Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems

Martijn Mes[*], Matthieu van der Heijden, Aart van Harten

Revised version, 22 December 2005

Abstract

We consider the real-time scheduling of full truckload transportation orders with time windows that arrive during schedule execution. Because a fast scheduling method is required, look-ahead heuristics are traditionally used to solve these kinds of problems. As an alternative, we introduce an agent-based approach where intelligent vehicle agents schedule their own routes. They interact with job agents, who strive for minimum transportation costs, using a Vickrey auction for each incoming order. This approach offers several advantages: it is fast, requires relatively little information and facilitates easy schedule adjustments in reaction to information updates. We compare the agent-based approach to more traditional hierarchical heuristics in an extensive simulation experiment. We find that a properly designed multi-agent approach performs as good as or even better than traditional methods. Particularly, the multi-agent approach yields less empty miles and a more stable service level.

Keywords: Transportation; Multi-agent systems; Auctions/bidding

1 Introduction

For operational planning and control of many transportation networks it is important to deal with uncertainties like transportation times (e.g. due to congestion), arrival of rush orders during schedule execution, and order modifications. In combination with sometimes tight restrictions (e.g. time windows) this leads to the need for a flexible, stable and robust planning and control

[*]Corresponding author. Address: Department of Operational Methods for Production and Logistics, Faculty of Business, Public Administration and Technology, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands; phone +31-53-489-4062; fax: +31-53-489-2159; e-mail: m.r.k.mes@utwente.nl.

system. It should be *flexible* in the sense that schedule adjustments in reaction to information updates should be easy. It should be *stable* in the sense that minor information updates (e.g. the arrival of a single rush order) should have impact on a small part of the schedule only. It should be *robust* in the sense that the overall network performance (e.g. transportation costs, on-time delivery performance) should remain acceptable under a large number of scenarios for unexpected events like rush orders.

Traditionally, operations research (OR) based global optimization methods are used to construct integral transport schedules. However, one may wonder whether such methods are most suitable for planning and control of stochastic, dynamic transportation networks. Firstly, most optimization algorithms require a lot of information in advance. Secondly, global optimization algorithms can be sensitive to information updates: a minor modification in information may have impact on the schedules of many vehicles. Thirdly, the time required for the algorithm may not permit timely response to unexpected events such as equipment failure and the arrival of rush orders. Finally, flexible transportation networks may consist of multiple independent organizational units that are working in an autonomous, self-interested and not necessarily cooperative way. Therefore, these individual players may not be willing to share all their information (like their cost structure, current vehicle locations and current schedule), so that traditional centralized or hierarchical approaches are not applicable anymore.

An alternative that has been proposed within the computer science literature is the multi-agent system (MAS). Such a system consists of independent intelligent control units linked to physical or functional entities (e.g. vehicle, order). It seems to be a promising solution for controlling complex networks, providing more flexibility, reliability, adaptability and reconfigurability. Agents act autonomously by pursuing their own interest and interact with each other, for example using information exchange and negotiation mechanisms. In a transportation network, each order (job) and each resource can have its own goal-directed agent. For example, a job agent may focus on on-time delivery against the lowest possible costs, and a resource agent may strive for utilization and/or profit maximization. A key issue is how to configure agents such that their self-interested behavior yields a near-optimal solution for the network as a whole. One option is to use a market mechanism like an auction. An overall goal for the network performance can be to balance the total lateness and the total relevant costs.

The principle of multi-agent systems is elegant and has clear advantages from an ICT point of view. However, it is unclear whether the system-wide performance will be similar to or even better than the performance of more centralized or hierarchically organized planning systems. It is even not guaranteed whether and when a multi-agent system will show a stable behavior. That is, will all orders be transported, will resources properly be utilized and will prices remain within reasonable bounds in the absence of a coordination mechanism?

Although many papers have appeared on multi-agents systems, also applied to logistics, literature on the performance comparison between traditional OR-based systems and multi-agents systems is scarce. In this paper, we aim to make such a comparison for a transportation network where orders (full truck loads) with varying soft time windows arrive during schedule execution and should be scheduled in real time. That is, an order should be assigned to a vehicle and a feasible start time should be determined. Because a fast response is required, we use local dispatch rules and serial scheduling as benchmarks, see Heijden, Gademan and van Harten (2002) and Ebben, van der Heijden and van Harten (2005). For the multi-agent system, we develop an auction mechanism with several pricing variants. To compare the agent-based approach with the two more traditional approaches, we use discrete event simulation for an extensive numerical experiment. As overall network performance criteria we focus on the average on-time delivery percentage as *service* measure, variation in the on-time delivery percentage as *robustness* measure and the empty mile percentage as *efficiency* measure. We also use total costs (transportation costs, penalty costs) to measures a combination of service and efficiency.

The remainder of this paper is structured as follows. In the next section, we give an overview of related literature and we explain the contribution of our paper. In Section 3, we present our model and in Section 4 we discuss our choice for a particular agent based planning concept. Next, we discuss several options for agent bidding and bid evaluation in Section 5. In Section 6, we briefly present the two more traditional planning approaches that we use as benchmarks in a simulation study. We describe the experimental settings in Section 7 and provide the numerical results from this study in Section 8. We end up with conclusions, remarks on generalizations and directions for further research (Section 9).

2 Related literature

2.1 Transport planning

Our problem of assigning jobs to vehicles in a transportation network is well-known in the area of vehicle routing problems (VRP) as a real-time multi-vehicle pickup and delivery problem with time windows. This problem type is also known as the dial-a-ride problem. Such problems arise a.o. in the transportation of elderly and/or disabled persons, shared taxi services, certain courier services and so on. We consider a variant with full truckloads, stochastic arrival of orders and stochastic handling- and travel times, where even the probability distributions are not known in advance.

The VRP and its variants have been studied extensively; see Laporte (1992) and Toth and Vigo (2002) for a survey. It is well-known that most variants of the VRP problem are NP-hard, so that it is virtually impossible to find an optimal solution within a short time. Most work focuses on static and deterministic problems where all information is known when the schedule has to be generated, see for example Desrosiers et al. (1995). When the input data (travel times, demands) are stochastic and depend on time, the planning result is not a set of routes but rather a policy that prescribes how the routes should evolve as a function of those inputs that evolve in real-time (Psaraftis 1988). Such policies have been studied in several papers, see for example Psaraftis (1988) and Gendreau and Potvin (1998). The most common approach to handle these problems is to solve a model using the data that are known at a certain point in time, and to reoptimize as new data become available. Because a fast response is required, it is common to use relatively simple heuristics or parallel computation methods, see Giani et al. (2003) for an overview.

The dynamic assignment problem as discussed in (Godfrey and Powell 2002) also shows some similarities. Here resources (e.g. vehicles) are also dynamically assigned to tasks that arrive during schedule execution. Key differences are (1) each individual vehicle schedule contains only one order at a time (2) the price of an order is exogenous and the only issue is whether to accept this order and if so, to assign a vehicle to this order (3) only the most profitable orders are accepted . In (Powell and Carvalho 1998), they use so-called Logistics Queuing Networks (LQN) to decompose the large and complex scheduling problem by a series of very small problems. In this way, many real world details can be included in the model that cannot be dealt with using traditional approaches. Still this is a centralized planning approach in contrast to the decentralized agent-based approach that we consider in this paper.

Closely related work can be found in Regan et al. (1995, 1996, 1998) who investigate the dynamic assignment of vehicles to loads for real-time truckload pickup and delivery problems. They provide relatively simple and fast local rules. Yang, Jaillet and Mahmassani (2004) extend this work to a formal optimization-based approach for the same problem class. They use simulation to compare this approach with the previously developed heuristics. Mahmassani, Kim and Jaillet (2000) present a hybrid approach combining fast heuristics for initial assignment with the optimization-based approach for the off-line problem of reassigning and sequencing accepted loads. Kim, Mahmasanni and Jaillet (2002) develop several approaches for routing and scheduling in oversaturated demand situations.

2.2 Agent-based logistic planning

According to Wooldridge and Jennings (1995), an agent is a hardware or software based computer system with key properties autonomy, social ability, reactivity and pro-activeness. A multi-agent system (MAS) is a group of agents that interact with each other to solve a complex problem. One way to achieve this interaction between agents is by using some market mechanism where resource agents compete for orders by dynamic pricing of orders. In this paper we will use a market-based control mechanism for the allocation of vehicles to transport orders.

In the last years, research on multi-agent systems also has boosted in the logistics and operations research community. Particularly, several papers have appeared in the area of manufacturing scheduling and control. For example Cardon, Galinho and Vacher (2000) who use genetic algorithms to solve job-shop scheduling problems, and derived schedule improvements by agent negotiations. There are also some applications in material handling and inventory management (Kim et al. 2002) and supply chain management (Ertogral and Wu 2000). Only Dewan and Joshi (2000) compare their agent approach with an exact solution found by CPLEX. They conclude that centralized models are an unattractive choice compared to decentralized models because of computational inefficiency and degradation in the quality of solution with increasing problem size.

Also, several papers on agent-based transport planning and scheduling have been published. In the area of railroad scheduling, Böcker, Lind and Zirkler (2001) present a multi-agent approach for real-time coupling and sharing of train wagons. In Zhu, Ludema and van der Heijden (2000) a multi-agent solution for air cargo assignment is considered. Although this paper contains an interesting agent-based application, it does not provide detailed information on the design of a multi-agent system itself in terms of goals, behavior, pricing strategies etc. An interesting contribution comes from Fischer, Muller and Pischel (1996) who developed a simulation testbed for multi-agent transport planning, called MARS. They describe the information architecture and decision structure for quite generic transport planning systems and test their model on the traditional vehicle routing problem with time-windows where all orders are known in advance.

In Hoen and Poutré (2004) a multi-agent system is presented for real-time vehicle routing problems with consolidation in a multi-company setting. Cargo is assigned to vehicles using a Vickrey auction. They show the advantage of truck decommitment, which is the option to break an agreement in favor of a better deal if another truck from the same company can handle the cargo. They use a simple bidding strategy, i.e. the vehicle bid equals the revenue of an order that is delivered minus the additional pickup, transportation and delivery costs. They do not consider time windows within a day.

Another interesting contribution comes from Figliozzi, Mahmassani and Jaillet (2003), who present a framework for the study of carriers' strategies in an auction marketplace for dynamic full truckload vehicle routing problems with time windows. They also use a Vickrey auction and a simple heuristic for generating bids, namely the additional costs of serving a shipment by appending it to the end of the vehicle schedule. They focus on profit allocation rather than on the efficiency of assignment decisions. In (Figliozzi, Mahmassani and Jaillet 2004) they study the impact of different assignment strategies on the travel costs under various demand conditions. They consider four fleet assignment methods that are related to the agent-based approaches considered in this paper. We explain the differences compared to our research in the next section.

2.3 Contribution to the literature

Although some results on multi-agent planning and scheduling are available in the area of transportation, the level of intelligence is still limited in many cases. Also, many papers deal with the design of an agent architecture rather than analyzing the relation between agent behavior and the overall network performance. Especially little is known about the performance of agent-based transportation control compared with more traditional control methods. Our contribution focuses on the following new issues for agent-based transport scheduling:

- A combination of soft time windows and incomplete information (demand, order handling times).
- A study of the impact of additional intelligence of agents (both vehicle agents and shipper agents) on the overall system performance.
- A comparison of our multi-agent system to more traditional approaches for real time transport planning based on fast look-ahead rules and OR algorithms (serial scheduling).
- An analysis of performance *robustness*, measured by the standard deviation of the daily service levels.

- An analysis of the impact of order characteristics (such as tightness of the time window) on the overall costs.

3 Model, assumptions, terminology and notation

The key issue in our research is to match available transportation capacity to orders that arrive during schedule execution. The matching of available vehicle capacity with incoming orders can be done using OR-based heuristics or using an agent-based approach. We make the following model assumptions:

- All transport orders have a size of one Full Truck Load (FTL);
- Vehicles are location aware and fleet owners are aware of the next node to be visited by their vehicles;
- No orders may be rejected, even if it is clear that an order cannot be delivered in time;
- The total transportation capacity is sufficient to handle all orders in the long run;
- An order in process cannot be interrupted (no preemption); that is, a vehicle may not temporarily drop a load in order to handle a more profitable load and return later on; however, empty moves may be interrupted any time;
- Communication between shippers, vehicles and fleet owners is possible any time.

In the next sections we describe our transportation problem in more detail.

3.1 Transportation network and demand

We consider a transportation network that is inspired by a case for an automated transportation network using AGVs (Automatic Guided Vehicles) as described in Heijden et al. (2002). The network consists of a set of nodes and a set of arcs connecting these nodes. In the case study, the arcs represent underground tubes through which the AGVs drive between nodes (terminals). Each node has a number of docks for loading and unloading cargo. As a consequence, vehicles may face significant waiting times at the nodes. For more details we refer to Section 7.1.

Orders to transport unit loads between these nodes arrive one-by-one according to some unknown stochastic arrival process. Orders are characterized by the following parameters: the origin node i, the destination node j, the earliest pickup time at the origin r, the latest delivery time at the destination d (due time) and the time a at which the order becomes known in the network $a \leq r$. The earliest pickup time is a hard restriction and the due time is a soft restriction. The time to handle an order from node i to node j (waiting for loading, loading, driving from node i to node j, waiting for unloading, and unloading) is a random variable and denoted by τ^f_{ij} . Variation in handling times may arise from traffic congestion, variation in loading and unloading times and waiting times at the nodes. We do not consider limitations in loading and unloading capacity at the docks explicitly, but include it as a stochastic effect in the transport order handling times. The time to drive empty from node i to node j is a random variable τ^e_{ij}. The order handling times and travel times of empty vehicles are unknown and should be learned from historic data.

3.2 Cost structure and performance measurement

To evaluate the system performance, we use the following key performance indicators:

- Service level, i.e. the fraction of orders that is delivered before the due time.
- Stability of the service level, measured by the standard deviation in service level per simulation period.
- Percentage of driving loaded, i.e. the fraction of the total distance that is traveled empty, being an indicator for energy waste and loss of vehicle capacity.
- Relative additional costs, defined as the ratio of the costs for empty driving and penalties and the costs for driving loaded, or (total costs - costs driving loaded) / costs driving loaded.

The relevant cost factors for vehicles are (i) variable costs c^t per time unit, both for loaded or empty driving (ii) penalty costs $c^p(T)$ as function of the tardiness T. We assume that the fixed costs are identical for all vehicles, so that they are not relevant for scheduling decisions.

3.3 Schedules

The transport schedule consists of a set of schedules per vehicle. Each vehicle has a list of jobs and a schedule to execute these jobs. Here we use the term 'job' for orders that have been accepted by a vehicle for execution.

Formally, we define a vehicle schedule as a sequence of actions of the following types: (i) move loaded along arc (i, j) (ii) move empty along arc (i, j) (iii) wait at node j until time t. If a job has been delivered at node i and the next job in the schedule has to be loaded at j later on, the vehicle moves immediately empty to j and waits over there. At any point in time, the first job in a vehicle schedule is in execution and cannot be interrupted. A schedule will always end with option (iii) at some node with $t = \infty$. Given a set of K jobs, the number of job sequences equals $K!$. Given a certain job sequence, the timing of the jobs and the corresponding empty moves should be determined.

Vehicle schedules are updated at the following events: (a) completion of the first action in a schedule (b) matching a new external load with available vehicle capacity. Depending on the control method, also periodical replanning is possible.

4 Agent-based planning concepts

In our agent-based planning concept, we assign vehicles to jobs using a market-like negotiation protocol that implicitly coordinates the agents' decisions. The definition of such an agent-based planning concept depend on three key choices: (i) which agents to distinguish with their tasks and goals, (ii) which products (services) to trade, and (iii) which market mechanism (auction) to define. We will address these three issues below. The goal-directed behavior of each agent will be discussed in Section 5.

4.1 Agent types

To assign orders to vehicles, we choose for an elementary structure with one agent per vehicle and one agent per order. Further, we use a fleet manager agent to collect and analyze auction and processing time data of all its vehicles and to distribute the results to its vehicles when needed. In this way, the vehicle agents have access to more information than their own history only. The same applies to the shipper agents for all the orders issued by the shipper. Hence our multi-agent structure consists of four agent types, see Figure 1.

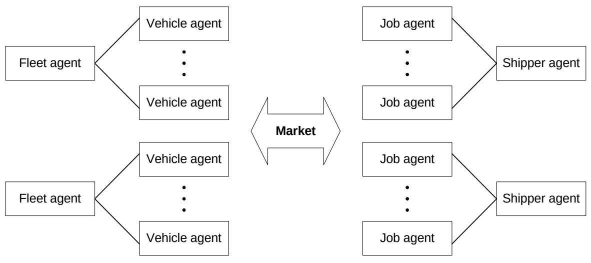


Figure 1: Agent structure for transportation networks

A vehicle agent has the goal to maximize its profit by deploying its capacity. A job agent has the goal to arrange transportation of the corresponding load before the due time at minimal costs. In a basic structure, all vehicle agents and job agents meet on the marketplace where they negotiate to assign jobs to vehicles. Each vehicle agent maintains its own schedule. Hence the solution to the global scheduling problem emerges from the local scheduling and pricing decisions of the vehicle agents. In this way, one complex overall plan is replaced by many smaller and simpler plans.

The introduction of hierarchy may improve the coordination between agents. We can define hierarchy both at the job level and at the resource level. At the job level, a shipper agent can be responsible for a set of orders. A possible task is to reallocate the transport capacity that has been acquired such that their orders are handled before the due times at lowest costs. For example, they may switch an order that has been scheduled but that has not been started yet with a rush order with a similar trajectory. To this end, they have full information on all orders under their control and all transport capacity that has been acquired for these orders. At the resource level, a fleet agent can be responsible for a subset of vehicles. If they know the positions and local schedules of all their vehicles, they can reassign vehicles to jobs to improve the profit of the fleet.

Although a hierarchical concept is interesting, we start with a fully decentralized concept. It is interesting to examine whether such a simple agent-based concept can already meet the performance of traditional OR based planning methods. However, we will use fleet agents and shipper agents to collect relevant information and to distribute it to the vehicle agents and job agents. In Sections 5.1 and 5.2 we present two extensions that require some form of hierarchical coordination.

4.2 Product definition

To create a marketplace, we need a product definition. We distinguish the following options:

- Transportation of an order from location i to location j, to be loaded not earlier than the release time r and to be delivered before the due time d.
- Transport capacity of a unit load that is available at node i at time t_1 to be used during a time period T. The advantage compared to the first option is that it provides the flexibility to reserve capacity for future jobs with some arbitrary destination. However, bidding is harder because not much can be said about the expected vehicle location at time $t_1 + T$.

- Transport capacity of N vehicle loads that can be used in some time interval $[t_1, t_2]$. Such a bulk trade may be advantageous for fleet management as a whole, but it is not suitable for a decentralized planning concept.

- Transport capacity of a unit load from node A to node B that has to be picked up at time t_1 and that has to be delivered at time t_2. Although this definition fits well with the order definition, it hampers flexibility for dynamic reallocation of capacity when additional (rush) orders arrive.

We choose for the first option, because it offers both simplicity for bidding and flexibility for schedule alteration, particularly if the order due time may be violated at some penalty costs as described in Section 3.2.

4.3 Auctioning mechanism

Several auction mechanisms have been proposed for distributed scheduling, see e.g. Wellman and Walsh (2001). Some common auction types are:

- Bargaining, this is a one-on-one negotiation protocol where all trading partners contact each other individually.

- Sealed-bid auctions where every bidder submits his bid only once and the best bid is selected; special cases are the first-price sealed-bid auction where exactly the price offered is paid, and the Vickrey auction in which the bidder receives the price of the one but best offer (second-price sealed-bid).

- Open outcry auctions consist of multiple bidding rounds where all bids are known to each bidder. Variants are (i) the English auction, where bidders sequentially either raise their bids or withdraw in each round until a single bidder is left, and (ii) the Dutch auction, where the price is reduced step by step starting from a high level until some bidder accepts the price.

We select the Vickrey auction as mechanism in our paper because of its simplicity. First of all it requires a single bidding round. Second, under some mild conditions the optimal bid is the net cost price of the bidder, who will make profit from the margin between the two best bids, cf. Vickrey (1961). Therefore, it provides a natural mechanism for acceptable profits. An advantage of this simple bid price is that it enables us to concentrate on the transportation control variables themselves rather than on learning and rationality issues of the agents. A drawback is that the profits may reduce to (almost) zero if the number of competitors becomes large.

We implement the market mechanism as follows. Each time an order l arrives, the corresponding job agent starts an auction by asking all vehicles to bid. Each vehicle agent $v \in \mathcal{V}$ creates a single bid b, consisting of a price, an expected departure time and an expected arrival time. Next, the job agent evaluates all bids and sends a grant or reject message to the vehicle agents. We allow the job agent to reject all bids if it expects to receive a better bid later on (see next section).

5 Bid calculation and evaluation

5.1 Bid calculation by vehicle agents

Let us denote the current schedule of vehicle v by S_v^0. The acceptance of an additional job will lead to a new vehicle schedule, for which we may consider several alternatives S_v^n, where n is the index of the vehicle schedule alternative. For example, we may insert the new job at various positions in the current schedule or we may shuffle the entire schedule to find a new optimum. Because we use a Vickrey auction, the bid price of vehicle v equals to the minimum additional costs over all alternative schedules n. As mentioned in Section 3.2, the additional costs depend on the additional time needed to move the load, possibly additional waiting time and the change in the total penalty costs for tardiness:

$$\widehat{P}_{v,l} = \min_n \left(c_v^t \Delta T_{v,l,n} + c_v^w \Delta W_{v,l,n} + \sum_{\forall o \in S_v^n} \left\{ c_o^d \left(D_{v,o,l,n} \right) - c_o^d \left(D_{v,o,l,0} \right) \right\} \right)$$

where

$\Delta T_{v,l,n}$ = expected additional travel- and handling time required for vehicle v in schedule alternative n to transport the job l

$\Delta W_{v,l,n}$ = expected additional waiting time for vehicle v in schedule alternative n after adding job l (which may be negative if the new job can be inserted in a gap in the current vehicle schedule thereby reducing waiting time)

$D_{v,o,l,n}$ = the tardiness of job o after adding job l to the schedule of of vehicle v using alternative n (where the tardiness $D_{v,l,l,0}$ of the new job l in the current schedule is zero because it has not been scheduled yet).

Note that we cannot simply include the difference in total tardiness in the bid price, because the penalty costs are not necessarily a linear function of the tardiness. It is obvious that a bid depends on the internal order scheduling of the vehicle agent. We consider three variants for internal vehicle scheduling.

First, the simplest method (called *AgentEnd*) is to add a new job to the end of the current schedule. So, we have a single schedule alternative (n=1). Then the change in penalty costs can only be due to tardiness for job l because the expected arrival times of the jobs in the current schedule are not affected. Additional waiting time $\Delta W_{v,l,1}$ can only occur at the origin of job l. The additional travel time $\Delta T_{v,l,1}$ equals the handling time of order l plus the time needed to move the vehicle empty from the end location of schedule S_v^* to the start location of job l.

A second option is to insert the new job at any position in the existing schedule S_v^* without altering the order of execution for the other jobs. We will refer to this option as *AgentInsert*. Hence the number of schedule alternatives equals the number of jobs in the current schedule, because the first job is in execution. For bid calculation we have to consider the cost components for the new job plus all jobs from the current schedule that will be served later on.

A third option is to construct a completely new schedule except for the job currently in execution. As this means solving a Traveling Salesman Problem (TSP), we refer to this method as *AgentTSP*. We use a depth-first, branch and bound algorithm, where we used an upper bound found with AgentInsert to test the lower bound for the remaining branch. This requires not too much computation time because the number of jobs in a vehicle schedule is usually small (say less than ten) and AgentInsert provides a reasonable upper bound. Otherwise we have to rely upon well-known fast heuristics for the TSP, such as tabu search, cf. Gendreau, Hertz and Laporte (1994).

Because of the dynamic nature of the problem it is not guaranteed that the initial assignment of a job to a vehicle remains optimal as new orders arrive and travel time realizations become known. Therefore we introduce an option to exchange jobs between vehicles that we call *Trade*. Whenever a vehicle, after unloading at a certain terminal i, has to travel empty to terminal j, its agent searches for another vehicle agent that has a job from i to j that has been released but that is not started yet. Then the job that yields the highest savings (if positive) will be transferred to the vehicle to avoid empty traveling.

5.2 Bid evaluation by job agents

The job agents have to evaluate all bids; determine for each bid whether to accept or to reject it. We consider two variants for job agent behavior. In the first variant, the agent simply accepts the best bid received by all vehicle agents. In the second variant, the job agent rejects all bids if they are all higher than a certain threshold. The idea behind this is that the job agent may expect to receive a better bid when reauctioning at a later point in time. After all, prices fluctuate over time due to changes in the available transportation capacity and in the vehicle schedules. So if the best bid is relatively high (which can be learned from history) and there is still quite some time until the latest pickup time of the job at its origin, it may be better to wait for a more attractive price. As the deadline for dispatch comes nearer, the job agent may increase the threshold to get transportation.

We assume fixed periods between reauctioning of an order that has not been assigned to a vehicle yet. We call this variant *DynamicThreshold*. The decision of the job agent is (1) to set an initial threshold price for the first auction round (2) to determine the threshold prices for all further auction rounds. The fixed time between auction rounds for the same order is a parameter of the job agent. In order to determine the thresholds, the job agents need insight in the cost and handling times for their routes. As mentioned in Section 4.1, the shipper agents keep track of travel times and prices and distribute it to the job agents.

The bid acceptance under DynamicThreshold works as follows. For the timing between successive auctions for the same order, we take a fixed period R. It is logical to relate the threshold price to the maximum number of auction rounds N before the job has to be transported. We have that $N = \lfloor (d - t - a)/R \rfloor + 1$ with d the due date, a the first announcement time of the order and t the expected handling time as obtained from the shipper agent. Without loss of generality, we assume that R is such that always $N \geq 2$ (if not, the DynamicThreshold variant coincides with the first variant discussed in this section in which the lowest bid is always accepted).

The threshold prices can be based on expectations of the outcomes of future auctions. In this case, the threshold price for a certain round equals the expected price we could receive in the next auction rounds given a certain threshold policy. However, it is difficult to model the outcomes because different auction rounds are not independent. Therefore we will consider a much simpler strategy.

The threshold price p_N for the last auction round is always infinite, i.e. any offer is accepted in order to force the job to be served. The first threshold price p_1 equals a certain minimum price P_{min} and the threshold price for the second last auction round p_{N-1} equals a maximum price P_{max}. These values P_{min} and P_{max} can be based on historical data provided by the shipper agent. We consider two pricing strategies: linear and quadratic. For the linear strategy, the threshold price p_r in round r is given by:

$$p_r = P_{\min} + \left(\frac{P_{\max} - P_{\min}}{N-2}\right)(r-1) \qquad \text{for } r = 1, ..., N-1$$

For the quadratic pricing strategy we define:

$$p_r = P_{\min} + \left(\frac{P_{\max} - P_{\min}}{(N-2)^2}\right)(r-1)^2 \qquad \text{for } r = 1, ..., N-1$$

We will examine the impact of DynamicThreshold in Section 8.

6 Traditional OR based heuristics as benchmark

Traditionally, heuristics from operations research are used for real-time scheduling in transport networks. We will use two of the methods from Heijden, Gademan and van Harten (2002) as benchmark for our agent system, because the focus in that paper is on a similar problem as we consider here.

Both methods that we consider are hierarchical methods. At the top level, vehicles are distributed amongst nodes based on actual and expected orders, without detailed job assignment. At the node level, vehicles are assigned to jobs, where only the vehicles can be used that are assigned to that node by the top level. The advantage of such an approach is that a complex schedule is decomposed into two simpler decisions. One of these decisions, assignment of vehicles to jobs, should be done in real time. The other decision, distribution of vehicles amongst nodes, should be done frequently, but not necessarily real time, because it is a higher-level decision without immediate consequences. We will use two methods that fit within this hierarchical framework, namely hierarchical coordination and integrated planning.

Under hierarchical coordination, the top level distributes vehicles using a simple priority rule, based on a central order list and a central overview of all vehicle positions and current activities. First, we calculate the latest departure time for each order as the due time minus an offset for the expected handling time (loading, transportation, unloading) and the variation in the handling time. Next, we sort the order list in increasing order of latest departure times. We process the list sequentially. To each order, we assign the vehicle that can be available at the earliest point in time. If a vehicle is waiting at or driving to a different node, the top level issues an empty vehicle repositioning orders with corresponding latest dispatch time to that node.

At the node level, we have a list of orders to be dispatched (with latest departure time) and a list of empty vehicle dispatch orders (with latest dispatch time). Every time a vehicle becomes available at the node, we choose the highest priority order from both lists. For efficiency reasons, we try to combine empty dispatch orders with load dispatch orders if possible. For example, if it is most urgent to dispatch a job from node A to node B, we look in the order list of node A whether there is a (lower priority) load dispatch order from A to B, and if so, the vehicle takes this load on its trip. Hence the node level operates independently of the top level, but within the conditions set by the top level. See Heijden, Gademan and van Harten (2002) for more details. In the remainder of this paper we refer to this method by *LocalControl*.

In the integrated planning approach, we construct a better planning to distribute vehicles over nodes. To this end, we use serial scheduling (Ebben, van der Heijden and van Harten 2005), where different priority rules are being used to create a sequence of jobs, which are virtually assigned to vehicles. At the node level, we still decide on the assignment of jobs to vehicles. However, to maintain the structure of the vehicle distribution planning from the top level, the node level has to handle all orders in a sequence that has been prescribed by the top level. In that sense, we move responsibility from the node level to the top level, hoping to receive a better performance in return in terms of fill rate and distance traveled empty. In the remainder of this paper we refer to this method by *SerialScheduling*.

The aim of a hierarchical control concept as described above is to construct a more flexible and fast schedule compared to a fully centralized concept. The difference between centralized, hierarchical and heterarchical (agent based) control structures is illustrated in Figure 2.

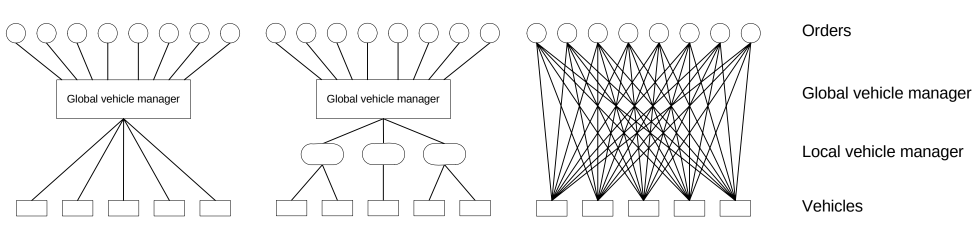


Figure 2: Control structures

Of course, a hierarchical control concept has some advantages compared to purely central control. It requires less data exchange and is capable of reacting quicker to unexpected events because of the allocation of tasks and responsibilities to two hierarchical levels. However, this hierarchical decomposition of control does not take into account the different roles of various independent stakeholders that negotiate on their mutual services and corresponding prices. Besides, a key difference with the agent approach is that under the hierarchical planning all order and vehicle information should be centrally available and that a central vehicle distribution plan is constructed.

7 Experimental setting

In this section we discuss the experimental design. We successively describe the network characteristics (7.1), the fixed parameters settings (7.2) and the experimental factors (7.3).

7.1 Network characteristics

To test the proposed multi-agent concepts and to compare them with other control methods, we use network settings inspired by a case study on a proposed underground transportation system near Amsterdam Airport Schiphol, the Netherlands (Heijden et al. 2002). We refer to this application as the OLS case, which is the Dutch abbreviation for underground logistic system. In this system, Automated Guided Vehicles (AGVs) carry cargo between terminals that are connected by tubes.

We use two different network settings. First a network layout that is derived from a specific network layout for the OLS-case with an internal transportation system at Amsterdam Airport Schiphol (AAS), see Heijden et al. (2002). This network consists of a connection of the airport with the world's largest flower auction market in Aalsmeer (VBA) and a planned rail terminal near the Zwanenburg landing strip (RTZ). At Aalsmeer there is 1 terminal, at Schiphol Airport there are 8 terminals and there is 1 rail terminal. Besides, there is a central parking area where AGVs can wait if there are temporary no jobs, because the parking space at (underground) terminals is limited. Each terminal has an internal track structure and consists of 4 docks where AGVs can be loaded or unloaded. The terminals are connected by tubes as illustrated by Figure 3.

Figure 1 shows the distances in meters between terminals, crossings and the central parking. The times to drive through or along a terminal are significant, as a terminal may have a length up to 200 meters and AGVs drive slower within terminals for safety reasons (see next section). When an AGV enters a terminal to pickup or to deliver an order, it is assigned to a specific dock

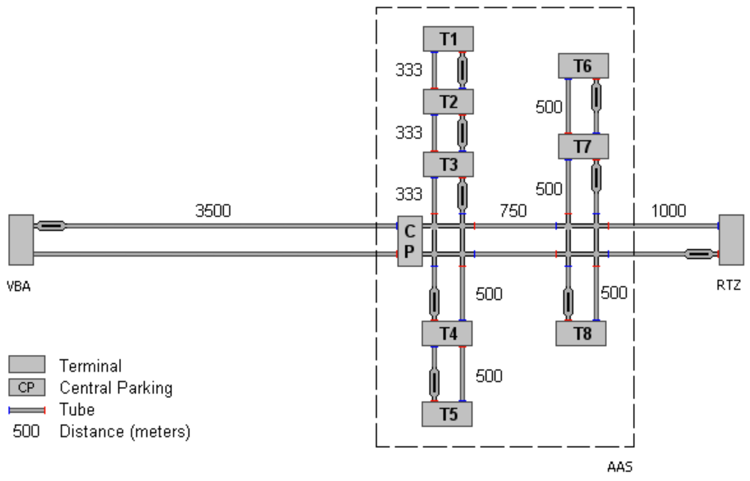


Figure 3: OLS network structure

within the terminal. The distance from the terminal entrance to the dock may vary between 100 and 400 meters, so the dock assignment causes a part of the variation in order handling times.

Because this is a quite specific setting, we also consider a second network structure consisting of 20 nodes that are uniformly placed in a squared region of 10x10 km. All nodes are mutually connected and the distances between these nodes are Euclidean. A central parking area is located in the centre of the area.

One way to deal with random networks is to generate a few (2-3) random network structures and perform some replications for each scenario. This provides insight in the performance of a few configurations only. Because we are interested in the average performance of the control methods over a range of random network structures, we chose for another option. That is, we start each replication with the creation of a new network, i.e. repositioning the location of all nodes. Of course the required number of replications will increase, but we will also get a better idea of the average performance of the various control methods.

Orders arrive according to a (non)stationary Poison process (cf. Section 7.3). Travel times between terminal entrances are deterministic and known in advance because they only depend on the distance and speed of vehicles. Although the distances are deterministic, the handling times show variation due to the following:

- Variations in loading and unloading times
- Waiting times at the terminals due to limitations on the number of AGVs on terminals

- Waiting times at the terminals due to limitations in dock capacity
- Dock-dependent distances on terminals

Therefore, we treat the handling times as random variables. The mean and standard deviation of the handling times are dynamically being updated using a standard exponential smoothing procedure, see Silver, Pyke and Peterson (1998). In case of agent-based control we use the fleet manager to keep track of all handling times and the corresponding estimates are available to all vehicles under its control.

To provide an indication of the stochasticity we found the following values in the OLS network settings. The expected handling times range from 5 minutes (T1 to T2) with standard deviation of 40 seconds, to 25 minutes (VBA to T6) with a standard deviation of 90 seconds. Although these deviations are not very large, they are significant in the handling times.

7.2 Fixed parameter settings

The vehicles have a speed of 6 m/s outside the terminals and 2m/s inside the terminals. The maximum number of AGVs allowed at a terminal is limited to 5.

As costs factors we use travel costs c^t=1 per minute and a linear penalty cost function $c^P(T) = 10T$, where the tardiness equals T minutes. These penalty costs are such that in case of agent-based control a job agent will almost always prefer an AGV that delivers the job with minimum tardiness.

The agent-based approaches also use waiting costs in their bid prices. We set these costs equal to the historical average profit per time unit. This information is collected and distributed by the fleet agent.

We set the parameters of DynamicThreshold as follows. P_{min} is equal to the mean price for a specific route, P_{max} to the maximum price paid so far for this route and the fixed time interval R between the auction rounds is set to 5 minutes, which equals the minimum handling time in the OLS network. The replanning period for the two hierarchical methods is set to 4 minutes.

7.3 Experimental factors

In this section, we discuss the factors that we will vary in our simulation experiments for both the OLS network case and the random networks.

7.3.1 OLS network

Table 1 shows the experimental factors and their settings for the OLS network.

Factor	**Range**
Demand structure	Stable, Dynamic, Highly Dynamic
Vehicle control	LocalControl, SerialScheduling, AgentEnd, AgentInsert, AgentTSP
Vehicle coordination	None, Trade
Job control	None, DynamicThreshold (linear), DynamicThreshold (quadratic)

Table 1: Experimental factors

The experimental factor "Demand structure" refers to the variation in transportation flows over time. We distinguish three cases: Stable, Dynamic, and Highly Dynamic. In the stable demand structure, (i) the order arrival rates are identical for all origin destination pairs, (ii) the order arrival rates are constant over the day, and (iii) all orders have a same time window of 60 minutes.

In the dynamic demand structure, (i) the order arrival rates are still identical for all origin destination pairs, (ii) the time between orders vary over hours of the day according to a sinus function with period of half a day, the same mean as in the stable demand structure, and an amplitude of 3 seconds, and (iii) we have three different time-windows of 30, 60 and 90 minutes that are drawn with equal probability.

The highly dynamic demand structure is similar to the dynamic demand situation, except that the order arrival rates are no longer identical for all origin destination pairs. The imbalance is given by Table 2. Within AAS all terminals have equal probability of being origin or destination.

From \ To	AAS	RTZ	VBA
AAS	0	26%	10%
RTZ	40%	0	4%
VBA	8%	12%	0

Table 2: Distribution of transportation flows

The average number of orders per day is 1800, that is a time-between-orders (TBO) of 48 seconds. These orders have to be transported by a fixed number of AGVs. This number is chosen such that all methods are capable of handing all orders in the long run, but not necessarily on time. In case of a stable or dynamic demand structure we choose to use 20 AGVs and in case of a highly dynamic demand structure 22 AGVs. The announcement times a for jobs are equal to the earliest departure time r. Therefore the time-windows can be defined as the time between the first auction for a job and the due time d.

To limit the number of experiments, we test the vehicle agent coordination (Trade) and the job agent control (DynamicThreshold) for the highly dynamic demand structure only. We consider four settings: Trade, DynamicThreshold with linear pricing, DynamicThreshold with quadratic pricing and the combination of Trade with DynamicThreshold, linear pricing. We omit the combination of Trade with DynamicThreshold, quadratic pricing because we observed that the differences between both price functions are small (see Section 8.1).

7.3.2 Random network

As a basic scenario we use a network consisting of 20 nodes, 20 AGVs, a time between orders of 1.5 minute and a time-window of 60 minutes. As key performance indicator we use the average relative costs per order, see Section 3.2. This value resembles the extra costs we make relative to the minimum costs for all orders.

The experimental factors can be found in Table 3. Each of these factors will be varied keeping the other factors equal to the basic scenario.

Factor	**Value**
Length time-windows (sec)	50, 70, 90, 110, 130
Look-ahead (min)	0, 3, 6, 9, 12
Time between orders (sec)	84, 87, 90, 93, 96
Amplitude in deviation from mean TBO (sec)	2, 4, 6, 8, 10
Number of nodes	12, 14, 16, 18, 20

Table 3: Experimental factors

We use as control methods LocalControl, SerialScheduling, AgentEnd, AgentInsert and the combination of AgentInsert with Trade and DynamicThreshold with linear pricing, referred to as *AgentInsertSmart*. To vary the time between orders during a day, we describe the TBO as a sinus with a period of half a day, a mean of 1.5 minute and change the amplitude.

We use a replication / deletion approach for our simulations (Law and Kelton 2000), where each experiment consists of a sufficient number of replications (each with different seeds) of six days, each including a one-day warm-up period. The number of replications will be determined in the next sections.

8 Numerical results

In this section, we present the results of our simulation experiments. First, we present the results for the OLS network (8.1) and afterwards for the random networks (8.2).

8.1 OLS network

We first determine the number of replications. To this end we consider the percentage of driving loaded (DL) and the service levels (SL) of all control methods for all three demand structures. The maximum number of replications needed with a confidence level of 99% and relative error of 5% is 10. To facilitate comparison, we use 10 replications for all scenarios.

We also performed a paired t-test on the key performance indicators SL and DL of SerialScheduling and AgentInsert using the 10 replications with a stable demand structure. Results show that both differences are significant with a confidence level of 99%. The results for all control methods for the different demand structures can be found in Table 4.

	Stable		Dynamic		Highly dynamic	
Control	DL	SL	DL	SL	DL	SL
LocalControl	73	95.9	72	91.4	78	93.6
SerialScheduling	73	99.2	74	96.6	78	94.8
AgentEnd	82	99.7	80	95.3	82	93.9
AgentInsert	83	100	80	97.8	82	97.0
AgentTSP	83	100	80	98.1	82	97.2

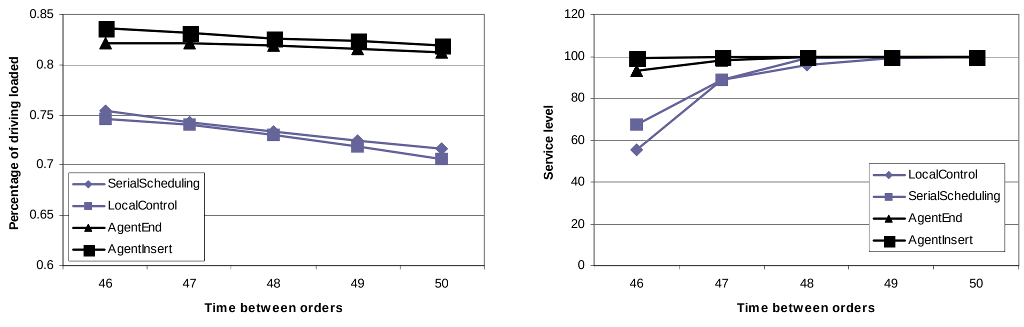
Table 4: Simulation results: comparing control methods

We see that AgentInsert and AgentTSP both yield similar results which are better than the hierarchical methods LocalControl and SerialScheduling. Because of the dynamic system behavior, solving a TSP problem exactly has apparently little added value. Because AgentTSP is also computationally intensive we skip this method in the remainder. We see that AgentEnd yields lower service levels in some cases. With regard to the percentage of driving loaded we see that our agent approach always perform better than the hierarchical control methods.

The differences in service levels is larger in case of dynamic demand compared to stable demand. In case of highly dynamic demand, we even observe that decreasing the number of AGVs from 22 to 21 yields lower service levels for the agent-based methods whereas the Local Control and Serial Scheduling heuristics are simply not able to handle all demand (the order backlog steadily increases).

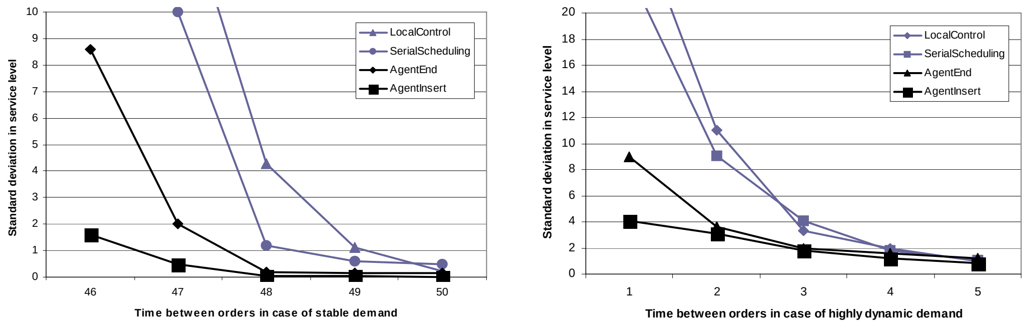
To gain insight in the sensitivity of the control methods to the order arrival intensity we vary the time between orders from 46 to 50. The impact on the percentage of driving loaded for the different control methods in case of stable demand can be found in Figure 4a and the impact on the service levels in Figure 4b.

From these figures we see that with increasing TBO, the service levels go to 100% for all control methods. However, the differences in percentage of driving loaded between the heuristics and the



(a) Impact on percentage of driving loaded (b) Impact on service level

Figure 4: Impact of varying time between orders.



(a) Stable demand structure (b) Highly dynamic demand structure

Figure 5: Impact of time between orders on standard deviation.

two agent methods increase. This means that the agent methods do not lead to unnecessary empty miles if the transportation capacity is sufficient to handle all orders in time. In case of (highly) dynamic demand situations this effective use of capacity can be used to cope with uncertainty such as rush orders. Therefore our agent control seems to be more robust than the two hierarchical control methods. This can also be seen from the standard deviation in service levels per replication over the ten replications for both the stable demand structure (Figure 5b) and the highly dynamic demand structure (Figure 5a). We see that our agent control is less sensitive to variations in demand volume, loading times and unloading times.

To examine the impact of additional agent intelligence, we show the key performance indicators DL and SL for agent systems with or without vehicle coordination (Trade) and DynamicThreshold (both linear and quadratic) in Table 5.

We see that the use of additional intelligence improves the performance, especially in case of

	AgentEnd		AgentInsert		AgentTSP	
Control	DL	SL	DL	SL	DL	SL
Normal	82	93.9	82	97.0	82	97.2
Trade	83	96.2	82	98.2	82	98.3
DP-lin	83	95.5	83	97.5	82	97.9
DP-Qdr	83	95.4	82	97.6	82	97.8
TR-DP-lin	83	96.4	83	98.4	83	98.4

Table 5: Simulation results: additional intelligence

AgentEnd. However, the improvement in service level is only significant at confidence level 98% for Trade. It might be surprising to see that the additional intelligence does not improve the percentage of driving loaded. The reason for this is that there is very little room for improvement. To illustrate this, we calculate a simple upper bound for the percentage driving loaded. Let us relax the problem by assuming that all orders are known in advance, there are no time windows and all travel- and handling times are deterministic. Then penalty costs are not relevant and the problem reduces to the minimization of the total empty travel time under flow conservation constraints. Using the average handling times and demand data resulting from our simulation experiments for the highly dynamic case, we find an upper bound of 89% for the percentage of driving loaded. Even though this upper bound is calculated under strongly simplifying assumptions, our agent methods still achieves a percentage of driving loaded that is only 6,7% less than the upper bound. On the other hand, LocalControl and SerialScheduling lead to a percentage driving loaded that is 12,4% worse than the upper bound.

8.2 Random networks

Again, we first determine the number of replications based on the basic scenario. With a confidence level of 95% and a relative error of 5%, we find that we need 6 replications in case of AgentInsert and 40 replications in case of LocalControl. The differences between the control methods are higher than before. First, this is due to changes in the network layout with each replication. Second, we are now looking at the relative costs for which the variances are much higher. We decided to use 20 replications because this provides enough significance to distinguish between the agent control methods and the two heuristics. This can be seen from the confidence intervals for the relative costs in the basic scenario (Table 6).

First we vary the time between orders. In Figure 6 we see that, for all methods, the relative costs decrease with the time between orders. We also see that SerialScheduling and LocalControl will converge to a same level. The two basic agent methods converge to a same, but lower level.

Control	**Confidence interval**
LocalControl	[59.5, 69.7]
SerialScheduling	[57.5, 63.6]
AgentEnd	[40.5, 44.1]
AgentInsert	[38.9, 41.4]
AgentInsertSmart	[39.0, 41.2]

Table 6: Simulation results: confidence intervals for relative costs

The addition of Trade and DynamicThreshold yields lower costs if the time between orders is small. The reason is that the average length of the vehicle schedules increases. Then there are more options for load exchange (Trade). Also, it can be more beneficial to use threshold prices.

In the next experiment we keep the mean time between orders the same for all simulation runs, but we change the deviation from the mean during the day. We see in Figure 7 that increasing the amplitude has negative effect on the relative costs. Not surprisingly, the AgentInsertSmart method is less sensitive from these deviations.

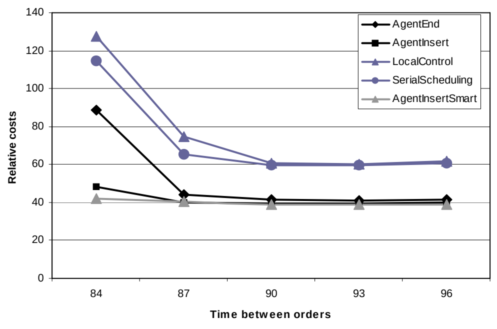


Figure 6: Varying time between orders

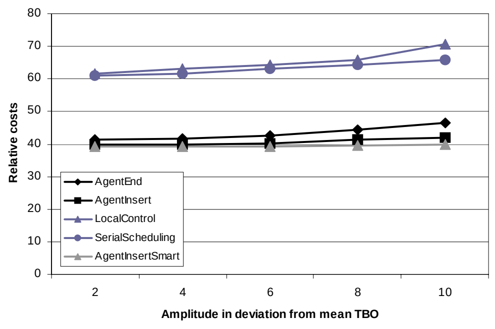


Figure 7: Varying amplitude in time between orders

Second, we vary the length of the time-windows. From Figure 8 we see that the relative costs for all methods decrease with the length of the time windows. The costs converge to a situation where the penalty costs are negligible. The methods LocalControl and SerialScheduling are not able to reduce their empty trips when the time windows increase. AgentInsertSmart benefits more from increasing time windows because the number N of possible auction rounds is higher.

Finally we vary the number of nodes in the network keeping the rest of the parameters the same, see Figure 9. In case of AgentInsert the relative costs first slightly increase converging to a value of about 40%. AgentInsertSmart works relative better if the network is small (less nodes) because then the average distances between nodes are higher and therefore trading jobs

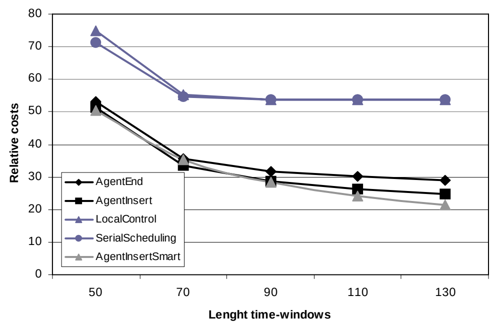


Figure 8: Varying lenght of time windows

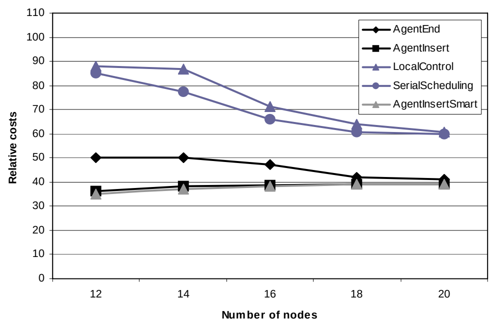


Figure 9: Varying number of nodes

will be more beneficial. The relative costs for all other methods decrease in the number of nodes. Closer inspection reveals that the penalty costs always decrease with the number of nodes while the amount of empty kilometers increase. However, both costs factors converge to a stable value. Because the penalty costs are relatively small and stable for AgentInsert, the increase in costs for empty trips will dominate the relative costs. The hierarchical methods SerialScheduling and LocalControl will benefit most from an increase in the number of nodes. However, based on our simulation results we expect that differences in relative costs will remain. Besides that, the computation time of the hierarchical methods (especially for SerialScheduling) increases with the number of nodes. Therefore we end with some notes about the computation time of the different control methods.

We implemented our methods in the simulation software eM-Plant and we performed the experiments using a Intel Pentium 4 processor at 3.4 GHz. The computation times (milli-seconds per order) for the basic scenarios can be found in Table 7. The computation times of the agent approaches include starting the auction, bidding by all vehicle agents (sequential whereas in practice parallel execution is plausible) and bid evaluation by the job agent. The computation times of the hierarchical methods consist of a periodical planning time and time for local decision making. In both hierarchical methods the computation times for local decision making per order are 2 ms on average, the computation times for periodical replanning can be found between the brackets.

We observed that the computation time of the agent methods is dependent on the number of vehicles (participants in the auctions). As can be expected, AgentInsert and AgentTSP also depend strongly on the average length of the vehicle schedules. The hierarchical methods are also dependent on the average number of open orders, but also on the number of nodes in the network. Therefore we see some longer computation times in the random network with 20 nodes.

Control	OLS	Random
AgentEnd	2	1.6
AgentInsert	8.3	6.6
AgentTSP	19.0	8.8
LocalControl	2.9 (4.9)	3.8 (5.3)
SerialScheduling	20.2 (91.8)	78.6 (109.7)

Table 7: Simulation results: computation time

The two agent extensions result in an increase in computation times. The extension Trade results in an increase of about 0.09 ms per order exchange (about 0.5% of the orders are exchanged in both network settings). Using DynamicThreshold has a bigger impact, because a load requires on average 2.6 auction rounds resulting in a proportional increase of the computation time.

9 Conclusions, generalizations and further research

In this paper, we proposed a distributed agent-based solution to real-time, dynamic transport scheduling problems. This approach has a number of advantages. First, it is more robust in the sense that it is less sensitive to fluctuations in demand or available vehicles than more traditional transportation planning heuristics (LocalControl, SerialScheduling). Second, it provides a lot of flexibility by solving local problems locally. Third, it provides online decision-making using auction mechanisms.

From our simulation experiments, we conclude that our agent approach yields a high performance in terms of vehicle utilization and service level. When we compare the best hierarchical method (of the two considered in this paper) with AgentInsert, we see that the differences in costs and percentage of driving loaded are always significant. With regard to the service levels, AgentInsert performs significantly better in most cases and never significantly worse.

We can improve the performance of the agent-based approach using two extensions: (1) we allow vehicle agents exchange jobs, and (2) we allow shipper agents to reject all bids and start a new auction later on. These extensions are particularly valuable if vehicle schedules contain many jobs on average.

Further research will mainly focus on the improvement of the agent behavior. For the job agents, formal methods for the dynamic threshold policy will be developed. For the vehicle agents, further improvement of the pricing strategy is relevant. Similar to the dynamic pricing used to sell airline seats, vehicles can price their services based on the available capacity. Although vehicles can schedule more jobs in advance, our model is still myopic. Vehicle agents only consider the direct cost of doing certain jobs, whereas it could be better to include an opportunity loss for arriving at a terminal without a next order with low expectations for an attractive load in the near future. To this end, formal methods for estimating the value of arriving at a certain location can be used, for example using approximate dynamic programming, cf. (Godfrey and Powell 2002). Then, we expect vehicles to drive pro-actively to other nodes with higher expected future revenues, or to calculate the changes of driving empty from certain terminals and include these cost in their bid prices.

References

Böcker, J., J. Lind, and B. Zirkler (2001). Using a multi-agent approach to optimise the train coupling and sharing system. *European Journal of Operational Research 134*, 242–252.

Cardon, A., T. Galinho, and J.P. Vacher (2000). Genetic algorithms using multi-objectives in a multi-agent system. *Robotics and Autonomous Systems 33*, 179–190.

Desrosiers, J., Y. Dumans, M.M. Solomon, and F. Soumis (1995). Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser (Eds.), *Network Routing*, Volume 8 of *Handbooks in Operations Research and Management Science*, pp. 35–139. Amsterdam, The Netherlands: Elsevier.

Dewan, P. and S. Joshi (2000). Dynamic single-machine scheduling under distributed decision-making. *International Journal of Production Research 38*(16), 3759 – 3777.

Ebben, M.J.R., M.C. van der Heijden, and A. van Harten (2005). Dynamic transport scheduling under multiple resource constraints. *European Journal of Operational Research 167*, 320–335.

Ertogral, K. and S.D. Wu (2000). Auction-theoretic coordination of production planning in the supply chain. *IIE Transactions 32*, 931–940.

Figliozzi, M.A., H.S. Mahmassani, and P. Jaillet (2003). Framework for study of carrier strategies in auction-based transportation marketplace. *Transportation Research Record 1854*, 162–170.

Figliozzi, M.A., H.S. Mahmassani, and P. Jaillet (2004). Competitive performance assessment of dynamic vehicle routing technologies using sequential auctions. *Transportation Research Record 1882*, 10–18.

Fischer, K., J.P. Muller, and M. Pischel (1996). Cooperative transportation scheduling: an application domain for dai. *Journal of Applied Artificial Intelligence. Special issue on Intelligent Agents 10*(1), 1–33.

Gendreau, M., A. Hertz, and G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. *Management Science 40*, 1276–1290.

Gendreau, M. and J.Y. Potvin (1998). Dynamic vehicle routing and dispatching. In T. Crainic and G. Laporte (Eds.), *Fleet Management and Logistics*, pp. 127–157. Kluwer, Academic Publishers.

Giani, G., F. Guerriero, G. Laporte, and R. Musmanno (2003). Real time vehicle routing: solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research 151*, 1–11.

Godfrey, G.A. and W.B. Powell (2002). An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science 36*(1), 21–39.

Heijden, M.C. van der, A. van Harten, M.J.R. Ebben, Y.S. Saanen, E. Valentin, and A. Verbraeck (2002). Using simulation to design an automated underground system for transporting freight around schiphol airport. *Interfaces 32*(4), 1–19.

Heijden, M.C. van der, M.J.R. Ebben, N. Gademan, and A. van Harten (2002). Scheduling vehicles in automated transportation systems. *OR Spektrum 24*, 31–58.

Hoen, P.J. 't and J.A. la Poutré (2004). A decommitment strategy in a competitive multi-agent transportation setting. In D. P. P. Faratin and J. Rodriquez-Aguilar (Eds.), *Agent Mediated Electronic Commerce V (AMEC-V)*, pp. 56–72. Berlin: Springer - Verlag.

Kim, B., S. Heragu, R.J. Graves, and A.St. Onge (2002). Intelligent agent based modeling of an industrial warehouse problem. *IIE Transactions 34*(7), 601–612.

Kim, Y., H.S. Mahmasanni, and P. Jaillet (2002). Dynamic truckload truck routing and scheduling in oversaturated demand situations. *Transportation Research Record 1783*, 66–71.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research 59*, 345–358.

Law, A.M. and W.D. Kelton (2000). *Simulation Modeling and Analysis*, Volume 3rd edition. Singapore: McGraw-Hill.

Mahmassani, H.S., Y. Kim, and P. Jaillet (2000). Local optimization approaches to solve dynamic commercial fleet management problems. *Transportation Research Record 1733*, 71–79.

Powell, W.B. and T.A. Carvalho (1998). Dynamic control of logistics queueing networks for large scale fleet management. *Transportation Science 32*(2), 90–109.

Psaraftis, H. (1988). Dynamic vehicle routing problems. In B. Golden and A. Assad (Eds.), *Vehicle Routing: Methods and Studies*, pp. 223–248. Amsterdam, The Netherlands: Elsevier.

Regan, A.C., H.S. Mahmassani, and P. Jaillet (1995). Improving efficiency of commercial vehicle operations using real time information: Potential uses and assignment strategies. *Transportation Research Record 1493*, 188–198.

Regan, A.C., H.S. Mahmassani, and P. Jaillet (1996). Dynamic decision making for commercial fleet operations using real-time information. *Transportation Research Record 1537*, 91–97.

Regan, A.C., H.S. Mahmassani, and P. Jaillet (1998). Evaluation of dynamic fleet management systems. *Transportation Research Record 1645*, 176–184.

Silver, E.A., D.F. Pyke, and R. Peterson (1998). *Inventory management and production planning and scheduling.* New York: Wiley.

Toth, P. and D. Vigo (2002). *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance 16*(1), 8–37.

Wellman, M. and W. Walsh (2001). Auction protocols for decentralized scheduling. *Games and Economic Behavior 35*, 271–303.

Wooldridge, M. and N.R. Jennings (1995). Agent theories, architectures, and languages: a survey. In *Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures and Languages*, pp. 1–39. Springer-Verlag.

Yang, J., P. Jaillet, and H.S. Mahmassani (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science 38*, 135–148.

Zhu, K., M. W. Ludema, and R.E.C.M. van der Heijden (2000). Air cargo transport by multi-agent based planning. In *Proceedings of the Thirty-third Annual Hawaii International Conference on Systems Sciences*, Maui, Hawaii.