}$, where $cap(r)$ is the load capacity of resource $r \in R^{tr}$. If $O_j$ is the set of transportation orders currently loaded by transport resource $r_j$, the constraint $\sum_{o \in O_j} f_j \leq cap(r_j)$ – the sum of all loaded freight is smaller than the capacity of the transport resource – always holds.

## 1.3 Transport agents

Although not necessary, in this paper it is assumed that each agent is responsible for a single transport resource (a notational convenience). Each vehicle is controlled by a single agent. Then, each agent $a \in A$ plans a route to execute the transportation orders it has been assigned. Such a route $Rt_a = (r_{a,1}, r_{a,2}, \ldots, r_{a,N_a})$ for agent $a$ through infrastructure $I$ is represented as a sequence of $n$ infrastructure resources such that resources $r_{a,i}$ and $r_{a,i+1}$ are connected to each other, i.e. $(r_{a,i}, r_{a,i+1}) \in E$ for $1 \leq i < N_a$. Accompanying this route $Rt_a$, the schedule $Sd_a$ of agent $a$ provides information on when each of these resources in $Rt_a$ are claimed. Schedule $Sd_a = (t_{a,1}, t_{a,2}, \ldots, t_{a,N_a})$ is a sequence of time points, where $t_{a,i}$ specifies the time agent $a$ claims resource $r_{a,i}$. This implies that agent $a$ uses $r_{a,i}$ during the time-window $[t_{a,i}, t_{a,i+1})$ for $1 \leq i < N_a$ and uses resource $r_{a,N_a}$ during time-window $[t_{a,N_a}, \infty)$. Obviously, at any time, the

route and schedule have the same length, i.e. $\forall a \in A : |Rt_a| = |Sd_a| = N_a$. It is assumed the last resource is claimed as long as the agent lives, e.g. $t_{a,N_a+1} = \infty$.

Furthermore, there is a load function $L_a : O_a \to T$ and an unload function $U_a : O_a \to T$ that, for each order $o \in O_a$ in the set of transportation orders planned for execution by agent $a$, specifies the time at which the pickup and delivery takes place, $L_a(o)$ and $U_a(o)$ respectively.

Agents have to claim the infrastructure resource they occupy. To be able to publicly share their plan information they can also make reservations for future claims. For this purpose, a claim function is defined. Using the plan representation described here, the claim function $claim : A \times T \to R^{inf}$ is defined as $claim(a,t) = r \Leftrightarrow r = r_{a,i} \in Rt_a \wedge t \in [t_{a,i}, t_{a,i+1})$.

## 1.4 Incidents

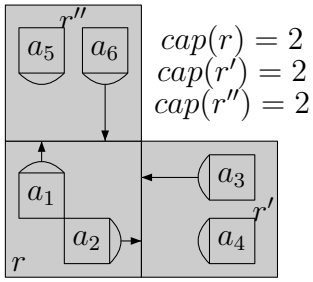
If a communication failure incident occurs, the affected agent is not able to communicate with any other agents during the specified time interval. That means the agent must fall back to simpler planning methods for which it does not need to communicate with others. A communication failure incident $(a, \tau)$ in the set of incidents $\mathcal{I}$ specifies that agent $a \in A$ is not able to communicate with other agents during time-window $\tau \in W$.

Resource failure indicates that a certain resource – either an infrastructure resource or a transport resource – does not function properly during a given interval in time. A resource failure incident $(r, i, \tau)$ in the set of incidents $\mathcal{I}$ is a triple consisting of the resource $r \in R$ the incident operates on, an impact value $0 \leq i < 1$ indicating the severeness of the incident and a time-window $\tau \in W$ within which the incident is effective. If the incident operates on a transport resource, i.e. $r \in R^t$, the vehicle's maximum speed is multiplied with impact $i$ during time-window $\tau$. If the incident operates on an infrastructure resource, $r \in R^i$, the maximum allowed speed of the resource is temporary multiplied with impact $i$. If the resource is a cargo unit (place where the freight is loaded), the maximum loading capacity is multiplied with impact $i$; this only affects new loading operation, not currently loaded transportation orders.

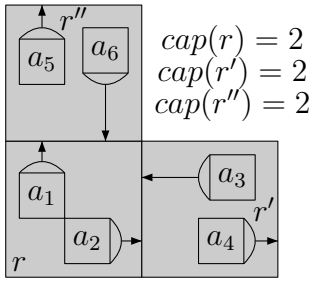
Incidents are often modelled using the Mean Time Between Failure (MTBF) approach. The MTBF time denotes the expected time a resource will function properly. Then, for a certain interval of time – often referred to as the repair time – the resource is malfunctioning. This process can be repeated indefinitely using the MTBF time again to determine the next point in time this resource is malfunctioning.

## 1.5 Feasibility constraints

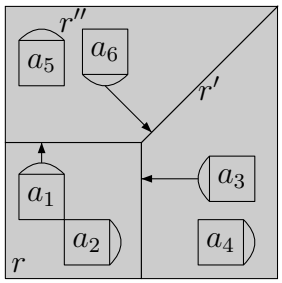
Given the capacity constraints on the transport network, it is clear that at all times no more transport resources can claim an infrastructure resource than the capacity of the infrastructure resource allows. But also, there is an additional constraint that prevents too many agents to swap positions with each other. For example, if two infrastructure resources both have capacity one, at most one agent is allowed to go from the one to the other resource at a certain point in time. Here, these equations are defined and Figure 1 illustrates some examples of what is and what is not allowed. The following help functions $interchange : R^{inf} \times R^{inf} \times T \to \mathbb{B}$ and $stay : R^{inf} \times T \to \mathbb{B}$ are used to specify the constraints:



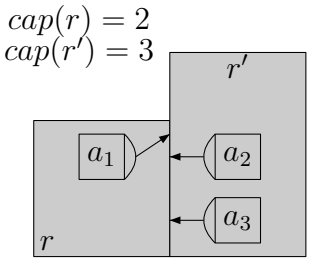

(a) Not allowed: $2 = interchange(r, r', t) \nleq min(2-1, 1) = 1$.




(b) Is allowed: $interchange(r, r', t) = 2$ and $stay(r', t) = 0$. Similar for $(r, r'')$.




(c) Is allowed, also in case both agents $a_2$ and $a_4$ would head to the right.




(d) Not allowed: $3 = interchange(r, r', t) \nleq min(2-0, 3) = 2$.

**Figure 1: These examples illustrate which simultaneous exchanges are allowed by Equation 2. All arrows in these figures indicate the desire of the vehicle to move to the resource the arrow points to. All these movements are instantaneous, assume they are all planned to take place at exactly the same time.**

$$\begin{aligned}
\forall (r, r') \in E, \forall t \in T: \quad & interchange(r, r', t) = \lim_{\epsilon \downarrow 0} \big| \{ a \in A : \\
& \big( (claim(a, t-\epsilon) = r \wedge claim(a, t+\epsilon) = r') \\
& \vee (claim(a, t+\epsilon) = r \wedge claim(a, t-\epsilon) = r') \big) \} \big|, \\
\forall r \in R^{inf}, \forall t \in T: \quad & stay(r, t) = \lim_{\epsilon \downarrow 0} | \{ a \in A : \\
& claim(a, t-\epsilon) = r \wedge claim(a, t+\epsilon) = r \} |.
\end{aligned}$$

On the resource level, at all times the capacity of the resource must be satisfied:

$$\forall r \in R^{inf}, \forall t \in T : |\{a \in A : claim(a, t) = r\}| \leq cap(r). \tag{1}$$

On the edge level, the following safety constraint is defined:

$$\begin{aligned}
\forall (r, r') \in E, \forall t \in T: \quad & interchange(r, r', t) \leq \\
& \min(cap(r) - stay(r, t), cap(r') - stay(r', t))
\end{aligned} \tag{2}$$

A *feasible* plan $P_a$ consists of a route $Rt_a = (r_1, \ldots, r_{N_a})$, a schedule $Sd_a = (t_1, \ldots, t_{N_a})$, load $L_a : O_a \rightarrow T$ and unload $U_a : O_a \rightarrow T$ information for transport resource $v_a \in R^{tr}$, for which the following must hold:

- The route and schedule have the same length, $|Rt_a| = |Sd_a| = N_a$,

- The first resource is claimed at the current time $t$, i.e. $claim(a,t) = r_1 \wedge t_1 \leq t \leq t_2$,
- All infrastructure resources adjacent in route $Rt_a$ must be neighbours in the transport network: $\forall 0 < i < N_a : (r_i, r_{i+1}) \in E$,
- The plan is conflict-free, i.e. Equations 1 and 2 are satisfied,
- All loading actions must be performed in the infrastructure resource that was specified by the customer: $\forall o_i \in O_a, \exists 0 < j < N_a : L_a(o_i) \in [t_j, t_{j+1}) \wedge r_j = s_i$, and likewise for unloading: $\forall o_i \in O_a, \exists 0 < j < N_a : U_a(o_i) \in [t_j, t_{j+1}) \wedge r_j = d_i$, and
- Not a single infrastructure resource is traversed faster than possible, given the speeds of transport resource and infrastructure resource and the incidents and reservations of other agents that are currently known: $\forall 0 < i < N_a : t_{i+1} \geq t_i + traveltime(r_i, v_a, t_i)$.

# 2 Operational planning methods

The planning methods can be categorised as follows. The first method, UNINFORMED-PLANNING, uses simple rules to solve conflicts. Each agent pretends it is alone in the world. Then, using INFORMED-PLANNING, agents take into account the reservations of other agents. Obviously, the first agent that announces its reservations has an advantage over other agents, who need to plan around these reservations. Therefore, a method was developed (REVISING-PRIORITIES) to be able to change reservations of agents done earlier. Finally, a method is considered that also attempts to reroute during the revision of priorities (REVISING-ROUTES), because, if an agent discovers other agents have higher priorities (as determined by the revising priorities planning method) it might prefer a detour.

## 2.1 Uninformed planning

The UNINFORMED-PLANNING method does not take information of the plans of other agents into account. As a result it can easily occur that too many agents plan to reside in the same infrastructure resource at the same time. Therefore, a set of social rules is developed that determines in which order the agents are allowed to enter infrastructure resources. These rules can be compared with (complex) traffic rules.

A distinction is made between *fixed* and *plan-based* social rules. The difference is that in case of *plan-based* rules the priority values depend on the plan of the agent and possibly also the plans of other agents. The following are examples of fixed social rules:

- Randomly assign an entity precedence.
- First-In-First-Out (FIFO): an entity that arrived first takes precedence.
- Longest queue takes precedence: an entity residing in the longest waiting queue takes precedence.

- Longest queue with increment: the same as above, though here starvation is taken into account. It could happen that an agent waits alone for a crossroad where other agents arrive continuously taking precedence. This traffic rule virtually increments the queue length with 1 of each queue that is not chosen as the longest queue.

And these rules are plan-based:

- An agent with the longest plan, or most urgent deadline, takes precedence.
- Each agent can compute the decrease in reward resulting from having to wait. An agent with this maximum decrease takes precedence.
- The previous decrease in reward due to waiting can also be summed for the queue. The queue having the maximum decrease takes precedence.

## 2.2 Informed planning

In the UNINFORMED-PLANNING method, a regular shortest path algorithm can be used. For example, Dijkstra's shortest path algorithm, which has a worst case time complexity of $\mathcal{O}(|R^{inf}|^2)$. The INFORMED-PLANNING method does take reservations of other agents into account. The worst case time complexity then becomes $\mathcal{O}(|R^{tr}|^4 \cdot |R^{inf}|^2)$, see Qiu & Hsu (1999). Notice that shortest path routing with reservations can be seen as three-dimensional shortest path routing, where the new axis is time and reservations temporary disable an infrastructure resource along this axis (the idea is to reach the desired $(x, y)$-destination with an as low as possible $z$-value).

Like the UNINFORMED-PLANNING method, the agent that publicly announces its reservations first, is best off. The other agents are forced to take these extra reservations into account. The next method tries to improve on this.

## 2.3 Revising priorities

In the previous planning method, if agents accepted new orders and changed their reservations accordingly, these reservations were, from then on, kept unchanged. Other agents planned around these reservations. In many cases, however, the welfare of the system can be further improved by reconsidering the potential conflicts and solving them according to the active rules.

The method described here is a modification of Hatzack & Nebel (2001), who suggested that conflict-free routing for a set of vehicles can be transformed to Job-shop scheduling with blocking. The advantage of this transformation is that heuristics that are known to produce good results for Job-shop scheduling can now be used here as well (although Job-shop scheduling with blocking is not a frequently occurring variant of Job-shop scheduling).

The REVISING-PRIORITIES planning method takes two parameters: an *agent selection* heuristic and a *resource block-size*. The agent selection heuristic determines which agent is to schedule first. Possible candidates for this heuristic are the same as those that were described for UNINFORMED-PLANNING in Section 2.1.

1: **function** REVISE_PRIORITIES($a \in A', agentsel, blocksize$)
2: **Pre:** Revising priorities for agent in $a$.
3: **Post:** Agent $a$ modified its schedule.
4: AnnouncePresence($a$)
5: $Sd_a \leftarrow nil$
6: **while** $|Sd_a| < |Rt_a|$ **do**
7: AnnounceValue($a, agentsel(Rt_a, Sd_a, L_a, U_a)$)
8: $V \leftarrow CollectValues()$
9: **if** agent $a$ has minimal value in $V$ **then**
10: $Sd_a \leftarrow Schedule(Rt_a, Sd_a, L_a, U_a, blocksize)$
11: ▷ Agent $a$ just scheduled the next $blocksize$ resources of its route.
12: **end if**
13: **end while**
14: Wait until all agents have a schedule
15: **end function**

First, the group of agents that start such a replanning round (not necessarily all agents) throw away their schedules. The agent selection heuristic determines the order in which order the agents schedule a part ($blocksize$) of their plan. For the agent selection heuristic, the same functions can be used as used for the rules given above. The second parameter, the resource block size $blocksize$, sets the granularity of the replanning. The smaller it is, the more iterations are needed by the algorithm, hence the more computation time. On the other hand, a smaller value perhaps slightly increases the performance.

### 2.4 Revising routes

In the previous methods the routes of an agent were only modified in case a new transportation order was assigned to the agent, or when an incident rendered the route of the agent infeasible. A natural improvement to REVISING-PRIORITIES, is to revise routes as well. The method in the previous section can easily be adapted by adding a line right before Line 7, where the agent announces its priority value. Here, the agent recomputes its shortest path taking into account the current reservations of other agents.

Intuitively, this method will perform better, because, suppose that a certain route has been chosen by an agent and this agent is selected only late to schedule its route, then the agent might have been better off having chosen another route due to conflicts with the agents who already make reservations.

## 3 Experiments with TRAPLAS

The experiments described in this section are performed using a home-made open-source transport planning simulator called TRAPLAS, see Zutt (2006). The experiments are conducted on 8x8 grid-like networks[1] with 64 infrastructure resources, 128 arcs and 32 transport resources. Ten instances with the lowest workload (160 orders) are generated in such a way that it is possible to execute all these orders without violating any

[1]The networks are not exactly grid networks, because several random diagonal arcs are added. This is done to be able to create a grid-like network that has the same number of resources and arcs as an instance of some other network topology.

time-windows, assuming there are no incidents. The bigger instances are constructed by merging these instances together. Because of this approach, an upper-bound is known on the optimal performance.

The level of incidents is increased by increasing the failure probability of each resource. For instance, if the failure probability of a resource is 0.1, the incident generator ensures this resource is malfunctioning about 10 percent of the time.

## 3.1 Hypotheses

As stated before, the goal of these experiments is to *(i)* discover what information is required to obtain good performance by comparing the different planning methods and *(ii)* to test the planning methods in incident situations, with increasing workload and increasing incident level. The experiments in this section are set-up in an attempt to verify or falsify the following hypotheses.

H1. Adding knowledge (e.g. taking into account reservations of other agents, use a clever agent selection heuristic) significantly improves performance at the cost of some extra computation time.

H2. The performance of the planning methods will strongly depend on the agent selection function chosen.

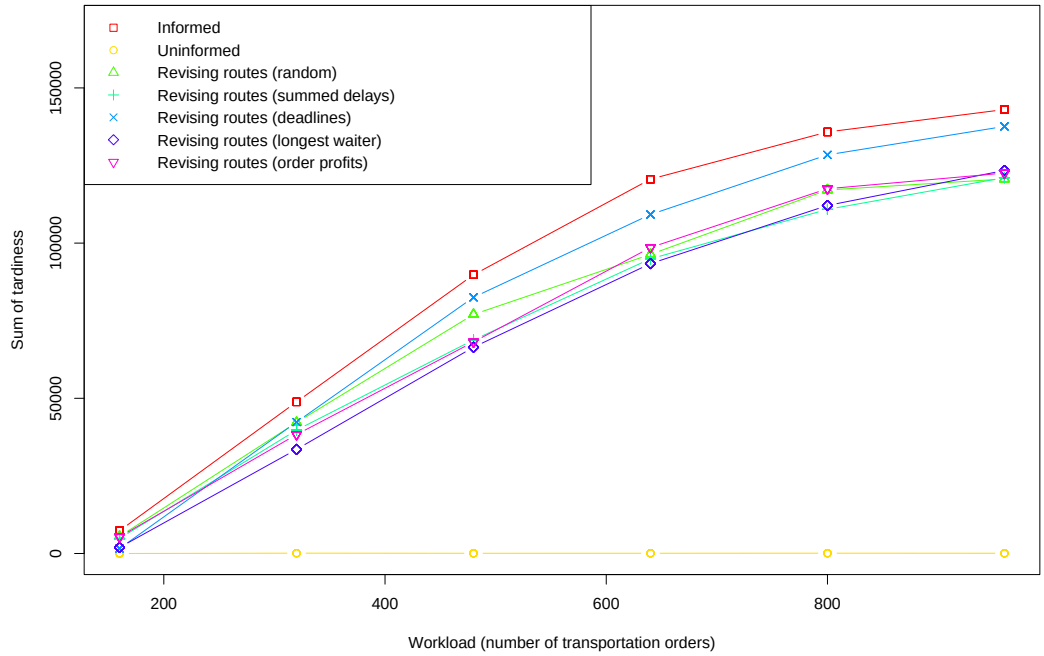
H3. REVISING-ROUTES is in all cases the best planning method. In case of incidents, its performance degrades less as compared to the other methods.

## 3.2 Evaluation

Figure 2 shows the performance, measured as the sum of pickup and delivery tardiness over all transportation orders, for the different planning methods when the workload is being increased. It can be seen that REVISING-ROUTES with the "longest waiter" agent selection function outperforms the other planning methods. The attached ANOVA table proves that the difference of the plotted means is significant. Furthermore, the difference slightly increases when the workload of the agents is increased, up to a certain point. The ANOVA table supports this interaction effect. Figure 4 clearly shows that when more information is taken into account during planning, this results in more computation time required by the planning method. All of the presented methods seem to have a computation time linear in the workload.

In Figure 3, another performance indicator, viz. the system welfare, is displayed. Again, it can be seen that REVISING-ROUTES – for some choices of agent selection heuristics – outperforms INFORMED-PLANNING. UNINFORMED-PLANNING turns out to be useless in all cases, because, when agents do not take into account reservations of other agents, the system is very deadlock prone.

Figure 5 illustrates the robustness of the different planning methods when the incident level is increased. The incident level is measured as the severeness of the incident multiplied by the length of the time-window in which the incident is effective, i.e. $\sum_{(r,i,\tau)\in\mathcal{I}}(1-i)\cdot length(\tau)$. The best planning method is again REVISING-ROUTES, but there does not seem to be a significant difference in performance degradation between these planning methods.




| *ANOVA table* | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| planning method | 6 | 3.3185e+11 | 5.5308e+10 | 46.179 | 3.020e-13 |
| workload | 1 | 5.7141e+11 | 5.7141e+11 | 477.094 | < 2.2e-16 |
| interaction | 6 | 9.8891e+10 | 1.6482e+10 | 13.761 | 3.123e-07 |
| residuals | 28 | 3.3535e+10 | 1.1977e+09 | | |

**Figure 2: Sum of pickup and delivery tardiness over all transportation orders by the different planning methods under normal circumstances while increasing the workload of the 32 agents.**

# 4 Conclusions and future work

This paper described a resource-based model for pickup and delivery transport planning supporting transport networks with limited capacities. After briefly summarising several planning methods experimental results were presented that were obtained using the simulation tool TRAPLAS.

It was shown that the more sophisticated planning methods performed better than the more simplistic methods. Information about plans of other agents and clever heuristics indeed improve the overall performance. Some further work is required to give more information on the performance degradation of the different categories of planning methods described in this paper. Also, a method is to be added that intentionally inserts slack into the agents plans.

Currently, other network topologies like completely random networks, small-world or scale-free and some realistic networks are used in similar experiments. It is hypothesised that performance degradation due to incidents depends on the transport network topology.

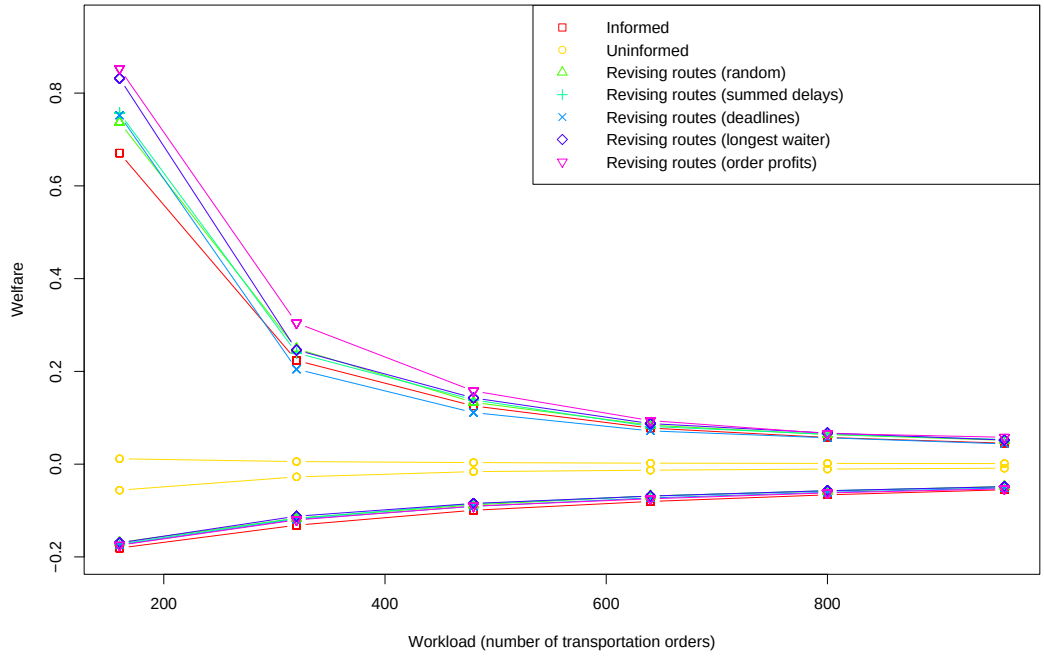



**Figure 3: The system welfare (order rewards minus transport resource costs) divided by an upper-bound on the optimal system welfare. One needs to subtract the costs (the negative plots) from the rewards (the positive plots) to obtain the actual relative system welfare.**

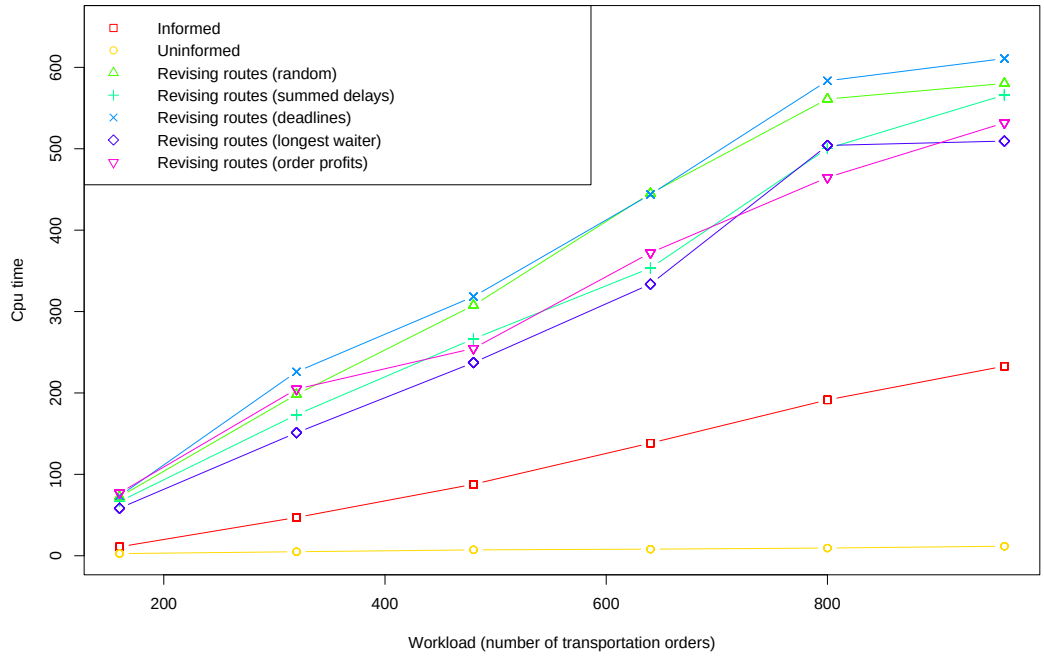



**Figure 4: The CPU-time [seconds] that was needed to do a complete simulation run.**

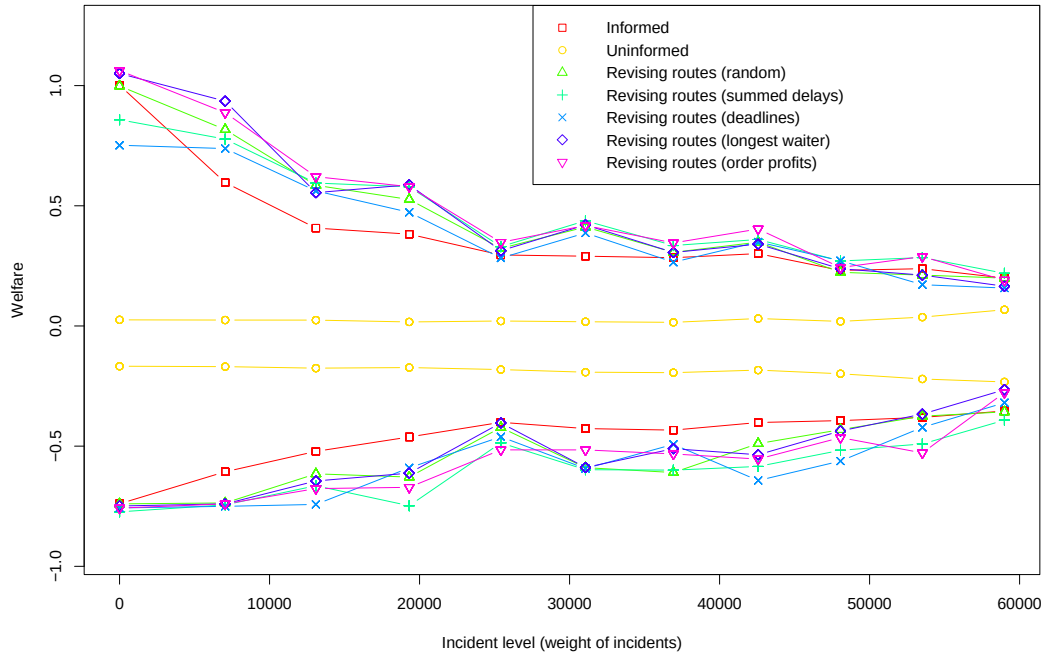



**Figure 5: Comparison of the robustness of the planning methods by increasing the level of incidents (malfunctioning infrastructure and transport resources).**

# Acknowledgements

The author is member of the Algorithms group in the Department of Software Technology at the faculty EEMCS of Delft University of Technology.

This research, which is supervised by Cees Witteveen, has been supported by the TNO-TRAIL project (16) *Fault detection and recovery in multi-modal transportation networks with autonomous mobile actors* and by the Dutch Ministry of Economic affairs under the SENTER TSIT program *Cybernetic Incident Management* (TSIT2021).

# References

Hatzack, W., B. Nebel (2001) Solving the operational traffic control problem, in: Cesta, A., ed., *Proceedings of the 6th European Conference on Planning (ECP'01)*.

Le-Anh, T., M. R. de Koster (2004) A review of design and control of automated guided vehicle systems, Tech. rep., Erasmus Research Institute of Management, eRIM Report Series Reference No. 2004-030-LIS.

Qiu, L., W. Hsu (1999) Scheduling and routing algorithms for agvs: a survey, Tech. rep., Centre for Advanced Information Systems, School of Applied Science, Nanyang Technological University, Singapore.

Zutt, J. (2006) Transport planning simulator (TRAPLAS), URL `http://traplas.sourceforge.net`.