Multi-Agent Transport Planning

Jonne Zutt[acd] Leon Aronson[ac] Roman van der Krogt[ac]

Nico Roos[b] Cees Witteveen[a]

[a] Delft University of Technology, ITS, P.O. Box 5031, 2600 GA Delft.
[b] Universitity of Maastricht, IKAT, P.O. Box 616, 6200 MD Maastricht.
[c] Supported by the Freight Transport Automation and Multimodality (FTAM) research program, carried out within the TRAIL research school for Transport, Infrastructure and Logistics.
[d] This research has been supported by the TNO-TRAIL project (16) *Fault detection and recovery in multimodal transportation networks with autonomous mobile actors*

Abstract

This paper describes the operational planning of complex multi-agent transportation problems. Based on a simple hierarchical agent-based model, agents execute transportation orders specified by clients. Hatzack and Nebel [3] pointed out a correspondence with job shop scheduling. Using this correspondence they are able to use scheduling heuristics to compute safe traffic flows, using a fixed route for each transportation agent. We adopt their approach and show a significant improvement in case alternative routes are considered by the agents.

1 Introduction

This paper focuses on the *operational planning* of complex multi-agent transportation problems. Such transportation problems consist of a *(i) static infrastructure*, *(ii) transport agents* and customers with *(iii) transportation orders*. The infrastructure is specified by a set of locations, roads between them, global travel times, and distances on the roads. The transportation agents are autonomous agents capable of moving cargo from one place to another place. The customers are able to place 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 properties of their orders in advance. The following properties of an order are distinguished: *(i)* its pickup and delivery location, *(ii)* its pickup and delivery time-windows, *(iii)* its revenues if it is executed within the time windows and *(iv)* its penalties if it is rejected or a transport agent fails to execute it within the time windows specified.

The problem now is to construct a dynamic planner that is able to find a feasible and robust plan for handling the orders and also is able to react on incidents (changes of orders, failures of agents) to minimize their impact on the total performance[1]. Here, the performance of the planner is specified as the sum of the costs the agents will make to execute all the orders received. The cost of an individual agent executing a series of orders will depend on the total distance traveled, the time needed to travel and the revenues/penalties associated with the orders executed. An optimal planner, of course, will try to minimize the total costs associated with executing the orders received.

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

1.1 Approach

Even much simpler problems than this problem are easily shown to be intractable [2], i.e., assuming P $\neq$ NP, they are not optimally solvable in an efficient way. We have chosen to search for clever heuristics, i.e., problem solving methods that produce feasible, not necessarily optimal, solutions.

We model the described planning problem using a simple *hierarchical agent-based* model. On the highest level, we have one or more *tactical planning* agents[2] responsible for taking care of incoming orders and allocating them to individual transportation agents. The transportation agents take care for the *operational planning* and order execution using the infrastructure and handling incidents. Such an transport agent / operational planner may be a truck driver or a computer program running on an AGV (Automated Guided Vehicle). Each operational planner can only see that part of the plan it is allocated to by the tactical planner.

The operational planner is also responsible for communicating any (expected) deviations from the orders as they are assigned to the tactical planner, such that the tactical planner can adequately adjust plans when necessary. Since operational planners operate on an infrastructure, it 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.

As we remarked before, both planners together aim at finding a solution minimizing the total cost of executing the orders. The advantage of the hierarchical approach is that the partition into tactical and operational planning aspects nicely corresponds to the two main factors determining the total performance of the planning system: a distribution of orders over agents such that the total amount of profit will be *maximized* and the execution of each order by an agent such that its execution costs are *minimized*.

1.2 Applications

There is a large variety of transport problems fitting into this simple model. All these problems are order-based, have pickup and delivery specifications, and consider autonomous transportation agents. We have explicitly accounted for the following problems.

- Taxi planning: here customers are passengers specifying a pickup time and location, and a delivery location. The taxi company has a number of taxis (transport agents) available. The goal is to minimize both travel costs and customer waiting times.

- AGV planning: 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 AGVs themselves.

- Arrival and departure scheduling: this is an example where customers are identical to transportation agents. Each agent as an airplane specifies an arrival and departure time window for a given airport. The tactical planning agents are the runway managers of an airport assigning time-slots for arrival and departure on a given runway, trying to satisfy the time-windows specified. The goal is to maximize throughput and minimize waiting times of airplanes.

[2] A tactical planner is a facility of a collection of agents and can be implemented by either a human or an automated process.

Elsewhere, we have discussed more details of the hierarchical planner architecture [1]. In this paper we will concentrate on the operational planning details, discussing some heuristics the agents might use to come up with a feasible plan.

2 Operational Planning

A transportation agent a has to execute the orders assigned to it by the tactical planner, using an infrastructure. This infrastructure $I = (R, A)$ consists of a set of resources R – representing locations, roads and road crossings – and a set A that defines the accessibility relation, e.g. $(\rho_1, \rho_2) \in A$ denotes that resource ρ_2 is accessible from resource ρ_1. Further, there is a set $H \subseteq R$ of home locations, which are special resources with unlimited capacity. To each agent a, one of these locations $h_a \in H$ is assigned denoting its home (initial) location [3]. For each resource $r \in R$, the cost function $C : R \to \mathbb{N}$ returns the cost[4] $C(r)$, measured in time units, for a transport agent to travel through resource r.

An order $o = (f, s, T_s, d, T_d)$ consists of a source location $s \in R$ in which a freight f must be picked-up and a destination location $d \in R$ where f is to be delivered. Both the pick-up and delivery must take place within time interval $T_s = [lt_s, ut_s]$ and $T_d = [lt_d, ut_d]$ respectively[5]. The deadline ut_d of the order o is also denoted by ϕ_o.

In order to execute order o assigned to it, agent a has to find a route from its current location to the pickup location and from there to the destination. Such a route $Rt_a = ((\rho_1, t_1), (\rho_2, t_2), \ldots)$ of a – where $t_i \leq t_{i+1}$ and $(\rho_i, \rho_{i+1}) \in A$ – is specified as a sequence of resource-time pairs, where t_i denotes the time at which a enters the resource[6] $\rho_i \in R$.

Since the agents use a common infrastructure, resource conflicts are likely to occur. Such a conflict is said to occur if at least two agents use the same resource at overlapping time intervals. To solve conflicts agents can either change their routes or insert delays[7] in their routes to avoid overlapping intervals.

The result of inserting delays, such that conflicts are resolved, into agent a's route Rt_a is called a *schedule* $Sd_a = ((\rho_1, t'_1), (\rho_2, t'_2), \ldots)$, where $t'_i \leq t'_{i+1}$.

In the absence of other agents, an agent a is able to select an *optimal route* $OptRt_a$ guaranteeing the earliest start and completion time for each order. With respect to this optimal route, the delay of orders in a schedule can be defined. The delay at the i-th resource used in schedule Sd_a is defined as:

$$delay(OptRt_a, Sd_a, i) = \begin{cases} t(Sd_a, 1) - t(OptRt_a, 1) & i = 1 \\ \big[t(Sd_a, i) - t(OptRt_a, i)\big] - & \\ \quad \big[t(Sd_a, i-1) - t(OptRt_a, i-1)\big] & i > 1, \end{cases} \tag{1}$$

where function $t(Sd, i)$ selects the time of schedule Sd at index i.

We define the earliest completion time M_o for order o to be the time at which the order o is executed by the responsible agent a in its optimal route $OptRt_a$ and C_o the real completion time (as derived from schedule Sd_a).

---

[3] Home locations are introduced to avoid deadlocks due to agents that claim certain resources for ever after (or before) they executed all their transportation orders.

[4] For simplicity, we assume that this cost is the same for every agent a.

[5] We assume that $lt_d \leq lt_s + mp$ and $ut_d \geq ut_s + mp$, where mp is the minimal processing time for the order. That is, in principle, the execution of an order can start on any time point in T_s.

[6] Note that ρ_i is released by a at time t_{i+1}, so $[t_i, t_{i+1})$ is the interval during ρ_i is claimed by a.

[7] Of course, assuming a finite number of agents, such delayed routes avoiding conflicts always exist.

The task of each operational planner is to find a schedule for its order that minimizes the order's tardiness.

2.1 Resolving resource conflicts

Hatzack and Nebel [3] pointed out a correspondence between computing conflict-free schedules for the transport agents and a particular scheduling variant called *Job Shop Scheduling with Blocking*. In their paper they subsequently used a fast job-shop scheduling heuristic to obtain conflict-free transport routing schedules. In this section, we briefly explicate this correspondence.

Scheduling is concerned with the optimal allocation of a set R of scarce resources to a set of activities (jobs) J over time. Each job $j \in J$ requires some specific set $R_j \subseteq R$ of resources and for each resource $r \in R_j$ the duration $t_{r,j}$ needed for j to use r is specified. Typically, each resource r can be used only by one job j at the same time.

A solution to such a problem is a *schedule*, i.e., an allocation of intervals $[\sigma_{i,k}, \phi_{i,k}]$ to each job $j_i \in J$ for using resource r_k such that the constraints (non-overlapping and minimal duration) are satisfied.

In a *job shop* scheduling problem each job j_i consists of a *sequence* of k_i operations $(o_{i,1}, \ldots, o_{i,k_i})$, where operation $o_{i,j}$ needs resource $r_{i,j} \in R$ for $p_{i,j}$ time units, with $r_{i,j} \neq r_{i,j+1}$ for $i = 1, \ldots, k_{j-1}$. *Blocking* means that a job j continues to claim resource r after processing, if the next resource r' it needs is not available. During that time, no other job can use resource r.

Now the similarity with the transportation problem discussed above is clear: Let the jobs correspond to agents $a \in A$ that have to execute a route Rt_a as a sequence of operations. Each operation (ρ_i, t_i) in fact is a request for using[8] the resource ρ_i during the time interval $[t_i, t_{i+1})$.

A feasible conflict-free schedule is a *set of agent schedules* $\{Sd_a\}_{a \in R}$ that is conflict-free. Scheduling heuristics for job-ship scheduling problems, therefore, can be used to compute agent schedules in which resources are claimed by at most one agent at a time thereby avoiding any resource conflicts from a given set $\{Rt_a\}_{a \in R}$ of agent routes.

In their paper Hatzack and Nebel applied a very fast delay minimizing heuristic[9] to obtain such a set of agent schedules. This heuristic will be used as part of a more refined routing heuristic for our operational planning problem.

2.2 An operational multi-agent planner

Unlike Hatzack and Nebel, we have to account for the time windows T_s and T_d and our objective is to minimize the *tardiness* summed over all agents. Moreover, we assume our agents to be intelligent enough to apply rerouting if the delay caused by resource conflict resolution becomes too costly. Therefore, we will discuss a refined heuristic that is capable to deal with *(i)* time windows, *(i)* rerouting as well as delaying routes and *(iii)* tries to minimize tardiness. We describe this heuristic in a distributed multi-agent setting.

Upon receiving an order o each agent a executes Algorithm 1. First, (line 3) the agent computes the shortest path along the source and destination locations of the newly arrived order. Then, (line 4), it creates a schedule – using the scheduling heuristic – based on this route. It negotiates with other agents about the order in which the agents may compute

[8] we assume the minimum time needed for a to use ρ_i to be given.

[9] This heuristic incrementally inserts jobs/operations in a first-come-first served manner into the schedule.

their schedule. This order is determined by the difference of their deadlines and earliest possible completion times $\phi_o - M_o$. Next, when all agents have computed a schedule, some agents have a possibility to change their route. Agents are only allowed to change their route if they changed it less than $max\_alternatives$ times already and their schedule still violates the order's deadline. The agent that is allowed to change its route tries to find the shortest route that avoids the resource that caused the biggest delay[10]. After an agent changes its route all agents reschedule.

Remark To determine which agent is to schedule (or reroute) first, the agents have to negotiate with each other. This can be implemented using a *blackboard*. In Algorithm 1 this is specified by the $Announce...Value$ and $CollectValues$ functions. The $Announce...Value$ functions put the value of an agent on the (sorted) blackboard, while the $CollectValues$ function either returns all values of all agents or at least some indication whether the agent is at the top of the list. Further, an agent can store on the blackboard whether it is satisfied or not ($Announce(Un)Satisfied$, in other words, whether the agent meets the order's deadline) and whether it has computed a schedule ($AnnounceHasSchedule$).

In Algorithm 2, agent a creates a schedule based on its route Rt_a and using information obtained from the infrastructure. Given a partial schedule $Sd_a = \ldots(\rho_{a,i-1}, t_{i_1})^\frown \ldots$ it schedules the next resource as given in route Rt_a and then recursively schedules the rest of the route. Time t_{i_1} is the time at which the previous resource $\rho_{a,i-1}$ is claimed. Time t is at least $C(\rho_{a,i-1})$ later (the minimal time at which resource $\rho_{a,i}$ can possibly be claimed). The result is the schedule $Sd_a = \ldots(\rho_{a,i-1}, t_{i_1})^\frown(\rho_{a,i}, t)^\frown Schedule(a, \mathrm{tl}(R)t_a)$.

Sometimes, the operational planner does not find a feasible solution that executes the order within its time-windows. Algorithm 2 does not deal with this problem. It is the responsibility of the tactical planner, so, if such a problem arises, the tactical planner decides whether the transport agent should execute the order anyway or cancel the transportation.

3 Results

In this section we show experimental results of using the scheduling heuristic to compute a safe traffic flow, while allowing each agent a fixed number of time to select an alternative route. The infrastructure used for the experiments is displayed in Figure 1. This hub-and-spoke alike infrastructure consists of a set of four completely connected cities of five locations each and a few inter-city connections. Each city has one home location, viz. the one without connections to other cities. In each home location there are five transport agents, which yields a total of twenty agents in each problem instance. The agents have to carry out one order with a randomly chosen source and destination location. The time-windows are fixed and the same for all transportation orders.

Figures 2(a) and 2(b) show statistical data about the experiments. Figure 2(a) gives information about the tardiness summed over all agents, which is defined as

$$\tau_{\mathcal{A}} = \sum_{a \in \mathcal{A}} \begin{cases} 0 & \text{if } C_o < \phi_o \\ C_o - \phi_o & \text{otherwise,} \end{cases} \tag{2}$$

[10]The route remains unchanged if this resource is obligatory for the agent to reach its goal.

Algorithm 1 process(Agent a, Order $o = (s, T_s, d, T_d)$)

L1. $no\_alternatives \leftarrow 0$

L2. $AnnouncePresence(a)$

L3. $Rt_a \leftarrow ShortestPath(h_a, s) ^\frown ShortestPath(s, d) ^\frown ShortestPath(d, h_a)$

L4. **while** $Sd_a = nil$ **do**

  L4.1. $AnnounceSlackValue(a, \frac{\phi_o - M_o}{M_o})$

  L4.2. $V \leftarrow CollectValues()$

  L4.3. **if** *agent a has minimal value in V* **then**

    $Sd_a \leftarrow Schedule(a, Rt_a)$

    $AnnounceHasSchedule(a)$

    if $C_o \leq \phi_o$ **then**

      $AnnounceSatisfied(a)$

    else

      $AnnounceUnsatisfied(a)$

L5. *Wait until all agents have a schedule*

L6. **while** $\exists b \in \mathcal{A} : UnsatisfiedAgent(b)$ **do**

  L6.1. *// determine which agent changes its route*

  L6.2. **if** $no\_alternatives \geq max\_alternatives$ **then**

    $AnnounceSatisfied(a)$

  L6.3. **elseif** $C_o > \phi_o$ **then**

    $AnnounceUnsatisfied(a)$

    $AnnounceTardinessValue(a, C_o - \phi_o)$

    $V \leftarrow CollectValues()$

    if *agent a has maximal value in V* **then**

      $forbidden \leftarrow \mathrm{argmax}_{i \in Sd_a}\, delay(Rt_a, Sd_a, i)$

      $reroute(Rt_a, forbidden)$

      increase $no\_alternatives$

  L6.4. *// reschedule all agents*

  L6.5. **while** $Sd_a = nil$ **do**

    $AnnounceSlackValue(a, \frac{\phi_o - M_o}{M_o})$

    $V \leftarrow CollectValues()$

    if *agent a has minimal value in V* **then**

      $Sd_a \leftarrow Schedule(a, Rt_a)$

      $AnnounceHasSchedule(a)$

      if $C_o \leq \phi_o$ **then**

        $AnnounceSatisfied(a)$

      else

        $AnnounceUnsatisfied(a)$

  L6.6. *Wait until all agents have a schedule*

where ϕ_o is the deadline of agent a's order and C_o the time at which agent a really completes the order. The average percentage of delay, as shown in Figure 2(b), is defined as

$$\delta_{\mathcal{A}} = \sum_{a \in \mathcal{A}} \frac{C_o - M_o}{C_o} \quad / \quad |\mathcal{A}|. \tag{3}$$

Algorithm 2 Schedule(Agent a, Route Rt_a)

L1. $(\rho_{a,i}, t_1) = \mathrm{hd}(Rt_a)$
L2. **if** $Sd_a = nil$ **then**
    $t2 \leftarrow t1$
L3. **else**
    $(\rho_{a,i-1}, t_1) = \mathrm{last}(Sd_a)$
    $t_2 \leftarrow t_1 + c(\rho_{a,i-1})$
L4. $t \leftarrow \min_{t \in \mathbb{N}} t \geq t_2 \wedge NotClaimed(\rho_{a,i-1}, [t_1, t))$
L5. **while** $\neg scheduled$ **do**
        $Sd_a \leftarrow Sd_a \frown (\rho_{a,i}, t)$
        if $Schedule(a, \mathrm{tl}(Rt_a))$ **then**
          $scheduled \leftarrow true$
        else
          Retract last scheduled resource from Sd_a
          $t \leftarrow t + 1$

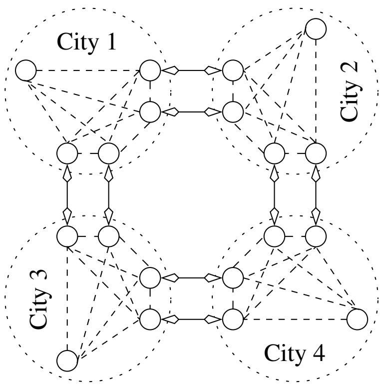
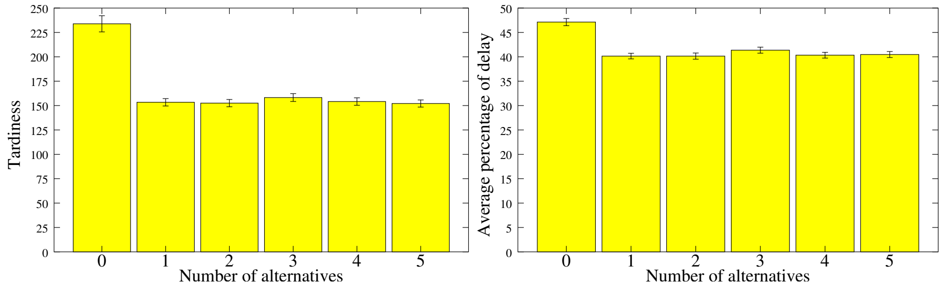


Figure 1: The infrastructure used to gather experimental results.

In case the agents are not allowed to select an alternative route (the leftmost bar in both figures) each transport agents drives that route that would have been the optimal route if there were no other agents. The scheduling approach of Hatzack and Nebel is used to create routes without conflicts considering all agents in the system.

The second bar shows a significant improvement if each agent has at most one opportunity to change its route. The mean tardiness summed over all agents decreases from 233.87 to 153.38. Furthermore, the average percentage of delay decreases from 47.12 to 40.15. Note that the time to complete an order o for agent a is determined by the earliest completion time M_o and the delay caused by solving conflicts with other agents. The indication that the tardiness summed over all agents does not further decrease while incrementing the maximum number of alternative routes of the agents leads to our expectation that it will not be possible to find a solution with a significantly lower average delay percentage. A possible explanation is that one change is sufficient to spread the traffic through the (simple) infrastructure and no gain can be achieved by rerouting agents.



(a) Tardiness summed over all agents $\tau_{\mathcal{A}}$.

(b) Average percentage of delay $\delta_{\mathcal{A}}$.

Figure 2: Statistical data of the experiments (averaged over 100 generated problem instances. Error bars indicate 95% confidence intervals).

Further research is needed to clarify this observation.

4 Conclusion

We have described a distributed algorithm for the operational planning of multi-agent transportation problems. Hatzack and Nebel [3] discussed a similar problem, where a fleet of vehicles must execute a set of transportation orders. They discovered a transformation of their transportation problem into a job shop scheduling problem with blocking.

Our problem definition also specifies two time-windows for each order. Each order must be picked up within a time-window and delivered within a time-window. However, the same approach as Hatzack and Nebel used can be applied. Further, we enriched their idea by allowing agents to reroute a fixed number of times. This showed significant improvements already for at most one opportunity to reroute for all agents.

In the future, we plan to execute more experiments in order to discover why there is no significant improvement if there are more than one alternative routes considered by the agents. We need to experiment with different kind of infrastructures and varying number of orders per agent. Furthermore, a comparison with our other operational planner using the D** algorithm [1], which is a variant of the D* algorithm[11] of Stentz [4], is desired.

References

[1] L.D. Aronson, R.P.J. van der Krogt, and J. Zutt. Automated transport planning using agents. In *Freight Transport Automation and Multimodality (FTAM'02)*, 2002.

[2] M.R. Garey and D.S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.

[3] W. Hatzack and B. Nebel. Solving the operational traffic control problem. In A. Cesta, editor, *Proceedings of the 6th European Conference on Planning (ECP'01)*, 2001.

[4] A. Stentz. The focussed d* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[11] D* is short for Dynamic A* and is a dynamic variant of A*, a well-known search algorithm in AI.