Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 167 (2005) 320–335
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

Dynamic transport scheduling under multiple resource constraints

M.J.R. Ebben [a,b,*], M.C. van der Heijden [a], A. van Harten [a]

[a] School of Business, Public Administration and Technology, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
[b] TPG Post, Quantitative Support, P.O. Box 30250, 2500 GG The Hague, The Netherlands

Received 29 April 2002; accepted 22 March 2004
Available online 1 July 2004

Abstract

This paper presents a heuristic for the dynamic vehicle scheduling problem with multiple resource capacity constraints. In the envisaged application, an automated transport system using Automated Guided Vehicles, bottleneck resources are (1) vehicles, (2) docks for loading/unloading, (3) vehicle parking places, and (4) load storage space. This problem is hard, because interrelated activities (loading, transportation, unloading) at several geographical locations have to be scheduled under multiple resource constraints, where the bottleneck resource varies over time. Besides, the method should be suitable for real-time planning. We developed a dedicated serial scheduling method and analyzed its dynamic behavior using discrete event simulation. We found that our method is very well able to find good vehicle schedules satisfying all resource constraints. For comparison, we used a simple approach where we left out the resource constraints and extended the processing times by statistically estimated waiting times to account for finite capacities. We found that our newly designed method finds better schedules in terms of service levels.

Keywords: Transportation; Scheduling; Heuristics; Simulation

1. Introduction

1.1. Motivation

In this paper, we present a serial scheduling method for the real-time dynamic resource-constrained vehicle scheduling problem. This research was motivated by a study on an automated transportation system near Schiphol Airport, the Netherlands; see Van der Heijden et al. (2002b). Automated Guided Vehicles (AGVs) transport time critical products between terminals, with distances up to 10 km. These terminals require a serious investment in equipment (AGVs, docks) and space (parking places, cargo storage), especially because some of them have to be constructed underground. The system should provide a reliable service (on-time delivery) against low costs. Therefore, the number of docks, parking places and cargo storage locations in the terminals should

* Corresponding author. Address: Prinses Beatrixpaan 23, 2595 AK, The Hague, Netherlands. Tel.: +31-70-3348183; fax: +31-70-3349784.
E-mail address: m.j.r.ebben@tpgpost.nl (M.J.R. Ebben).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2004.03.020

be minimal. Good vehicle schedules can possibly reduce the resource requirements while still meeting service level agreements (e.g. 98% on-time delivery). Scheduling is especially critical in this particular situation where time windows are tight and resource utilization is high.

1.2. Problem

We consider a transportation network consisting of terminals, between which cargo needs to be transported by vehicles, and central parking areas, where vehicles can be positioned when temporarily not needed or when there is a lack of space in other parts of the network. A transport job is defined as a request to load some cargo at a certain terminal after a given release time, to move it to its destination terminal and to unload the cargo before a given due time. Jobs arrive over time and real-time decision making is required, given their short inter-arrival times and processing times (in minutes rather than in hours). Besides the number of vehicles, the docking, parking and cargo storage capacity of the terminals are limited. The question is how to integrate these capacity restrictions in the planning procedure for the vehicles. Our goal is to maximize the percentage of jobs served on time, given the finite resource capacities. As a secondary criterion, we aim to reduce the fraction of kilometers driven empty to reduce operational transportation costs.

1.3. Literature

The problem described above can be seen as a dynamic resource scheduling problem (Powell, 1998). More specifically, we can characterize it as a dynamic real-time vehicle scheduling problem (Gendreau and Potvin, 1998; Powell, 2003). Vehicles are scheduled over a time horizon, while new service requests arrive in real-time. After each arrival, the current schedule may be reconsidered to include the new arrival, but periodic rescheduling is also an option. In the dynamic case, the planner has to react on events that occur in real-time, such as job arrivals, congestion and disturbances. Most literature on vehicle routing and scheduling deals with the static problem, in which all data are known before the routes are constructed (see e.g. Fisher, 1995; Desrosiers et al., 1995). The dynamic vehicle scheduling problem is receiving increasing attention in the literature, caused by an increasing availability of real-time data and the need for real-time decision making to ensure high customer service (Powell et al., 1995; Gendreau and Potvin, 1998; Ichoua et al., 2000). Given the requirement of real-time decision support, most authors resort to heuristics. Gendreau et al. (1999) describe a parallel tabu search heuristic for the real-time vehicle dispatching problem and compare it with other heuristic methods. Ulusoy and Bilge (1993) propose an iterative algorithm to integrate the vehicle scheduling with the overall scheduling activity in a flexible manufacturing environment, but they do not indicate whether this approach can be used in real-time dynamic scheduling.

In the vehicle scheduling literature, we could not find anything about multiple node capacity restrictions. Only in the literature on AGV systems some authors take buffer sizes at machine locations into account in dispatching the vehicles. Kim et al. (1999) try to balance the workload in a manufacturing job shop with automated guided vehicles. The proposed procedure shows good results compared to existing dispatching rules when the buffer sizes are limited. They indicate that it is important to identify the most critical resource and to make good use of this resource in the operational control.

A research area where resource constraints are common is resource-constrained project scheduling; see Brucker et al. (1999) for an overview. Two well-known heuristics for the resource-constrained project scheduling problem are the serial and parallel scheduling scheme (Kolisch, 1996). Van der Heijden et al. (2002a) apply a serial scheduling method to the vehicle scheduling problem in an automated transportation network and show promising results compared with other heuristics. In their method, they neglect finite resource capacities at the terminals. In their cases, there are no parking and storage restrictions, so finite docking capacity induces waiting times of vehicles at terminals until they can be (un)loaded.

1.4. Novel aspects

Given the promising results in Van der Heijden et al. (2002a) and the structure of the serial scheduling method, sequentially selecting and scheduling one job, it appears that this method can be extended to the situation with terminal capacity constraints. This is the focus of our paper. Such an extension is not straightforward, however. Because each job consists of a sequence of activities that each requires a different set of (constrained) resources, strong interrelations between the resource constraints arise that complicate the scheduling. We will illustrate this below.

A vehicle, a dock and the cargo need to be available at the terminal of origin for the loading activity. For a vehicle that is scheduled to arrive a little too early, a parking place has to be available. If parking space is insufficient, the vehicle has to arrive later and so it has to depart later from its current location (say another terminal), but then the parking space at the other terminal should be sufficient. An alternative is to select another, more remote, vehicle. Furthermore, the time at which the cargo is loaded determines the arrival time of the loaded vehicle at the destination terminal. At that time, either a dock should be available for unloading or a parking place has to be available until the time that the vehicle can be unloaded. If that is not possible, loading of the vehicle at the origin terminal has to be postponed. However, then cargo remains longer in storage, which may cause violation of the storage capacity. It will be clear that resource conflicts can easily arise. Avoiding such conflicts is not trivial; it is even not guaranteed that a feasible schedule exists.

In practice, occasional violation of a capacity constraint does not always cause serious problems. For example, a temporary lack of dock and parking capacity means that a vehicle cannot enter the terminal and that it has to wait in front of the terminal entrance. Because it can block other traffic, it may cause delay of other jobs. Also, insufficient storage capacity means that cargo arriving at the terminal for loading cannot be stored. As a consequence, the cargo has to be stored at another location (in the truck delivering the cargo, outdoor). This may induce truck delay and/or degradation of cargo quality. Although we could model this as soft resource constraints in theory, costs of capacity violation can be hard to quantify in practice (what are the costs of a vehicle that has to wait in front of the terminal entrance?). Therefore, we treat the capacities as hard constraints in our scheduling procedure and we study the degree in which these constraints are violated in a simulation study.

At this point, it will be clear that the scheduling problem is hard, especially if it has to be used in real-time. As a solution for such a scheduling problem is not available in the literature (see Section 1.3), we will develop a new method in the remainder of this paper.

1.5. Approach

At first sight, it seems logical to check the various resource capacities one-by-one and to postpone activities if the capacity is insufficient. Then an important issue is in which order resource capacities should be checked such that (1) due times are met, and (2) schedules are constructed in an efficient way, because we focus on real-time planning. We first design a serial scheduling procedure based on sequential capacity checking. Next, we extend this basic serial scheduling method with improvement heuristics to reduce the amount of empty kilometers. Because the resulting schedule might not be feasible with respect to parking capacity, we solve these infeasibilities using a post-processing method. We use discrete event simulation to investigate the dynamic behavior and performance of our scheduling procedure.

The paper is organized as follows. In the next section, we present the main principles of the serial scheduling method and we describe our problem setting and assumptions. Then we explain how a schedule is constructed given a set of transportation jobs (Section 3). In Section 4, we discuss the implementation in a dynamic environment, i.e. a rolling horizon. We tested the proposed method by a simulation study on an automated transportation network. We describe the experimental design in Section 5 and the numerical results in Section 6. In the last section, we present our conclusions.

2. The serial scheduling method, problem setting and assumptions

2.1. Main principles

Serial scheduling, a priority rule based scheduling method, is an important heuristic solution technique. This method (1) is intuitive and easy to use, (2) is fast, and (3) shows good results compared to other heuristics (Kolisch, 1996). Generally, a priority rule based scheduling heuristic consists of two elements: a schedule generation scheme and a priority rule. A feasible schedule is generated by extending a partial schedule. In each stage, an activity that is not yet scheduled is selected according to the specified priority rule. Each activity can only be scheduled once.

The original serial scheduling method consists of J stages. At each stage, one activity is selected, based on some priority rule, and next it is scheduled at its earliest resource and precedence feasible start time (Kolisch, 1996). There are two disjoint sets of activities, the set S which contains all scheduled activities and the decision set D, which contains the activities that still have to be scheduled and which predecessors are all in the set S. At each stage, an activity is scheduled and moved from the set D to the set of scheduled activities S. The decision set D is updated and a new activity is selected until all activities have been scheduled.

The serial scheduling method can be used in a single-pass or a multi-pass approach. In a single-pass approach, the entire decision set D is scheduled once using a single priority rule. In the multi-pass approach, each pass uses a different priority rule to select an activity from the decision set D. The best schedule is selected according to some objective function, such as minimum total lateness.

2.2. The process to be scheduled

Fig. 1 displays the transportation process. A job arrives at the terminal of origin and is available for transportation at the release time. In case the job cannot be transported directly, the cargo is stored temporarily in the in-buffer. When a vehicle and a dock are available, the job can be loaded and the vehicle starts driving to the destination. When the vehicle arrives at the destination, it is possible that the vehicle has to wait at a parking place until docking and storage capacity are available. The vehicle can be unloaded when both resources are available. After unloading, the cargo is stored in the out-buffer until the due time. Because multiple resources have to be available at both the origin and the destination during linked time intervals, resource utilization at different terminals has to be tuned. The start time of a job has to be feasible with respect to all resources required, which can make it difficult to find the earliest feasible start time.

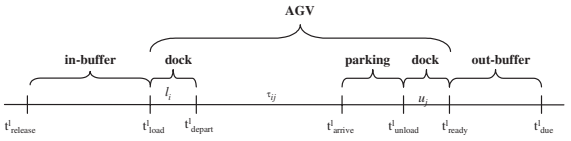

Fig. 1. Resource requirements for job l over time.

2.3. Serial scheduling with capacity constraints

In principle, finite resource capacities do not affect the job selection step (priority rule), but it seriously complicates finding the earliest resource and precedence feasible start time. As mentioned before, an important issue is the order in which resource capacities should be checked and how the timing of the various activities needed to process a job should be determined. A problem that we encountered is the parking capacity needed for vehicles between unloading one job and loading the next job. Because we schedule jobs one-by-one based on a (time-dependent) priority rule, it is not clear when scheduling a certain job which job the vehicle will handle next. As a consequence, we also do not know how long the vehicle will have to wait in which parking. If the parking capacity seems to be exceeded, we can only prevent this by planning an empty trip to the nearest location with sufficient parking space. Later on, we can find out that this is not an optimal location, so that unnecessary empty trips are scheduled. Therefore, we decided to take parking capacities only into account

insofar it is required for loaded vehicles waiting between arrival at the destination terminal and unloading. We use two steps to move from the preliminary schedule to the final schedule. First, we use improvement heuristics to reduce empty trips, for example by combining empty trips and load jobs. Second, we use a post-processing method to correct parking capacity infeasibilities. We will explain these steps in more detail in the next section.

We start the serial scheduling method as follows. All transport jobs to be scheduled are in the decision set D; there are no precedence relations between the jobs. The total number of vehicles is V and we have a set of terminals T. Each transport job entails a transport of cargo from terminal i to terminal j, with $i, j \in T$ with a given release and due time All terminals have limited docking, storage and parking capacity. For each resource, we define a profile, which expresses the available capacity of that resource at a specific terminal over time. Time is divided into discrete time intervals. Jobs should be scheduled before their latest departure time in order to have the possibility to be on time at the destination. The latest departure time does not take into account finite capacities. Our objective is to optimize due-time performance, or, stated otherwise, to minimize the number of late jobs. However, other objective functions, such as the total lateness, can be included in our method without major modifications.

2.4. Assumptions

In the remainder of this paper, we use the following assumptions:

- Cargo is always transported directly from origin to destination. Still, we note that multi-stage routes can be modeled by separate jobs with precedence constraints. For example, consider a job that has to be transported from terminal i to terminal j via terminal k. We can replace this job by two jobs, one from i to k and one from k to j, with the precedence constraint that the first job has to be finished before the second job may start. Precedence constraints can be included in the serial scheduling method, as the decision set D has to be updated every time a trip has been planned. So, the second job can be added to the decision set when the first job has been scheduled.
- The route between two terminals is fixed and chosen according to minimum distance.
- The terminals have separate in-buffers and out-buffers for storage of cargo.
- The storage space in the entire system is sufficient. Thus, when terminal i has limited buffer capacity, other terminals should have sufficient storage space to keep the products with destination i longer in storage.
- All docks are generic, i.e., they can be used for loading and unloading. Separate docks for loading and unloading can easily be included. This can be attractive to separate the incoming and outgoing flows at a dock.
- All cargo has the same loading/unloading time. This is a realistic assumption because load bearers are used in transporting the cargo and the dock operations are automated.
- All transport jobs are unit loads, so we do not take into account consolidation or pick-up and delivery trips of combined loads. In our application, consolidation is a separate decision process preceding vehicle scheduling; see Van der Heijden et al. (2002b).
- All handling (loading, unloading), travel and arrival times are deterministic. These times might actually be random variables, for example because of variations in travel time due to congestion. Thus, the assumption of deterministic information serves as an approximation. Usually, the variation in travel times is small relative to the travel time itself, and therefore a deterministic approach seems acceptable. Also, given that the loading and unloading of vehicles is completely automated, using load bearers that can be rolled on or of the vehicle, the loading and unloading times will only vary slightly. However, variations in travel time and docking time cause deviations between planning and realization.

2.5. Notation

In the next sections, we will use the following notation:

Input to the scheduling procedure

t^l_{release} the time job l is available for transport
t^l_{due} the time job l has to leave the destination terminal
DC^i_t the remaining dock capacity at terminal i at time t
IBC^i_t the remaining in-buffer capacity at terminal i at time t
IBC^i the maximum in-buffer capacity at terminal i
OBC^i_t the remaining out-buffer capacity at terminal i at time t
OBC^i the maximum out-buffer capacity at terminal i
PC^i_t the remaining parking capacity at terminal i at time t
τ_{ij} the travel time from terminal i to terminal j
l_i the loading time at terminal i
u_i the unloading time at terminal i
t^l_{LDT} the latest departure time for job l: $t^l_{\text{due}} - \tau_{ij} - l_i - u_j$ if there is no delay

Output from the scheduling procedure

t^l_{veh} the time a vehicle is available for job l
t^l_{load} the time a dock is available for job l to be loaded
t^l_{depart} the scheduled time job l has been loaded and the vehicle starts driving to the destination
t^l_{arrive} the scheduled time job l will arrive at the destination terminal
t^l_{unload} the scheduled time a dock will be available for unloading job l at its destination
t^l_{ready} the scheduled time job l will be unloaded and the handling will be finished

3. Constructing a schedule

When constructing a schedule, we select the highest priority job according to some logical priority rule and we schedule this job at the earliest resource feasible start time. To schedule the job, i.e., to determine the timing of all activities needed to process the job, we have to check all resource capacities. As mentioned in the introduction, the order in which the capacities are checked is important for efficient and effective scheduling. An important criterion in this respect is to check first the resource for which postponement of the activity later on (because another resource constraint is violated) does not cause new capacity problems. Because the resource requirements at the destination terminal depend on the timing of activities at the departure terminal, it seems logical to start at the departure terminal (timing is easier to determine there).

At the departure terminal, we check the vehicle capacity before the dock capacity. If a dock is available and no vehicle, we have to re-check the dock capacity at a later point in time, because it could be scheduled to unload an earlier job that arrived from another terminal. However, when a vehicle is available but no dock, we can easily postpone the docking operation, because we are sure that the vehicle will still be available.

At the destination terminal we also use a fixed order to check the capacity restrictions. The departure time of a vehicle determines its arrival time at the destination terminal. First, we look at the storage space, which has to be available from the finishing time of unloading until the due time. Once we know from when storage space is available, we also know the time interval in which we can unload the vehicle. Then we look at the docking capacity and determine the time at which we can unload the vehicle. The start of the unloading operation determines whether a parking place is required between arrival of the vehicle and the unloading operation. Therefore, it is natural to check the parking capacity after the unloading time has been determined. Now we discuss the steps in more detail.

I. Select the first priority rule and generate a schedule based on that rule

Step 0: Initialization

Derive initial resource profiles for vehicles, docks, parking places and storage locations from the status of the system, i.e., given that all jobs that are already started will be finished and that jobs are rescheduled if they have not been started yet.

Step 1: Selecting the highest priority job

A job l has to be selected from the decision set D according to a priority rule. We will address the following options:

- Earliest Due Time (EDT); that is, the job that has to be finished at the earliest point in time;
- Earliest Release Time (ERT); that is, the job that can start at the earliest point in time;
- Minimum Latest Departure Time (LDT); the latest departure time is defined as the latest time a job should be loaded in order to be sure that it will arrive at its destination before the due time; A minimum LDT rule might be more appropriate than an EDT rule, because the remaining processing time is taken into account.
- Minimum Slack; the slack is the still available time for transporting a job, i.e. the LDT minus the earliest departure time, which is the maximum of the release time and the current time (t_0).

We can use these four priority rules in a multi-pass approach, where in each pass a different priority rule is used. The EDT and minimum slack rule are also used in for example machine scheduling (see e.g. Enns, 1996). In selecting a job, we should not only look at the priority of the job, but also at the storage capacity at the terminals. The out-buffer capacity is taken into account in the next step, but the in-buffer capacity should be taken into account here. When the in-buffer capacity at terminal i will be exceeded within the planning horizon, we should first dispatch cargo from terminal i. Therefore, we have to take this into account in selecting a job. In selecting a job, we first select the terminal that will exceed its in-buffer capacity at the earliest point in time:

$$\min_{i\in T}\left\{\min_{t>t_0}\left\{t\middle|\mathrm{IBC}_t^i > \mathrm{IBC}^i\right\}\right\}. \tag{1}$$

From this terminal, we select the highest priority job. When there are no terminals with capacity problems, we select the job based on the priority rule only.

Step 2: Scheduling the highest priority job

(a) Determine the earliest resource feasible time. To determine the start time for the highest priority job, say job l, we have to check resource capacities: departure terminal capacity and destination terminal capacity.

A. Resource availability at the departure terminal. The earliest feasible start time for transporting job l from terminal i to terminal j, only looking at the resources at terminal i, is the time t for which the following conditions hold:

1. Job l is released.
2. A vehicle is available for transporting job l.
3. A dock is available for the loading time l_i.

The release time of job l is t^l_{release}. A vehicle has to be available for transporting job l. Let us define $t^l_{\mathrm{veh},k}$ as the earliest time a vehicle can be available for transporting job l at terminal i, arriving from terminal $k \in T$. A vehicle is available at a terminal k at time t when $V^k_{t'} > 0, \forall t' > t$, because the vehicle profile at terminal k should not become negative when we send this vehicle to terminal i. Then the earliest time a vehicle can be available at terminal i is

$$t^l_{\mathrm{veh}} = \min_{k\in T}\left\{t^l_{\mathrm{veh},k}\middle| V^k_{t'-\tau_{ki}} > 0,\ \forall t' > t^l_{\mathrm{veh},k} - \tau_{ki}\right\} \tag{2}$$

Note that we do not assign a specific vehicle. We only look at vehicle *profiles*, i.e., the number of vehicles available as a function of time. The selection of a specific vehicle is left to a lower control level (see Section 4). We also need a dock at terminal i for a time interval l_i, which is represented in Equation (3). This means that the earliest feasible start time given the resource constraints at terminal i is $t^l_{\mathrm{load}} = t^l_{\mathrm{dock}}$.

$$t^l_{\mathrm{dock}} = \min\left\{t\middle|\mathrm{DC}^i_{t'} > 0, \forall t' \in [t, t+l_i];\ t > \max\left\{t^l_{\mathrm{release}}, t^l_{\mathrm{veh}}\right\}\right\} \tag{3}$$

B. Resource availability at the destination terminal. The earliest starting time for transporting job l determines the earliest arrival time at destination terminal j: $t^l_{\mathrm{load}} + l_i + \tau_{ij}$. At the destination terminal, the following processes take place:

1. The vehicle possibly has to wait at a parking place until a dock is available.
2. A dock has to be available for the unloading time u_j.
3. Out-buffer capacity has to be available from the finishing time of the unloading operation, t^l_{ready}, until the due time of the job, t^l_{due}. At the due time, the customer collects the cargo.

Given these capacity restrictions, the earliest possible unloading time is the earliest time that all these capacity constraints hold

$$t^l_{\text{unload}} = \min \left\{ t \;\middle|\; \begin{array}{l} t > t^l_{\text{load}} + l_i + \tau_{ij} \\ PC^j_{t'} > 0,\ \forall t' \in \left[t^l_{\text{load}} + l_i + \tau_{ij}, t\right] \\ \text{DC}^j_{t''} > 0,\ \forall t'' \in \left[t, t + u_j\right] \\ \text{OBC}^j_{t'''} > 0,\ \forall t''' \in \left[t + u_j, t^l_{\text{due}}\right] \end{array} \right\}. \tag{4}$$

Because t^l_{load} is given in (4), there might not be a feasible solution with respect to the parking capacity. When there is no parking capacity available between arrival and unloading, the loading operation should be postponed. This leads to an iterative procedure that stops when a feasible solution has been found for both t^l_{load} and t^l_{unload}. When the time horizon is long enough, it is always possible to find such a feasible solution, but it can possibly lead to late delivery of job l: $t^l_{ready} > t^l_{\text{due}}$. Given that we always take the minimum over time, t^l_{unload} is the earliest feasible time that we can finish job l. Because job l has highest priority, this is exactly what we want. Note that Eq. (4) only takes parking capacity into account between the arrival of a loaded vehicle and the unloading of the vehicle. As mentioned before, we solve possible infeasibilities in a post processing method (see Step 4). In Fig. 2, we show a graphical representation for job l. Note that a vehicle is only claimed between t^l_{load} and t^l_{ready}.

Minimize waiting time and empty kilometers. It is possible that a scheduled job has to wait at the destination terminal until a dock is available for unloading (see Fig. 2). The waiting time w is equal to $t^l_{\text{unload}} - t^l_{\text{arrive}}$. During this waiting time, the vehicle cannot be used for another job. To increase vehicle utilization, we would like to minimize this waiting time. Given that the vehicle cannot be unloaded earlier, the only option is to postpone the loading operation, which can maximally be postponed by a time w. The dock and storage profile at terminal i have to be checked for the possibility to postpone the loading operation. Now we postpone the loading operation as much as possible in order to minimize the waiting time. Such a solution is still feasible with respect to parking, docking and storage capacity at the destination terminal.

Another improvement of the schedule might be possible with respect to the selection of a vehicle. Suppose that the earliest available vehicle is an empty vehicle that has to arrive from another terminal. When $t^l_{\text{load}} > t^l_{\text{veh}}$, it might be possible to use an empty vehicle from another terminal, that will arrive later but still in time, resulting in less empty kilometers. Maybe there is even a vehicle available at the terminal of origin of job l at time t^l_{load}. We select the nearest vehicle that can be available before time t^l_{load}.

(b) Try to combine job l with another job. When we would use the serial scheduling method in the way described above, there could be a lot of empty vehicle trips between terminals, while at the same time cargo has to be transported between these terminals. These jobs will have lower priority than job l, but it is probably beneficial to combine the empty trip with the transport of cargo. This reduces the number of empty trips and increases vehicle utilization.

By our definition, a job m can only be combined with job l if the destination of job m equals the origin of job l. If such a combination is possible, it prevents unnecessary empty vehicle movements, but it may also cause a delay to job l. To combine transportation, we determine the jobs that can be combined with job l, such that job l is still handled before its due time. Amongst all these jobs, select the job m that leads to the smallest delay for job l. Planning job m requires the same procedure as the one described for job l, taking into account the limited docking, parking and storage capacity. Note that when $t^m_{\text{ready}} > t^l_{\text{load}}$, job l will be delayed. Then it is necessary to reschedule job l, following

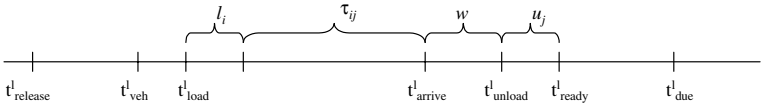

Fig. 2. Representation of job l.

the same procedure as in the previous step, but using $t^l_{\text{veh}} = t^m_{\text{ready}}$.

(c) Update the job list and the resource profiles. Remove job l from the decision set D and add job l to the set of scheduled activities S. In the case of combined transportation do the same for job m. Update all resource profiles.

Step 3: Termination criterion

Repeat Step 1 and 2 until all jobs have been scheduled.

Step 4: Correcting schedule infeasibilities

Infeasibility with respect to parking capacity. After unloading is finished a vehicle is available for another transport job. The vehicle can start loading, leave for another terminal or may have to park temporarily. The next assignment is done at a later stage in the planning procedure, when new jobs will be scheduled. Thus, only after all jobs have been planned, it is clear whether vehicles have to park at a terminal in between two jobs. But the parking capacity is limited, and therefore the whole schedule might be infeasible, i.e. $\text{PC}^i_t < 0$ for some $t > t_0$.

When we have a closer look at parking capacity problems, we see that parking may occur when:

- loaded vehicles are waiting for unloading;
- empty vehicles are waiting without any further assignment;
- empty vehicles are waiting for loading;
- empty vehicles are waiting to depart empty for another terminal.

Loaded vehicles that have to wait for unloading are taken into account by the planning procedure (cf. Eq. (4)). For the empty vehicles we try to solve this problem by postponing empty vehicle trips, by introducing additional empty vehicle trips, or by planning empty trips earlier.

Assume that there is a problem with parking capacity at terminal i at time t. This means that $V^i_t > \text{PC}^i$. First, we send empty vehicles that have no further assignment to the central parking area. A vehicle is not needed at terminal i anymore when $V^i_{t'} > 0$ for all $t' > t$. When all vehicles that are present at time t are needed in the future ($V^i_{t'} = 0$ for at least one $t' > t$), we try to exchange empty vehicles between terminals. An empty vehicle is not needed until the first time t that $V^i_{t'} = 0$. When we can send an empty vehicle to terminal j at time t, and another vehicle can arrive at terminal i from terminal j at time t', we have solved this infeasibility. Here we try to exchange an empty vehicle with the nearest terminal.

The last step we take to remove remaining infeasibilities is to plan empty vehicle trips at an earlier point in time. First, determine when the empty vehicle is needed for an assignment, i.e. the first time that $V^i_{t'} = 0$, because at that time the last empty vehicle is used for an assignment. When in the interval between time t and time t' an empty vehicle trip is planned, we can send this empty vehicle already at time t, instead of waiting till a later point in time. This might cause a parking capacity problem at the destination terminal, but at least we have shifted the capacity problems to a later point in time. Starting at the beginning of the time horizon and working towards the end of the planning horizon we try to solve all infeasibilities.

It might be impossible to solve all infeasibilities. For example, assume the scheduling method constructs a feasible schedule at time t. The realization in the simulation will differ from the proposed schedule because of traffic congestion and disturbances. The next time the scheduling method is called the starting situation, the prevailing system state, might be such that more loaded vehicles than the available parking capacity arrive at the same time at a terminal. This infeasibility cannot be solved by the scheduling method. Such problems should be solved locally in practice. For example, the vehicles can simply wait in front of the terminal until parking capacity is available. If no space in front of the terminal is available, it can be decided to dispatch the vehicle into a loop, with the same terminal as destination, so that the AGV drives around through the system until terminal capacity is available. Of course, the schedule should be such, that these events are exceptions and can be treated like exception handling, which is a different issue than planning and scheduling.

Infeasibility with respect to in-buffer capacity. The schedule can also be infeasible with respect to the in-buffer capacity. That is, the number in storage is larger than the available capacity. The available resources, docks, vehicles and out-buffer capacity should be sufficiently large to handle all jobs fast enough to prevent in-buffer problems. We

note that the in-buffer capacity is used as first criterion when prioritizing jobs. Therefore, the in-buffer capacity can only be exceeded if the resource capacities are insufficient. We will illustrate this problem in Section 6.

II. Multi-pass procedures

The serial scheduling method can be applied in a multi-pass procedure, where in each pass a different priority rule is used. Step I should be repeated for each priority rule.

III. Select the best schedule

In a multi-pass procedure, an objective function has to be present to evaluate the different schedules and to select the best schedule. Possible objectives are:

1. Minimize the number of late jobs.
2. Minimize the maximum or total lateness.
3. Minimize the empty kilometers or empty trips.
4. Maximize the total earliness.
5. Maximize the minimum slack.

Given the systems objective to maximize the service levels, we want to minimize the number of late jobs; this is our main objective. When all jobs are delivered at the destination on time, the first two objectives give the same result for each schedule. Then we can use a secondary objective to select one of the schedules, because the schedules will probably not be the same on the last three objective functions.

Summary of the scheduling steps

Summarizing, the following steps are distinguished in our (multi-pass) serial scheduling method:

I. Select the first priority rule and generate a schedule based on that rule:
   0 *Initialization*. From the status of the system, compute *initial resource profiles* for vehicles, docks, parking places and storage at terminals.
   1 *Select the highest priority job*. Pick a job from the decision set D based on in-buffer capacities and priority rule.
   2 *Schedule the highest priority job*
      (a) *Determine the earliest resource feasible time*. Let job l be the highest priority job selected in Step 2. Schedule job l at the earliest resource feasible time. Minimize waiting time and empty kilometers.
      (b) *Try to combine job l with another job*. When an empty vehicle has to arrive from another terminal, determine whether any of the other jobs on the list can be *combined* with job l.
      (c) *Update the job list and the resource profiles*. Remove job l from the decision set D and add job l to the set of scheduled activities S. In the case of combined transportation, do the same for job m. Update all resource profiles.
   3 *Termination criterion*. Repeat Step 1 and 2 until all jobs have been scheduled.
   4 *Infeasibility solving*. Try to solve parking capacity infeasibilities.

II *In case of a multi-pass procedure*. Select the next priority rule and repeat Step I until all priority rules have been used.

III *Schedule selection*. Choose the schedule that performs best on the performance measure(s) chosen.

4. Implementation in a dynamic context: A case study

4.1. Periodic rescheduling

The serial scheduling method plans the vehicles, given the system state and known transportation jobs. This can be seen as an off-line planning approach. A transportation system is very dynamic, new jobs arrive over time and disruptions due to failures or traffic congestion might change the system state considerably. In a rolling horizon approach, we can call the serial scheduling method either periodically (with a fixed time interval) or event triggered (at each job arrival). As the second option would lead to excessive planning time because of the high job arrival rate, we chose scheduling with a fixed time interval. One option to make a new schedule is to use the schedule from the previous planning and to try to insert the newly arrived jobs. Such a method is not really appropriate in this case, because the realization in the simulation will deviate from this schedule.

Thus, the previous schedule does not correctly present the system state anymore.

We chose to periodically construct a whole new schedule. New transportation jobs will only then be incorporated. Therefore, for such an approach to be fruitful, the planning period should not be too long, since otherwise too many new transport jobs are missed. However, it should also not be too short either, since otherwise the efficiency of the schedule might not be obtained. We will deal with this issue in Section 5 when we discuss the numerical experiments.

4.2. The Schiphol-case

To test the proposed serial scheduling method, we use a case study on a proposed underground transportation system near Amsterdam Airport Schiphol, the Netherlands, see Fig. 3 (cf. Van der Heijden et al., 2002b). This system focuses on the processing of time-critical products between five terminals at Amsterdam Airport Schiphol (AAS), with 2–3 docks per terminal, two terminals at the flower auction in Aalsmeer (VBA), with 5–6 docks per terminal, and a rail terminal in Hoofddorp (RT, with 8–10 docks). The terminals are connected by an underground tube system. Automatic Guided Vehicles (AGVs) carry cargo between terminals. Depending on order patterns about 125–175 AGVs are required. AGVs that are not needed for a while or for which there is no space at a local parking place are dispatched to a central parking area located just south of Schiphol Airport. Order patterns are time dependent and vary over days in the week and over hours on a day. On average there are about 150–200 jobs per hour, with peaks of more than 300 jobs per hour. Most of the system has two-way traffic; only the five terminals at Schiphol Airport are connected by a loop where only one-way traffic (counter clockwise) is possible. Travel times are up to 30 minutes and loading and unloading times are 2 minutes on average.

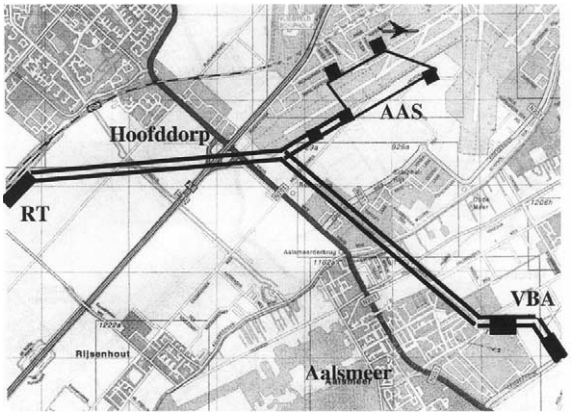

Fig. 3. Layout Schiphol case.

As mentioned before, the serial scheduling method only looks at profiles of available resource capacity at the terminals. The assignment of vehicles to docks and the selection of a vehicle for a transport job is done by the terminal managers. In this way, the terminal managers can respond to local disturbances, such as failures of vehicles or docks. The terminal managers try to follow the proposed schedule from the serial scheduling method as closely as possible. For an overview of the decision processes in such an automated transportation network, see Van der Heijden et al. (2002b).

5. Experimental setting

5.1. Goals of the experiment

To evaluate our serial scheduling procedure, we constructed a discrete event simulation model. We designed an experiment with the following three purposes. First, we are interested in the performance of our method compared to a much simpler approach, based on unconstrained scheduling with estimated waiting times to correct throughput times for finite resource capacities. Second, we are interested in the added value of a multi-pass approach compared to a single-pass approach. Third, we aim to find out to which extend we are able to satisfy resource constraints. As an example we consider storage capacity constraints.

We can explain the unconstrained scheduling procedure as follows. When a vehicle arrives at a terminal and no dock and/or storage space is available, the vehicle has to wait until both re-

sources are available. If no parking place at the terminal is available, the vehicle has to wait somewhere outside the terminal (for example, in front of the entrance), thereby possibly hindering other traffic, or it has to be dispatched to a new destination. During the waiting time incurred, the vehicle is not available to transport other jobs, so transport capacity is lost. However, we can implicitly take resource constraints into account by including an estimate for the delay in the handling times. If we know that there are capacity problems at a specific terminal during some time period, we can increase the handling times (loading, transport, unloading) by an estimate for the delay caused by waiting for capacity. As a consequence, all jobs that have to be transported to that terminal will be scheduled at an earlier point in time, because we take the expected delay into account. A simple approach to estimate the delay is by using exponential smoothing on the observed handling times.

Now the question is whether our more advanced serial scheduling method is better than such a rough approach to grasp the impact of finite resource capacities. Therefore, we compare our new procedure to the unconstrained serial scheduling approach, where we include the delay caused by finite resource capacities in the estimated handling times.

5.2. Experimental factors

In Table 1, we give an overview of the experimental factors. We compare our dedicated serial scheduling method (SRC) with the unconstrained serial scheduling method (S), as described above (cf. Van der Heijden et al., 2002a). Another experimental factor is the distribution of transportation flows. We distinguish three cases: one in which transportation flows between locations are equally distributed over all origin–destination pairs (Case 1), one in which most traffic is on one route, the route from VBA to RT, which should make the planning rather easy (Case 2), and one in which peak levels in transportation flows move quickly from one route to another, which means that it is essential to anticipate heavily fluctuating transportation demands (Case 3).

For the dock capacity, we distinguish two scenarios: the dock capacity is either low (L, dock utilization >80%) or ample (H, dock utilization <70%). For the parking capacity, we distinguish three scenarios. In the first scenario (L), there is only one parking place at each dock. In the second scenario (M) there is an additional number of parking places on the terminal, equal to the number of docks. In the last scenario (H) the additional parking capacity equals two times the number of docks.

For the storage capacity, we distinguish between a case with infinite capacity (there are no storage problems) and a case with finite capacity at the rail terminal (RT). In the finite capacity case, we limit the in-buffer capacity to 80% of the maximum occupation in the infinite capacity case with high dock capacity. For this case, we want to illustrate the difficulties that arise with limited in-buffer capacities.

As objective in the multi-pass scheduling approach, we minimize the number of late jobs using four priority rules: Earliest Due Time, Latest Departure Time, Earliest Release Time and Minimum Slack. When several schedules have the same number of late jobs, we use the total earliness

Table 1
Experimental factors

Factor	Range
Scheduling method	Serial scheduling with resource constraints (SRC), Serial scheduling (S)
Transportation flows	Case 1: balanced flows; Case 2: one heavy link; Case 3: location of peak changes quickly in time
Dock capacity	High (utilization <70%), low (utilization > 80%)
Parking capacity	High, medium, low
In-buffer capacity	Infinite, 80% of max. occupation

as a secondary objective. Some preliminary experiments indicated that an alternative secondary objective performs worse (cf. Section 3).

From some preliminary experiments, we derived the rescheduling frequency. Rescheduling every 20 minutes is too long, because in the meantime new jobs have arrived that have to be delivered at the destination terminal within 45 minutes. The performance of the algorithm is rather insensitive to rescheduling every 5 or 10 minutes. Based on these experiments, we chose a rescheduling period of 10 minutes. Van der Heijden et al. (2002a) found similar results with respect to rescheduling. In all cases, we select the number of vehicles such that the service levels in the high dock and high parking capacity case are between 90% and 100%.

6. Numerical results

6.1. Comparison constrained versus unconstrained serial scheduling

In this section, we present the results of the simulation experiments. In Table 2, we show service levels for the experiments with infinite storage capacity, i.e. the fraction of orders that is delivered before their due time. We use a code of two letters for the different scenarios. The first letter indicates the dock capacity, high (H) or low (L). The second indicates the available parking capacity, high (H), medium (M) or low (L). SRC is the serial scheduling method where resource constraints are taken into account, as opposed to the unconstrained method, S.

When the dock capacity is low, the SRC method performs significantly better than the unconstrained serial scheduling approach for Case 1 and Case 3. Case 2 is easier to plan, because most transport is on one route. There we see smaller differences between the two methods. When dock capacity increases, the performance difference between the methods decreases, but the SRC method still shows better results.

We see a significant performance difference between high and low dock capacity. Low dock capacity leads to more waiting times at the terminals. The vehicle utilization drops and this leads to worse results. To improve the performance, it is an option to increase the number of docks or the number of vehicles. Based on cost aspects a choice can be made between increasing the number of docks or the number of vehicles. It turns out that some parking capacity is required to have some flexibility on the terminals. Increasing the parking capacity further, from M to H, does not increase the service level.

6.2. Single pass versus multi pass

It is interesting to see how often a specific priority rule (ERT, LDT, EDT, Slack) is used in selecting a schedule. In Fig. 4, we see how often these rules are used in the different cases for the scenario of low dock capacity and medium parking capacity. The other scenarios showed similar results for the different cases. We see that in Case 1 and Case 2 all four priority rules have a share of about 20–30%. For Case 3, we see that the ERT-rule only about 5% of the time results in the best schedule of the four. The minimum LDT-rule and

Table 2
Service levels for the experiments without storage capacity restrictions

Flows →	Case 1		Case 2		Case 3	
Dock–Parking capacity ↓	SRC	S	SRC	S	SRC	S
LL	84.9	81.7	87.3	87.3	88.7	77.9
LM	88.8	85.5	89.0	88.5	98.3	96.6
LH	89.1	85.5	88.8	88.5	98.4	96.5
HL	93.1	92.6	91.5	91.3	99.4	99.2
HM	93.7	92.6	92.1	91.9	99.8	99.7
HH	93.7	92.7	91.9	91.8	99.8	99.7

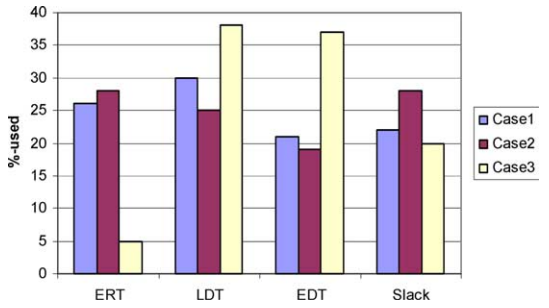

Fig. 4. Usage of the different priority rules in case of low dock capacity and medium parking capacity.

EDT-rule both have a share of about 35%. Probably, an ERT-rule only works well in relatively quiet time periods, because it does not take into account the priority of a job. Because there are almost no quiet time periods in Case 3, this can explain the low contribution of the ERT-rule. Fig. 4 does not show any redundancy of a priority rule in a multi-pass approach.

We also looked at the benefits of a multi-pass (MP) approach as opposed to a single pass (SP) approach. For the single pass experiments, we used the minimum LDT priority rule, because this rule is used most frequently in selecting a schedule (see Fig. 4). Constructing one schedule generally requires less than half a second of CPU time (Pentium III 500 MHz). Computing several schedules in a multi-pass is therefore no problem with respect to computation time. In Table 3, we show the results of these experiments.

The multi-pass approach shows significant improvements for Case 1 and Case 2. The results for Case 3 are in line with Fig. 4, which indicates that the minimum LDT rule is already used more than 35% in the multi-pass approach. We do not see a significant improvement in service levels for Case 3. Looking at all results, we can conclude that a multi-pass approach is more robust than a single pass approach.

6.3. Storage capacity restrictions

For storage capacity restrictions, we will illustrate to which extend our new serial scheduling method is able to meet storage capacity requirements. We will use Case 1 with low dock and low parking capacity, and the opposite scenario, high dock and high parking capacity. In this last scenario, it should be easier to fulfill the storage capacity restrictions. In the simulation, we limit the in-buffer capacity at the rail terminal (RTH) to about 80% of the maximum occupation in a scenario with no capacity limit and sufficient resources (HH). This means that the in-buffer capacity is set to 40 jobs. We compare the buffer occupation over time and we will see whether the scheduling method gives more priority to the RTH jobs.

In Fig. 5, we see the in-buffer occupation for the rail terminal for the two scenarios. When there is sufficient docking and parking capacity, we see that the in-buffer capacity limit will not be exceeded if the serial scheduling method takes the storage limit into account (HH-40). When there is little docking and parking capacity, it is impossible to handle the jobs at the rail terminal in time. The limit is exceeded a few times, because there are not sufficient resources to handle the jobs fast enough. In this case, it is impossible for the serial scheduling method to find a feasible solution. But the profile of LL-40 is much closer to the storage limit than the case where the serial scheduling does not take this in-buffer capacity into account (LL-INF).

Some in-buffer overflows are easier to solve than others. The ability to solve in-buffer problems depends on the available resources. When there are not a lot of high priority jobs on other routes, it is

Table 3
Service levels for a multi-pass (MP) and a single pass (SP) experiment

Flows →	Case 1		Case 2		Case 3	
Dock–Parking capacity ↓	MP	SP	MP	SP	MP	SP
LM	88.8	86.6	89.0	87.0	98.3	97.7
HM	93.7	92.4	92.1	90.6	99.8	99.9

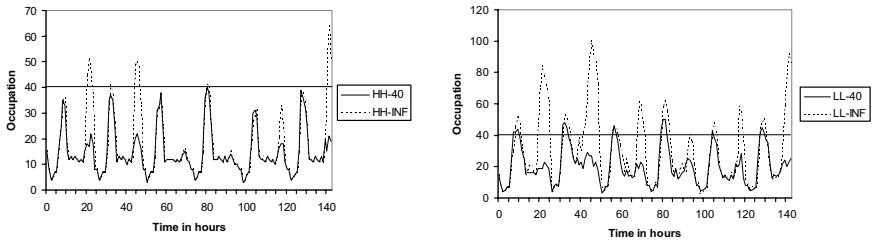

Fig. 5. Time dependent in-buffer occupation at the rail terminal in case of high dock and parking capacity (left figure) and low dock and parking capacity (right figure).

easier to attract vehicles from other routes, and the occupation of the in-buffer stays well below the limit, for example between 20 and 25 hours in Fig. 5.

When a system has limited storage capacity, it turns out that it is necessary to incorporate these capacity limits in the planning method. Otherwise, the capacity limits will certainly be exceeded (see Fig. 5 HH-INF and LL-INF). But to be able to comply with the capacity limitations, it is also necessary that sufficient other resources, such as docks and parking places, are available at the terminals.

7. Conclusions

From our simulation experiments, we draw the following conclusions. Including the multiple resource constraints in the serial scheduling method is possible and especially in highly constraint cases, significant improvements in service levels can be achieved. When constraints are less tight, the method performs at least as good as an unconstrained serial scheduling method. Note that the unconstrained procedure can be used because violation of resource capacities is possible at the expense of additional delay (except for storage capacity). If the latter is not true, only our new method provides a good quality, feasible solution. Also, we found that a multi-pass procedure combining various priority rules yields better results than a single pass procedure with the most attractive priority rule only.

In case of storage capacity restrictions, it is crucial to include these restrictions into the planning method. When there is a limited storage capacity of the in-buffer on a terminal, this terminal receives higher priority to ensure that cargo is transported before the capacity limit is reached. Sometimes it might be impossible to transport all cargo on time because of a large batch arrival or because at that moment the other resources at the terminal, docks, parking places and vehicles, are insufficient. Therefore, sufficient other resources should be available to be able to find a feasible schedule with respect to in-buffer capacity.

Because of the formalization of capacity constraints in resource profiles and our general description, the method can be used in other applications where similar resource constraints play an important role, for example in an AGV served flexible manufacturing system or an automated container terminal such as ECT in Rotterdam. Furthermore, our assumptions (cf. Section 2) do not pose a lot of restrictions on possible applications, and it is rather straightforward to relax some of these assumptions. For example, separate docks for loading and unloading can be included. Capacity profiles should be maintained for both dock types to present the remaining capacity of that type for each time period. For load jobs we can just look at the load dock profile, and for unload jobs at the unload dock profile.

Different types of cargo with different loading/unloading times can also easily be included in the model. The loading time (l_i) and the unloading time (u_i) have to be separated in a loading time and unloading time for each type (l_i(type), u_i(type)). These can then be used in Eqs. (3) and (4). It is not directly clear how one buffer, both for incoming

and outgoing cargo, should be included in the method. Jobs can only be transported when there is sufficient storage space at the destination terminal, but to be able to receive cargo, the destination terminal might have to send cargo away first. Since the arrival at the destination terminal is in a future time period, new jobs can have been planned in the meantime, and the storage problem might not even be present. Despite this, we think that a separate in- and out-buffer is an acceptable approximation of reality.

References

Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: Notation, classification, models and methods. European Journal of Operational Research 112, 3–41.

Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., 1995. Time constrained routing and scheduling. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), Network Routing. North-Holland, Amsterdam, pp. 35–139.

Enns, S.T., 1996. Finite capacity scheduling systems: Performance issues and comparisons. Computers and Industrial Engineering 30, 727–739.

Fisher, M., 1995. Vehicle routing. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), Network Routing. North-Holland, Amsterdam, pp. 1–33.

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

Gendreau, M., Guertin, F., Potvin, J.Y., Taillard, E., 1999. Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science 33, 381–390.

Ichoua, S., Gendreau, M., Potvin, J.Y., 2000. Diversion issues in real-time vehicle dispatching. Transportation Science 34, 426–438.

Kim, C.W., Tanchoco, J.M.A., Koo, P.H., 1999. AGV dispatching based on workload balancing. International Journal of Production Research 37, 4053–4066.

Kolisch, R., 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research 90, 320–333.

Powell, W.B., 1998. On languages for dynamic resource scheduling problems. In: Crainic, T.G., Laporte, G. (Eds.), Fleet Management and Logistics. Kluwer Academic Publishers, location, pp. 115–126.

Powell, W.B., 2003. Dynamic models of transportation operations. In: Graves, S., de Kok, A.G. (Eds.), Handbook in Operations Research and Management Science: Supply Chain Management. North-Holland, Amsterdam, pp. 677–756.

Powell, W.B., Jaillet, P., Odoni, A., 1995. Stochastic and dynamic networks and routing. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), Network Routing. North-Holland, Amsterdam, pp. 141–295.

Ulusoy, G., Bilge, U., 1993. Simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 31, 2857–2873.

Van der Heijden, M.C., Ebben, M.J.R., Gademann, A.J.R.M., Van Harten, A., 2002a. Scheduling vehicles in automated transportation systems. OR Spektrum 24, 31–58.

Van der Heijden, M.C., Van Harten, A., Ebben, M.J.R., Saanen, Y.A., Valentin, E., Verbraeck, A., 2002b. Using simulation to design an automated underground system for transporting freight around Schiphol Airport. Interfaces 32, 1–19.