Performance Evaluation of Real-time Scheduling Approaches in Vehicle-based Internal Transport Systems

Tuan Le-Anh, M.B.M. de Koster and Yu Yugang

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2006-063-LIS
Publication	November 2006
Number of pages	37
Persistent paper URL	http://hdl.handle.net/1765/8129
Email address corresponding author	rkoster@rsm.nl
Address	Erasmus Research Institute of Management (ERIM) RSM Erasmus University / Erasmus School of Economics Erasmus Universiteit Rotterdam P.O.Box 1738 3000 DR Rotterdam, The Netherlands Phone: + 31 10 408 1182 Fax: + 31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

Bibliographic data and classifications of all the ERIM reports are also available on the ERIM website: www.erim.eur.nl

ERASMUS RESEARCH INSTITUTE OF MANAGEMENT

REPORT SERIES

RESEARCH IN MANAGEMENT

ABSTRACT AND KEYWORDS	
Abstract	This paper studies the performance of static and real-time scheduling approaches in vehicle-based internal transport (VBIT) systems, which can be found in manufacturing and warehouse facilities. We propose three heuristic approaches for static VBIT problems (insertion, combined and column generation), extend them to a dynamic, real-time setting and compare their performance over a rolling time horizon. This time horizon can be seen either as a fixed-time interval in which advance information about loads' arrivals is available, or as a fixed number of loads which are known to become available in the near future. We also propose two dynamic assignment approaches: with and without look-ahead, respectively. Performance (primarily average load waiting time) of the above five dynamic scheduling approaches is compared with two nearest-vehicle-first rules (with and without look-ahead), which are the best vehicle dispatching rules known from literature and which are commonly used in practice. Experimental results show that, if sufficient prior information is available, our dynamic scheduling approaches consistently outperform vehicle dispatching rules. Results also reveal that guide-path layout, load arrival rate and variance, and amount of load pre-arrival information have strong impacts on the performance of vehicle control approaches. Column generation or the combined heuristics are recommended in small or medium-scale VBIT systems, whereas for large scale VBIT systems, both the combined heuristic and the dynamic assignment approach with look ahead perform best.
Free Keywords	Vehicle-based Internal Transport, Dynamic Scheduling, Dispatching, Material Handling
Availability	
Classifications	

PERFORMANCE EVALUATION OF REAL-TIME SCHEDULING APPROACHES IN VEHICLE-BASED INTERNAL TRANSPORT SYSTEMS

Tuan Le-Anh

RSM Erasmus University

P.O. Box 1738, 3000 DR Rotterdam

The Netherlands

René (M.) B.M. De Koster (Corresponding author)

RSM Erasmus University

P.O. Box 1738, 3000 DR Rotterdam

The Netherlands

Telephone : +31-10-4081719

Fax : +31-10-4089014

E-mail : rkoster@rsm.nl

Yugang YU

RSM Erasmus University

P.O. Box 1738, 3000 DR Rotterdam

The Netherlands

PERFORMANCE EVALUATION OF REAL-TIME SCHEDULING APPROACHES IN VEHICLE-BASED INTERNAL TRANSPORT SYSTEMS

Abstract

This paper studies the performance of static and real-time scheduling approaches in vehicle-based internal transport (VBIT) systems, which can be found in manufacturing and warehouse facilities. We propose three heuristic approaches for static VBIT problems (insertion, combined and column generation), extend them to a dynamic, real-time setting and compare their performance over a rolling time horizon. This time horizon can be seen either as a fixed-time interval in which advance information about loads' arrivals is available, or as a fixed number of loads which are known to become available in the near future. We also propose two dynamic assignment approaches: with and without look-ahead, respectively. Performance (primarily average load waiting time) of the above five dynamic scheduling approaches is compared with two nearest-vehicle-first rules (with and without look-ahead), which are the best vehicle dispatching rules known from literature and which are commonly used in practice. Experimental results show that, if sufficient prior information is available, our dynamic scheduling approaches consistently outperform vehicle dispatching rules. Results also reveal that guide-path layout, load arrival rate and variance, and amount of load pre-arrival information have strong impacts on the performance of vehicle control approaches. Column generation or the combined heuristics are recommended in small or medium-scale VBIT systems, whereas for large scale VBIT systems, both the combined heuristic and the dynamic assignment approach with look ahead perform best.

Keywords: vehicle-based internal transport, dynamic scheduling, dispatching, material handling.

1 Introduction

In many industrial facilities such as manufacturing plants, warehouses and transshipment terminals, vehicle-based internal transport (VBIT) systems (or VBITSs) are responsible for internal transport. In VBITSs, a control system dispatches vehicles (or automated guided vehicles - AGVs) using simple and intuitive online dispatching rules such as the nearest-vehicle-first (NVF) rule (Egbelu and Tanchoco 1984; Nakano and Ohno 1999; De Koster et al. 2004). An important practical reason for selecting simple vehicle dispatching rules is that they are easy to adapt for shop-floor control (SFC) systems or warehouse management systems (WMSs). Moreover, the dynamic and stochastic environments in which vehicles have to work and the relatively short travel times make such a vehicle dispatching approach dominant in VBITSs. Still, a vehicle scheduling approach with a rolling horizon and frequent rescheduling might lead to a better overall system performance than a dispatching approach, possibly at the expense of a substantial computation effort. If it works, such an approach may become more popular with the increasing application of computer-aided technologies in VBITSs. Scheduling approaches in static and real-time VBIT systems have hardly been investigated in literature.

A VBIT scheduling problem involves assigning a set of vehicles to transport a given set of loads within certain time-windows. In this paper, we make the following assumptions for studying VBIT systems: (*a*) vehicles operate continuously without breakdown; (*b*) there are no traffic problems (congestion, deadlock, etcetera.; this is not a real shortcoming as serious congestions will have been considered in the layout design of such systems); (*c*) all vehicles have uni-load capacity; (*d*) vehicles choose the shortest path to pickup and deliver loads; (*e*) loads are generated in batches of one; (*f*) there is sufficient space for waiting loads; (*g*) vehicles can always park at their drop-off locations; (*h*) and vehicle loading and unloading times are negligible. The

main objective of the scheduling problem in most real-life VBITSs is minimizing the average load waiting time (De Koster et al. 2004).

Although many researchers have studied the vehicle scheduling problem, have developed solution procedures and have compared scheduling methods with dispatching (see the literature section 3) in external transport, none of the research we found focused on developing scheduling-based approaches in a systematic comparison with dispatching rules for VBITSs. The scheduling problem in internal transport differs from the corresponding problem in external transport in several respects. For example, (1) the objectives of the two problems are different: minimizing the average load waiting time is the most important objective of a VBIT scheduling problem (see De Koster et al. 2004) while minimizing the vehicles' travel distances and the number of vehicles are usually objectives chosen for external transport systems (see Savelsbergh and Sol (1998), Laporte et al. (2000)); (2) travel times in VBIT environments are much shorter (this leaves little time for scheduling vehicles); (3) advance information about load arrivals in VBITSs is normally limited and less certain than in external transport systems (this leads to a shorter planning horizon and a higher rescheduling frequency); (4) vehicle parking policies are usually different (see assumption (g)). Because of these differences, there are no guarantees that dynamic scheduling approaches, successful in external transport, also perform well for VBITSs. In this paper, we therefore evaluate the performance of different dynamic vehicle scheduling approaches for VBITSs, depending on the amount and certainty of prior information.

In general, the VBIT scheduling problem can be formulated as a pick-up and delivery problem with time windows (PDPTW), in which a vehicle picks-up loads at some locations and delivers them to their destinations satisfying certain time-window restrictions. Since vehicles in most

VBITSs can transport only one (pallet) load at once, we reformulate the VBIT scheduling problem as a multiple traveling salesman problem with time windows (*m*-TSPTW) (section 2).

The *m*-TSPTW is an NP-hard problem (Desrochers et al. 1988). Depending on the load arrival rate, even a small instance of *m*-TSPTW can be very difficult to solve to optimality by commercial optimization software. Thus, it is impractical to apply optimal schedules in real-life vehicle scheduling problems. Therefore, in this paper, for solving static (offline) instances of the scheduling problems, we propose three heuristics which are later applied with rolling horizons (Psaraftis 1988). We also propose a look-ahead dynamic assignment algorithm for the real-time VBIT scheduling problem which is based on Fleischmann et al. (2004). The heuristics and the dynamic solution approaches are described in greater detail in sections 4 and 6.

In the static case, we numerically compare the performance (measured by average waiting time) of the three heuristic. In the real-time case, using simulation, we systematically compare the performance of the above scheduling methods with the performance of the *NVF* rule (two variants: with and without look-ahead - De Koster et al. (2004)), by varying several parameters such as guide-path layout, load arrival rate and load arrival variance. Our main contribution is therefore that we are able to indicate under which conditions, with which amount of pre-arrival information which method performs best in VBIT environments. It appears that, although dispatching is the dominant approach in practice, scheduling can bring substantial improvements, even with little pre-arrival information.

The rest of paper is organized as follows: the next section describes the mathematical formulation of the static and real-time VBIT scheduling problems; section 3 provides a literature review on vehicle scheduling; section 4 describes three heuristics for static scheduling problems; section 5 provides a performance evaluation of the proposed static scheduling approaches with

experiments; section 6 describes the dynamic scheduling approaches; section 7 provides a performance evaluation of the proposed dynamic scheduling approaches with experiments; section 8 summarizes the paper's results.

2 Mathematical formulation

For offline VBITS scheduling, we define a set of available vehicles (K) and a set of jobs (N) which need to be picked-up within time-windows $[e_p, l_p]$ ($p \in N$) and dropped-off at their delivery locations. The scheduling problem for VBIT systems can be formulated as a PDPTW. However, we reformulate this problem as an m-TSPTW by projecting time-windows at delivery locations to the corresponding pick-up locations (assuming a deterministic transport time) and logically considering a pick-up and a corresponding delivery job as a single job-node. If the time-window at the pick-up location is $[e_p, l_p]$, and at the delivery location is $[e_d, l_d]$, and the travel time between the two locations is t_{pd}, the time-window of the job-node will be $[e_n, l_n]$ with $e_n = e_p$, $l_n = \min(l_p, l_d - t_{pd})$. We assume that the time-window projection for job-nodes is always feasible ($[e_n, l_n] \neq \varnothing$). In many VBIT systems, only one-sided time-windows are present at pick-up locations (load release times, or r_p) and no time-windows are present at delivery locations, so $[e_n, l_n]$ is always $\neq \varnothing$. The travel time from job-node i to job-node j (t_{ij}) equals the travel time from the origin of job i (i^+) to the destination of i (i^-) ($t_{i^+i^-}$) plus the travel time from the destination of i to the origin of j ($t_{i^-j^+}$).

The m-TSPTW can be seen as a graph $G = (V, A)$, in which V is a set of vertices and A is a set of arcs. $V = \{0\} \cup N \cup \{n+1\}$, where $\{0\}(\{n+1\})$ denotes the depot (end depot) and $N = \{1,...,n\}$ is the set of (job-)nodes. $A = \{0\} \times N \cup I \cup N \times \{n+1\}$, where $I \subset N \times N$ is the set of arcs connecting job-nodes. $\{0\} \times N$ contains the arcs from the depot to job-nodes and $N \times \{n+1\}$ contains the arcs

from job-nodes to end depot (which is the same physical location as the depot in our computations). For each arc $(i,j)\in A$, there is an associated travel time (distance) t_{ij} and for each job-node i there is an associated time-window $[e_i, l_i]$. In the following, K is the set of vehicles and B is a big number. D_0^k, D_{n+1}^k are the starting time of vehicle k at the depot and the arrival time of vehicle k at the end depot respectively. Decision variables are: x_{ij}^k ($(i,j)\in A$, $k\in K$), which equals 1 if arc (i,j) is covered by vehicle k and 0 otherwise; D_i ($i\in N$) indicates the service start time of (job-)node i, with $e_i \le D_i \le l_i$.

As mentioned before, minimizing the average load waiting time is the most important objective of VBIT scheduling problems. The model formulation becomes then:

$$\textit{Minimize } \frac{1}{|N|}\sum_{i\in N}\left(D_i - e_i\right) \qquad (1)$$

subject to:

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1 \qquad \forall i \in N \qquad (2)$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0 \quad \forall i \in N, \forall k \in K \qquad (3)$$

$$\sum_{j\in N} x_{0j}^k = 1 \qquad \forall k \in K \qquad (4)$$

$$\sum_{i\in N} x_{i,n+1}^k = 1 \qquad \forall k \in K \qquad (5)$$

$$D_i + t_{ij} - D_j \le B\left(1 - x_{ij}^k\right) \qquad \forall i, j \in N, \forall k \in K \qquad (6)$$

$$D_0^k + t_{0j} - D_j \le B\left(1 - x_{0j}^k\right) \qquad \forall j \in N, \forall k \in K \qquad (7)$$

$$D_i + t_{i,n+1} - D_{n+1}^k \le B\left(1 - x_{i,n+1}^k\right) \quad \forall i \in N, \forall k \in K \qquad (8)$$

$$e_i \le D_i \le l_i \qquad \forall i \in N \qquad (9)$$

$$x_{ij}^k \quad \textit{binary} \qquad \forall i, j \in V, \forall k \in K \qquad (10)$$

Constraints (2)-(5) form a multi-commodity flow formulation. The constraint (6) indicates that if a vehicle k serves node j after node i, the constraint $D_i + t_{ij} \le D_j$ must be satisfied. Constraints (6)-(8) ensure feasibility of the schedule. Equations (9) and (10) are time-window and binary constraints.

During the execution of the algorithm in real-time scheduling problems, a vehicle may start at any load's drop-off location, not at the depot. Therefore, we must modify the formulations (1)-

(10) to reflect this change. By setting the current load's drop-off location as a virtual depot when a new rolling horizon begins, we can obtain a new formulation by replacing (4) and (7) with the following constraints: $\sum_{j\in N} x^k_{0_k j} = 1 \quad \forall k \in K$ and $D^k_{0_k} + t_{0_k j} - D_j \le B\left(1 - x^k_{0_k j}\right) \quad \forall j \in N, \forall k \in K$, respectively, where 0_k is the virtual starting depot of vehicle k, $\forall k \in K$.

In the formulations for the static and real-time scheduling problems, the number of binary and linear variables equals $|K|\times(|N|+2)\times(|N|+2)$ and $(|N|+2)\times|K|$ respectively. In principle, we can use general-purpose optimization packages such as CPLEX to solve the proposed model. However, such software can solve only small instances of the model, which makes them unusable for practical problems. We used CPLEX 7.1 to solve some instances of our problems (2 vehicles, 12 loads). On these instances, CPLEX 7.1 sometimes took more than 30 minutes (sometimes a few hours), and required much computer memory (>128 MB) to solve them. For real-time scheduling or medium-sized static situations, this is not acceptable. In this paper, we therefore propose some heuristics to cope with realistic cases.

3 Literature overview on the scheduling problem solutions

In the previous section, we have formulated the vehicle scheduling problem as an *m*-TSPTW (a special case of the PDPTW). In the literature, the PDPTW and the *m*-TSPTW have been studied extensively (Desrochers et al. 1988; Savelsbergh and Sol 1995). Desrochers et al. (1988) mention two main types of optimization algorithms for PDPTW: dynamic programming and branch-and-bound. Both methods are very time consuming and cannot solve practical problems within an acceptable time limit. Dumas et al. (1991) introduce an exact algorithm to solve PDPTW using a column-generation scheme. The sub-problem (or pricing problem) is a constrained shortest-path problem. Their algorithm can handle multiple depots and different vehicle types. Desaulniers et

al. (1998) propose a similar approach to solve multi-depot vehicle scheduling problems with time windows and waiting costs. In order to solve practical-size problems, they also propose a heuristic to speed up the branch-and-bound process. Savelsbergh and Sol (1998) propose some adaptations for speeding up the column-generation algorithm. They use several heuristics to generate columns with negative reduced costs and eliminate unattractive columns by sophisticated column management schemes. Besides set-partitioning and column- generation approaches, several heuristics have been proposed, such as saving heuristics (Kindervater and Savelsbergh 1992; Laporte et al. 2000).

Psaraftis (1988) provides a survey on solution approaches for dynamic vehicle routing problems. Two main approaches include an adaptation of the static solution and an implementation of static algorithms under a rolling horizon. Savelsbergh and Sol (1998) use the rolling horizon approach to solve a dynamic PDPTW. Gendreau et al. (1999) adapt the Tabu search approach which is used for the static problem to dispatch vehicles (trucks) dynamically. Powell (1996) considers the assignment problem of dynamically assigning drivers for a truckload motor carrier to handle loads that arise randomly over time. Powell and Carvalho (1998) use a logistics queuing network to solve a dynamic fleet management problem.

Yang et al. (2004) study a dynamic truckload PDP. They propose several benchmark local policies. They also propose two re-optimization policies (MYOPT and OPTUN) to solve the problem dynamically. The MYOPT policy solves a static instance each time when information about a new job arrival is received. OPTUN uses some opportunity costs based on probabilistic knowledge of future requests to improve the solution quality. Yang et al. (2004) prove that two re-optimization policies outperform local policies. Fleischmann et al. (2004) use a dynamic assignment algorithm to assign jobs to vehicles by minimizing the total cost due to empty moves,

loaded moves, waiting, and delay. They show that their approach is superior to assignment rules and some insertion algorithms. Kim and Bae (2004) propose a look-ahead dispatching method to dispatch AGVs at a container terminal, in which tasks must be carried out according to a fixed order. The main objective is to minimize the delay times of container cranes. They formulate the dispatching problem as a mixed-integer programming problem and propose a heuristic to solve it. They apply this heuristic dynamically to schedule AGVs. The dispatching heuristic is invoked each time an AGV becomes free. The dispatching procedure takes only limited tasks into consideration. Using simulation, they show that their look-ahead dispatching methods outperform the shortest-travel-distance first, earliest-due-date and revised shortest imminent operation dispatching rules.

The above survey shows that most real-time scheduling studies concern external transport. Kim and Bae's (2004) study is an exception. In this paper, we systematically compare the performance of different real-time scheduling approaches and dispatching rules for two experimental environments under various working conditions.

4 The static scheduling problem

In order to solve the static (or offline) scheduling problem in a VBIT system it in an acceptable computation time, we introduce three heuristic methods. The first one, insertion, is mainly used as a benchmark. We introduce a column-generation and a combined heuristic (a combination of existing heuristics designed to suit our problems). The *cost* of a vehicle tour is defined as the average load waiting time of the loads served in this tour.

- **Insertion heuristic**

The insertion heuristic (Van der Meer 2000; Laporte et al. 2000) is frequently used for real-time dynamic scheduling problems (Psaraftis 1988). Its main advantages are simplicity and calculation speed. We implement a least-cost insertion strategy where the cost corresponds to waiting time. In the static scheduling problem, all vehicles start at the depot and the insertion heuristic works as follow: (1) Initialize all vehicle routes at the depot node $\{0\}$, let the set S contain all (job-) nodes arranged in increasing order of the load (job) release times ($S \neq \varnothing$), set all tours' costs to zero. (2) Remove the first node from the set *S* and insert it into a specific tour with least cost, respecting the time-window constraints (6)-(9). By doing this, we expand vehicle routes gradually. (3) Repeat the above process until $S = \varnothing$, compute total cost, stop.

- **Combined heuristic**

<Insert Figure 1 here>

This heuristic starts with an initial solution created by the insertion heuristic and applies several improvement algorithms sequentially to improve the solution. We introduce well known improvement algorithms: *Re-insertion*, *Exchange* and *Relocation* (Kindervater and Savelsbergh 1992; Laporte et al. 2000) to the internal transport system. The three improvement heuristics are illustrated in Figure 1:

Re-insertion: The Re-insertion (or forward Or-exchange) algorithm works as follows: *Step 0*: set *it* (iteration index) to 1 (0 is the depot node). *Step 1*: remove the node (job) at position *it* and search for the best insert position while respecting the constraints of the proposed model from node *it*+1 to the end of the route. *Step 2*: if a cost reduction is found, then insert this node into the best insertion position, otherwise increase *it* by 1. *Step 3*: if node *it* is the last node in the route, stop. Otherwise go to Step 1.

Relocation: *Step 0*: set it_1 to 1 (node index for route 1), set *previous total cost* to total cost of routes 1 and 2. *Step 1*: find the best insert position of node it_1 in route 2. *Step 2*: if a cost reduction is found (*total cost of two new routes* < *previous total cost*), insert node *it1* of route 1 at the best insertion position in route 2. *Step 3*: increase it_1 by 1, re-compute total route cost, and set *previous total cost* to the new total route cost. *Step 4*: if all nodes in route 1 have been investigated, stop. Otherwise go to *Step 1*.

Exchange: *Step 0*: set it_1 to 1 (node index for route 1), set *previous total cost* to total cost of route 1 and route 2. *Step 1*: find the best exchange position of node it_1 and a node in route 2. *Step 2*: if a cost reduction is found (*total cost of two new routes* < *previous total cost*), exchange node *it1* of route 1 with the best exchange node in route 2. *Step 3*: increase *it1* by 1, re-compute the total route cost, and set *previous total cost* to the new total route cost. *Step 4*: if all nodes in route 1 have been investigated, stop. Otherwise go to *Step 1*.

The combined heuristics works as follows: (1) create initial (vehicle) routes using the *Insertion* algorithm; (2) apply the *Re-insertion* algorithm for initial routes; (3) apply the *Exchange* algorithm for every pair of routes of the previous step; (4) apply the *Relocation* algorithm for every pair of routes of the previous step, (5) apply the *Re-insertion* algorithm again for all routes of the previous step. STOP. It is clear that we should improve individual vehicle route at steps 2 and 5. We select the sequence *Exchange* (3) -> *Relocation* (4), since this sequence gives us a better solution (on average) than the reversed sequence.

According to Van der Meer (2000), the complexity of the *Insertion* algorithm is $O(n^2)$ (n is the total number of loads). Kindervater and Savelsbergh (1992) show that the complexity of the three improvement algorithms is $O(m^2)$ ($m \leq n$ and n is the maximum number of loads served by any vehicle route). In the above framework, we apply *Re-insertion* for all routes, so the worst

case complexity is $O(km^2)$ (k is the number of vehicles). Two other improvement algorithms are applied for all route pairs. The number of route pairs equals $k(k-1)/2$, so the worst case complexity of each assignment improvement algorithm applying for all pairs of routes is $O(k^2m^2)$. In conclusion, the overall complexity of the combined algorithm is $O(k^2m^2)$ which is $O(k^2n^2)$ in the worst case. For our scheduling problems, the number of loads served by a vehicle is about the same, so $m \cong n/k$. Hence, the average complexity is $O(k^2m^2) \cong O(n^2)$. Therefore, the complexity of the combined heuristic does not increase much in comparison with the insertion heuristic.

- **Column generation (heuristic)**

The number of columns (or feasible vehicle tours) for the m-TSPTW can be huge ($O(k \times n!)$), it is impossible to enumerate all columns in a reasonable time. Thus, we use the column generation approach to obtain only 'good' columns. The column-generation approach has been used by many authors for solving the PDPTW (Dumas et al. 1991; Savelsbergh and Sol 1998) and proved to be a very promising approach. In this study, we apply this approach to solve the m-TSPTW. In order to apply the column generation heuristic we re-formulate the m-TSPTW as a set-partitioning problem. This heuristic includes two steps: (1) generating columns for the master problem and (2) obtaining an integer solution.

Generating columns for the restricted master problem

The master problem (*set-partitioning problem*)

$$\textit{Minimize} \sum_{k \in K} \sum_{r \in S_k} c_r^k z_r^k \qquad (11)$$

subject to:

$$\sum_{k \in K} \sum_{r \in S_k} \delta_{ir}^k z_r^k = 1 \quad \forall i \in N \qquad (12) \qquad \sum_{r \in S_k} z_r^k = 1 \qquad \forall k \in K \qquad (13)$$

$$z_r^k = 0 \text{ or } 1 \qquad \forall k \in K, \forall r \in S_k \quad (14)$$

where: $z_r^k = 1$ if route $r \in S_k$ is selected, 0 otherwise; $\delta_{ir}^k = 1$ if job i is served on route $r \in S_k$, 0 otherwise; c_r^k: cost of route r served by vehicle k; S_k : set of routes for vehicle k; K: vehicle set. A route starts at the depot (or at the vehicle's drop-off location in the dynamic case) visiting some nodes (each node exactly once) within their time-windows and finishes at the end depot.

The set-partitioning model selects routes covering all nodes, each node exactly once, with minimal cost. The linear relaxation of this problem (binary constraint set (14) is replaced by $z_r^k \geq 0$) is called the *restricted master problem* (*RMP*). The optimal solution of the *RMP* is a lower bound on the objective value of the *integer master problem* (*IMP*).

The pricing problem (*shortest-path problem with time-windows*)

Let u_i ($i \in N$) be dual variables corresponding to the constraint set (12), and v_k ($k \in K$) be dual variables corresponding to the constraint set (13). According to the linear programming duality (Ahuja et al. 1993), z (a feasible solution of *RMP*) is optimal for the *RMP* if and only if for all $k \in K$ and $r \in S_k$ the reduced cost $d_r^k = c_r^k - \sum_{i \in N} \delta_{ir}^k u_i - v_k$ is nonnegative for all $k \in K$ and $r \in S_k$.

The pricing problem is $\min \left\{ c_r^k - \sum_{i \in N} \delta_{ir}^k u_i - v_k \mid k \in K, r \in S_k \right\}$, in which the cost of route $r \in S_k$ is $c_r^k = \sum_{i \in N} \left(D_{ir} - e_i \right) \delta_{ir}^k$ (D_{ir}: the service start time of node i in the route $r \in S_k$). This problem is a type of *shortest-path problem with time-windows* (SPPTW - Desaulniers et al. (1998)). If the solution of the pricing problem (z) results in $\min d_r^k \geq 0$, z is an optimal solution to the *RMP* and we are done. If z results in $d_r^k < 0$, we add the current solution z into the master problem. In this

research, we solve the SPPTW using the *generalized permanent labeling* (*GPL*) algorithm (Desrochers and Soumis 1988).

In many VBITSs, there are only one-sided time-windows at pick-up locations and no time-windows are required at delivery locations. In that case, we add artificial time-windows for nodes, since the *GPL* algorithm needs two-sided time-windows to perform.

The column-generation algorithm works as follows: (1) solve the *RMP* by the simplex algorithm (CPLEX); (2) get dual variables (u_i and v_k); (3) solve the pricing problem using the *GPL* algorithm. If the pricing problem's objective value ≥ 0, STOP. Otherwise, add the newly generated column into the *RMP* and go to Step 1. When the column-generation algorithm stops, we also get a good lower bound for the *IMP* (the optimal solution of the *RMP*).

Obtaining an integer solution

The algorithm in the previous column-generation step provides a set of columns for the *RMP*, which is now used to calculate an integer solution. We can obtain a good solution by solving the *IMP* with this set of columns. We may then improve the integer solution using improvement algorithms. In our implementation, we replaced (13) by $\sum_{r \in S_k} z_r^k \le 1$, (14) by $z_r^k \ge 0$ and (12) by a set of set-covering constraints ($\sum_{k \in K} \sum_{r \in S_k} \delta_{ir}^k z_r^k \ge 1$), since we found in the experiments that using a set-covering formulation leads to better overall solutions. We denote the new formation the *RMP'*.

Framework for column-generation heuristic

- *Step 1*: solve the *RMP'* by the column-generation approach. The optimal value of this problem is a lower bound for the *IMP*.

- *Step 2*: solve the *IMP* with the columns obtained in the previous step using CPLEX.
- *Step 3*: if the objective value equals the lower bound (obtained in Step 1), Stop. Otherwise improve the resulting solution using the improvement steps of the combined heuristic, STOP.

5 Performance evaluation for the static case

5.1 Experiment setups

- **System input parameters**

We selected two warehouse layouts for experimenting. Depending on function, several basic warehouse layout types exist. We select U- and I-layout type warehouses, which are very common in practice (Tompkins et al. 2003; Van der Meer 2000).

<Insert Figure 2 here>
<Insert Table 1 **here>**

In the U-layout, locations with transportation requests are more concentrated than in the I-layout (see Figure 2). In the latter layout, the receiving area is located further from the other areas. The distances between different areas are given in Figure 2. Table 1 shows the load flow matrices of the two layouts in percentage. In both layouts, loads needing transportation are generated at receiving, labeling and storage areas. Three load flows (from receiving to the storage areas, from the storage areas to labeling and from labeling to shipping) are kept identical in the numerical experiments in order to balance the load flows in the warehouses. The load flows (job nodes) are then generated randomly from the same load inter-arrival distribution (mean value τ). The resulting transport jobs are then executed using the three different methods. All experimental factors and their values are described below:

- Number of vehicles (K): 2 levels (6 and 2 vehicles).

- Load inter-arrival distribution (*Dist*): 2 levels (uniform and exponential),
- Load inter-arrival time (mean value τ): 2 levels (τ = 3, 8). This implies a variance of τ^2 for exponential and $\tau^2/3$ for uniform distributions. We combine τ = 3 with $|K|$=6 and τ = 8 with $|K|$=2. This leads to rather high vehicle utilization, which is typical in practice (fork lifts need to be manned). Low vehicle utilization leads to better performances of scheduling methods (see Table 3 and Table 5 in section 7).
- Time windows: 50 seconds.

▪ **Computation environments**

- All approaches have been coded in C++.
- For solving set-covering problems in the subproblems of column generation, we use CPLEX 7.1 from ILOG.
- All experiments were run on a Toshiba Satellite Pro 2100 notebook (CPU: Mobile Intel Pentium 2GHz, 256MB RAM).
- For each combination of experimental factors, we use 10 replications. The result is the average value of these 10 replications.

5.2 Computational results for the static case

Experimental environments are described in section 5.1 and the results are shown in Table 2.

<Insert Table 2 here>

From Table 2, we can draw the following conclusions:

- The column-generation heuristic obtains the best results overall at the expense of computation time. Its application in static settings is promising with the average gaps less than 10% in any case. But in online settings where the computation time is critical, its application may be limited in a small to medium scale warehouses. When the number of

vehicles increases to 15 or more, this heuristic may run half an hour or more depending on the problem according to our test.

- The combined heuristic performs not as good as the column-generation heuristic, but significantly outperforms the insertion heuristic without increasing running-times much. Potentially, the heuristic can be used for large scale VBIT problems when computation time is critical.
- In result, the column-generation heuristic is preferred in static cases. The combined heuristic is recommended for large-scale internal transport systems if the computation time is critical.

6 The real-time scheduling problem

- **Dynamic scheduling using three static heuristics**

In VBITSs, we may know information about load arrivals during a time period T in advance (this information may be not one hundred percent certain). Based on this information we propose two rolling-horizon strategies including rolling by time and rolling by the number of loads when the above three heuristics used in the dynamic scheduling case. When a vehicle starts to serve a load, it has to finish its current job. Cancellation of a job is not allowed.

Rolling by time horizon *(see Figure 3)*

For this rolling horizon policy (Psaraftis 1988), we schedule all (known) loads during a time period H ($0 < H \leq T$) using the three proposed heuristics (section 4). Depending on load arrival rates and load inter-arrival distributions during the operating period, the number of scheduled loads can differ significantly for a given time horizon H. The larger the load arrival rates are, the busier the considered VBIT systems are, the more number of loads a vehicle needs to serve for a given H. The busier a VBIT system is, the shorter we set the time horizon H. in order to prevent unnecessary job scheduling, which need updating before they have been executed.

Vehicles only follow the resulting schedule during a time period $h = aH$ ($a < 1$, normally 0.4 – 0.6). After every time period h the system invokes the scheduling algorithm again to schedule all known loads (excluding those in transport and for which vehicles are on their way for picking-up) in the period $[h, h + H]$. The process stops when all loads have been transported.

<Insert Figure 3 here>

Updating the problem formulation

The set N now contains loads which have not been served during the period $[t_l + h\ ,\ t_l + H]$ and loads that have release times satisfying: $t_l + H < e_j \le t_{l+1} + H$. A vehicle k becomes available at its last drop-off location at time $D^k_{0_k}$, which is the maximum of $t_l + h$ and the drop-off time of the last load served by vehicle k in the previous schedule.

Rolling by the number of loads *(see Figure 3)*

We propose a second rolling horizon policy - rolling by the number of loads. Suppose that during time period T, we know at least L loads in advance. This policy works as follows:

- Schedule M loads which are known in advance ($0 < M \le L$) using all three proposed heuristics (insertion, combined, and column generation heuristics),
- Re-schedule vehicles after the m^{th} load ($m = \lceil a*M \rceil$, $a < 1$) has been picked-up by solving the scheduling problem again for the next following M loads,
- Repeat this process until all loads have been transported. Stop.

Updating the problem formulation

For this type of the rolling horizon, we update the original formulation similar to the rolling by time approach. However, the set N now contains loads which have not been served in the current schedule execution ($M – m$ loads) and the next m loads.

- **Dynamic scheduling using an assignment algorithm**

Dynamic assignment scheduling (DAS)

An intuitive scheduling approach is to assign loads to all vehicles at each scheduling step, using an assignment algorithm. Fleischmann et al. (2004) use this approach to dynamically solve the full-truckload dispatching problem of a courier service. The main objectives in Fleischmann et al. (2004) include minimizing the order delay and the vehicle empty travel time. As we focus on minimizing the average load waiting time, we adopt new cost functions in our implementation. By introducing dummy loads or dummy vehicles (as in Fleischmann et al. (2004)) to balance the number of loads and vehicles for the assignment algorithm, we distinguish the following three types of costs:

- The cost of assigning a real vehicle to a real load (f_{main}) equals $C_{empt} \times Travtime$ plus $C_{wait} \times (Lwaittime)^{\alpha}$, in which *Travtime* is the vehicle travel time from its available location (current location for an idle vehicle or the vehicle's current load drop-off location for a busy vehicle) to a load release location and *Lwaittime* is the estimated waiting time of corresponding load.
- The cost of assigning a real vehicle to a dummy load is the unattractiveness cost of a location (vehicle waits at it current location) which is $C_{loc} \times 1$.
- The cost of assigning a dummy vehicle to a real load (load waits and remains unassigned at its release location) ($f_{urgency}$) equals $C_{urg}/(load\ release\ time + time\ window\ size - current\ time)^{\beta}$ if $(load\ release\ time + time\ window\ size) > (current\ time)$ and equals ∞ otherwise.

The values of the cost coefficients in our implementation are $C_{empt} = 10$, $C_{wait} = 2$, $C_{loc} = 5 \times 10^3$, $C_{urg} = 2 \times 10^7$, $\alpha = 2$, $\beta = 1$ or 2 (for I- and U-layout respectively). $\alpha > 1$ and $\beta \geq 1$ are used to increase the impact of large load waiting times and to urge timely pick up of long waiting loads

by real vehicles. Several of the cost coefficients are taken from Fleischmann et al. (2004) (C_{loc}, C_{urg}, α). Other cost coefficients are obtained from experiments. In our problem, we have only one-sided time-windows for loads and the cost function f_{main} is in favor of loads with smaller waiting times. This may lead to a very high value of the maximum load waiting time, so we introduce an artificial time-window for loads to guarantee an acceptable value of the maximum load waiting time. The general operating framework for the scheduling approach using the *DAS* algorithm is illustrated in Figure 4.

<Insert Figure 4 here>

Look-ahead dynamic assignment algorithm (LAS)

Obviously, the assignment algorithm works best for the case where we may assign about one load to each vehicle, but normally, with the implementation of Figure 4, we do not have enough loads to assign to all vehicles. In VBITSs, we may know some information about future load arrivals, which we could use to improve *DAS*. Ichoua et al. (2000) and De Koster et al. (2004) also use this idea in their studies. Therefore, we introduce a look-ahead dynamic assignment algorithm (*LAS*), which is a variant of *DAS*. *LAS* schedules vehicles using the same approach as *DAS*, however besides free loads the assignment algorithm also takes into account loads which are known to arrive during a look-ahead period T_L. A good length for T_L is the period during which about K (the set of vehicles) loads are known to arrive or in waiting ($T_L = |K| \times \tau$, τ is the load inter-arrival time). We may consider *LAS* a special case of the rolling by time policy in which H equals $|K| \times \tau$ and h equals min\{time that a new load arrives, time until the first vehicle drops-off its load\} from current time.

- **Vehicle dispatching rules**

We selected two dispatching rules, nearest-vehicle-first (*NVF*) and *NVF* with look-ahead, for comparison. These two rules are among the best rules for VBITSs (De Koster et al. (2004)).

Nearest-Vehicle-First (NVF)

According to the *NVF* rule, when a load enters the system, it places a move request; the shortest distance along the traveling paths to every available vehicle, is then calculated. The idle vehicle, whose travel distance is the shortest, is dispatched to the point of request. When a vehicle becomes idle, it searches for the closest load.

Nearest-Vehicle-First with look-ahead (NVF_LA)

NVF_LA operates similarly to *NVF*. The difference is that the load gives a signal Δ time units prior to its actual release time. The time between the actual release, and the virtual release Δ time units before, can be interpreted as a look-ahead time. This gives the vehicle the opportunity to travel to the load before the load is physically ready for transport. The vehicle can therefore arrive just before or after the load is ready for transport, thereby reducing load-waiting times.

7 Performance evaluation for real-time cases

7.1 Experiment setups

The experiment setups related to system input parameters, and computation environments are similar to those of the static case in subsection 5.1. There are two differences: we only consider a setting with |K|=6 (such a number of vehicles –like fork lifts- can typically be found in a warehouse). Varying the load inter-arrival time has similar effects as varying the number of vehicles. We have two levels for the load inter-arrival time (τ = 3, 3.6). These levels lead to rather high vehicle utilizations (80% or more). In environments with low vehicle utilizations scheduling will outperform dispatching.

- **Performance criteria**

In VBIT systems, the crucial performance criterion is minimizing the average load waiting time (*Avg_wait*). Following De Koster et al. (2004), we consider minimizing the average load waiting time as the main performance criterion. We also use other performance indicators: the maximum load waiting time (*Max_wait*), vehicle utilization (*Util%*), and the maximum number of loads in queues (*Max_inQ*) as side criteria. To rank the performance of the dispatching rules and the scheduling approaches we use Tukey's test (Hsu 1996) with 95% confidence level (CL).

- **Experimental approach parameters**

- Scheduling algorithms and dispatching rules:
  - Two variants of the *NVF* rule (*Disp. Rules*): *NVF* and *NVF_LA*. The best length of the look-ahead period (T_L) is taken. This value is estimated using simulation experiments.
  - Two variants of the dynamic assignment algorithm (*Assign. Algs*): *DAS* and *LAS* ($T_L = |K| \times \tau$),
  - Three heuristics, namely the insertion (*Insertion*), combined (*Com-Heur*) and column-generation (*Column-Heur*) heuristics under two rolling horizon policies: by time (T) and by the number of loads (M),
- Rolling horizon parameters (general values):
  - Rolling by the number of loads: $M = |K| \times 4$, $m = |K| \times 2$ ($M = 24$, $m = 12$). Scheduling 4 loads per vehicle on average leads to good scheduling results from column generation within a short computation time (less than 10 seconds).
  - Rolling by time: $H = |K| \times 4 \times \tau$, $h = |K| \times 2 \times \tau$ ($H = 72$ and 86.4, $h = 36$ and 43.2 corresponding to $\tau = 3$ and 3.6, respectively).

- The lengths of the planning horizons (simulation periods) are 900 ($\tau = 3$) and 1080 ($\tau = 3.6$) time units (seconds).

To limit the maximum load waiting time resulting of *DAS* and *LAS*, we introduce an artificial time fence (T_W). A T_W equaling about the value of the maximum load waiting time when *NVF* is used, appears to perform quite well.

7.2 Performance evaluation

The calculation results for U and I-layout are given in Tables 3 and 5 respectively for different combined parameters. In order to rank the scheduling methods, we apply a Tukey test with 95% confidence level using SPSS version 11 on the average load waiting time. The ranking results can be found in Table 4 and 6 for the U- and I-layout, respectively. Since the two rolling horizon policies (by *T* and *M*) perform quite similarly (see Table 3 and also the Tukey test), we use only one entry to represent both of them in Table 4 and 6. For example, the entry "column generation" represents both rolling horizon policies (by *T* and *M*) using the column-generation heuristic.

- **Performance evaluation for the U-layout**

<Insert Table 3 and Table 4 here>

From Table 3 and 4, some observations can be obtained as follows:

- When we schedule vehicles using two dynamic scheduling strategies: *Com_Heur Column_Heur,* the average load waiting time reduces dramatically compared to disptaching. Best results are obtained when we apply the column-generation heuristic to solve the instances of real-time scheduling problems. The largest improvement of the average waiting time of *Column_Heur* over *NVF* is 86.22% (uniform distribution, $\tau = 3.6$).

- Considering other side performance criteria (max load waiting time, max number of loads in queues, vehicle utilization), we also find that scheduling algorithms perform better than vehicle dispatching rules.
- *LAS* performs very well and is nearly as good as *Com_Heur*, particularly for the large load inter-arrival time cases ($\tau = 3.6$) with respect to average load waiting time. It is even better for the maximum load waiting time.
- Comparing *LAS* with the other two scheduling approaches (*Com_Heur* and *Column_Heur*) in side performance criteria, *LAS* performs worse in terms of the maximum number of loads in queues. *LAS* also results in a very high value of vehicle utilization. This is because *LAS* is a more local policy, implying that vehicles may travel longer distances for *LAS* than for the other scheduling approaches, which is similar to the observation of Kim and Bae (2004).
- The combined scheduling heuristic performs much better than insertion, for which the largest improvement is 42.2%.
- The *NVF_LA* and *LAS* perform significantly better than *NVF* and *DAS*, respectively since they have more information about future load arrivals.
- *DAS* performs a little better than *NVF* in general, but not significantly. *DAS* can make an assignment between multiple vehicles to multiple released loads at one time. The best assignment result of *NVF* (fixing one given load to its nearest vehicle) is only a feasible solution of *DAS* .
- By comparing all scheduling strategies with NVF, it can be seen that the lower the load arrival rates (or τ: larger load inter-arrival time), the bigger the improvements of the two scheduling approaches (Com_Heur, and *Comlum_Huer*), (see also Yang et al., 2004) This is

fairly obvious, since in highly utilized systems there is little gain in prematurely sending vehicles to pick-up locations as there are often loads in the neighborhood to be picked up.

- Scheduling and dispatching approaches perform relatively better for the uniform load inter-arrival distribution than for the exponential distribution. This can be explained by the fact that with the same mean interarrival time used in our experiments, the variance of the uniform distribution ($\tau^2/3$) is only one third of that of the exponential distribution (τ^2).

In conclusion, the column-generation heuristic performs better than the combined heuristic. However as we indicated in the static case, the running-time of the column-generation heuristic grows rapidly for large-scale real problems. So it is only suitable for small- and medium- scale instances (less than 15 vehicles), but for the large scale instances (especially more than 15 vehicles) and the computation time is critical, *LAS* and *Com_heur* are preferred.

- **Performance evaluation for the I-layout**

<Insert Table 5 **and Table** 6 **here>**

From Tables 5 and 6, and comparing them with Tables 3-4 for the U-layout, we observe similar phenomena for the I-layout of using different dynamic scheduling and dispatching strategies. The following differences can be observed: (1) *LAS* performs more impressively (in the top group in half of the cases from Tables 6). Especially when τ=3.6. (2) The largest improvement of the average waiting time of Column_Heur over *NVF* is 83.3% (uniform distribution, τ = 3.6). (3) In this layout, the performance of the *NVF_LA* rule is less impressive than in the U-layout. The improvement of the average waiting time of *NVF_LA* compared with that of *NVF* is only 27.7%. (4) The average load waiting time in the U-layout is smaller than the corresponding value in the I-layout.

- **Value of information and further discussion**

In order to identify which factors influence performance ranking of different approaches in two layouts, Table 7 and Table 8 give some selected results corresponding to some selected load look-ahead times, scheduling approaches, and system layouts. Since the dynamic scheduling heuristics behave similarly, only the combined heuristic is selected for experiments. The results of *NVF* and *DAS* are excluded due to their bad performance. The results are therefore only provided for *NVF_LA*, *LAS* and *Com_heur*. From Table 7 and Table 8, we obtain several important observations and their initial causes.

Value of look-ahead information. For each scheduling approach, we make the following four observations:

- The best look-ahead period values for *NVF_LA* are different in the two different layouts. They are between 2τ and 3τ for the U-layout and smaller (between 0.5τ and 2τ) for the I-layout. It seems difficult to recommend a specific value for the best length of the look-ahead period for different τ and layouts, however it is fairly small (less than 3τ), and can be determined by experiments.
- The best values of look-ahead periods for *LAS* are about the same for the two layouts, and equal to $|K|\times\tau$. Beyond $|K|\times\tau$ ($=6\times\tau$) time units little average waiting time reduction can be obtained. This value is reasonable since, for the assignment algorithm, it is logic to assign only one load for each vehicle. Look-ahead too far in advance cannot reduce the average load waiting time resulting from using *LAS*.
- The best values of look-ahead periods for the *Com_heur* are $4\times|K|\times\tau$ for both layouts. No further reduction of the average waiting time can be realized beyond $4\times|K|\times\tau$ ($24\times\tau$) time units, which is due to the fact that the algorithm plans about 4 loads ahead for each vehicle

with the given parameters. Similar to *LAS*, look-ahead too far in advance helps little in reducing the average load waiting time for this method.

In conclusion, for every approach, selecting an appropriate look-ahead period, decreases the average load waiting time significantly. The usable look-ahead time lengths (i.e. the best values of the look-ahead periods) are different for different approaches; *Com_heur* and *LAS* can better use larger look-ahead values than *NVF_LA*.

Which approach to select? For a given amount of prior load arrival information, or a given length of the look-ahead period (e.g. τ or 2τ), we have the following two observations:

- *LAS* always leads to better results than *NVF_LA* for both layouts, since *NVF_LA* can only assign one load to a (nearest) vehicle at a time, while *LAS* can assign multiple loads and vehicles at a time.
- *Com_heur* is usually better than *LAS* for both layouts, since *Com_heur* schedules more loads than *LAS* at a time. Still, *LAS* runs faster (less than a second computation time) and is easier to implement.

<Insert Table 7 **and Table** 8 **here>**

Influence of warehouse layouts. In warehouses, the receiving area is the main load generation source and the shipping area is the main sink. At the shipping area, vehicles become available after dropping off their loads. It can be considered as a main vehicle source. Since vehicles at the receiving area only pick-up loads, this area needs vehicle dispatches from other areas. In the I-layout, the receiving area is the area farthest from the shipping area. Therefore, this area may sometimes have difficulty qualifying for a vehicle dispatch from the shipping area (particularly when using *NVF* or *NVF_LA*). This may lead to a vehicle shortage at the receiving area and

explains the poor performance of the vehicle dispatching rules in the I-layout. De Koster et al. (2004) call this the 'remote-area' phenomenon, in which *NVF*-based rules perform poorly.

In conclusion to this section, the influences of main factors found are summarized in Table 9.

8 Concluding remarks

In this paper, we study real-time vehicle scheduling in internal transport systems. These systems can be characterized by a high degree of uncertainty, short travel times, stopping or parking positions spread over the building, and often high vehicle utilization rates. In practice, dispatching vehicles is the common method. Literature on external transport has shown that scheduling vehicles may lead to better performance than dispatching them. Applying this to internal transport does not automatically lead to similar results, as the objectives of external transport and the circumstances are often quite different. The available look-ahead information may be just too short to be of any value. Furthermore, optimal vehicle scheduling is time-consuming, if not infeasible. This paper is one of the first to investigate the potential contribution of scheduling methods for internal transport.

We propose a mathematical model for the VBIT problem, and introducing three heuristics for the static vehicle scheduling problem. We apply these static-situation heuristics dynamically under a rolling horizon (for which we use two variants). We also propose two easy-to-implement assignment methods for VBIT problems, with (LAS) and without look-ahead information (*DAS*), adapted from Fleischmann et al. (2004). For comparison, we introduce one of the best-performing dispatching rules, applied with and without look-ahead information (*NVF* and *NVF-LA*, respectively). Using simulation, we systematically compare and rank the performances

(measured by the average waiting time) of these seven methods (two dispatching and five scheduling, of which three are used with two different rolling horizon methods), by varying the following parameters: load inter-arrival distributions, load arrival variances (determined by the distributions and mean inter-arrival rates), and layouts (U and I-layout). Results show that the scheduling approaches (*Com-Heur, Column-Heur*, and *LAS*) perform significantly better than the dispatching rules. Depending on layouts and working conditions the improvements can be about 85%. *Com-Heur, Column-Heur*, and *LAS* are recommended for dynamic VBIT, but *Column-Heur* is not recommended for large scale dynamic VBIT due to its computation complexity.

We find that the two rolling horizon methods (rolling by time and by the number of scheduled loads) perform similarly. When sufficient load pre-arrival information is available (see Table 7 and 8), the scheduling with rolling horizon approaches perform significantly better than dispatching wit *NVF_LA*. Although *Com-Heur* and *Column-Heur* perform (slightly) better than *LAS* for a given level of pre-arrival information, the latter has the advantage of easy implementation and shorter calculation times.

The results obtained in this paper suggest some future research topics: (1) developing better (particularly faster) static-situation heuristics; (2) take the vehicle parking problem into account when generating vehicle schedules.

References

Ahuja, K., Magnanti, T.L., and Orlin, J.B. 1993. Network flows: Theory, algorithms, and applications. Prentice Hall, Inc.

De Koster, R., Le Anh, T. and Van der Meer, J.R. 2004. Testing and classifying vehicle dispatching rules in three real-world settings. *Journal of Operations Management,* 22 (4), 369-386.

Desaulniers, G., Lavigne, J. and Soumis, F. 1998. Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research,* 111 , 479-494.

Desrochers, J., Lenstra, J.K., Savelsbergh, M.W.P. and Soumis, F. 1988. Vehicle routing with time windows: optimization and approximation. In: *Vehicle Routing: Methods and Studies* (ed B.L.Golden and A.A.Assad (Eds.)). Elsevier Science Publishers, pp. 65-84.

Desrochers, M. and Soumis, F. 1988. A generalized permanent labeling algorithm for the shortest path problem with time windows. *INFOR,* 26 (3), 191-212.

Dumas, Y., Desrosiers, J. and Soumis, F. 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research,* 54 , 7-22.

Egbelu, P.J. and Tanchoco, J.M.A. 1984. Characterization of automated guided vehicle dispatching rules. *International Journal of Production Research,* 22 (3), 359-374.

Fleischmann, B., Gnutzmann, S. and Sandvoß, E. 2004. Dynamic vehicle routing based on online traffic information. *Transportation Science,* 38 (4), 420-443.

Gendreau, M., Guertin, F., Potvin, J.-Y. and Taillard, E. 1999. Parallel Tabu search for real-time vehicle routing and dispatching. *Transportation Science,* 33 (4), 381-390.

Hsu, J.C. 1996. Multiple comparisons: theory and methods. Chapman & Hall, UK.

Ichoua, S., Gendreau, M. and Potvin, J.-Y. 2000. Diversion issues in real-time vehicle dispatching. *Transportation Science,* 34 (4), 426-438.

Jeong, B.H. and Randhawa, S.U. 2001. A multi-attribute dispatching rule for automated guide vehicle systems. *International Journal of Production Research,* 39 (13), 2817-2832.

Kim, K.H. and Bae, J.W. 2004. A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation Science,* 38 (2), 224-234.

Kindervater, G.A.P. and Savelsbergh, M.W.P. 1992. Local search in physical distribution management. *Report EUR-CS-92-05.*

Laporte, G., Gendreau, M., Potvin, J.-Y. and Semet, F. 2000. Classical and modern heuristics for the vehicle routing problem. *Intl. Trans. in Op. Res.,* 7 , 285-300.

Nakano, M. and Ohno, K. 1999. Decomposition algorithm for performance evaluation of AGV systems. *Production and Operations Management,* 8 (2), 193-205.

Powell, W.B. 1996. A Stochastic Formulation of the Dynamic Assignment Problem with an Application to Truckload Motor Carriers, *Transportation Science,* **30**, 195-219.

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

Psaraftis, H.N. 1988. Dynamic vehicle routing problems. In: *Vehicle Routing: Methods and Studies* (ed B.L.Golden and A.A.Assad (Eds.)). Elsevier Science Publishers, pp. 233-248.

Savelsbergh, M.W.P. and Sol, M. 1995. The general pickup and delivery problem.

Transportation Science, 29 (1), 17-29.

Savelsbergh, M. and Sol, M. 1998. DRIVE: Dynamic routing of independent vehicles. *Operations Research,* 46 (4), 474-490.

Tompkins, J.A., White, J.A., Bozer, Y.A., and Tanchoco, J.M.A. 2003. Facilities planning, 3rd edn. John Wiley & Sons, Inc..

Van der Meer, J.R. 2000. Operational control of internal transport system. Ph.D Thesis. Erasmus University Rotterdam.

Yang, J., Jaillet, P. and Mahmassani, H. 2004 . Real-Time multi-vehicle truckload pickup and delivery problems. *Transportation Science,* 38 (2), 135-148.

List Figures and Tables

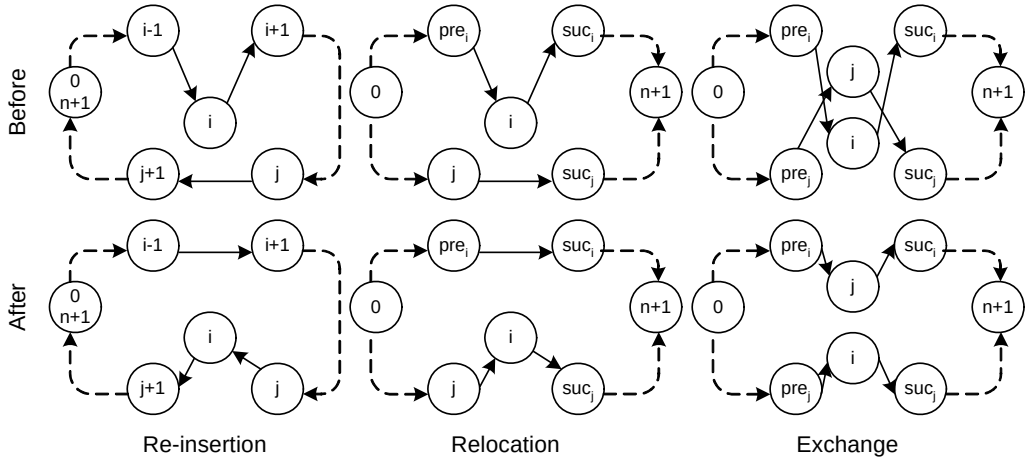


Figure 1. Improvement heuristic illustrations (Kindervater and Savelsbergh (1992)

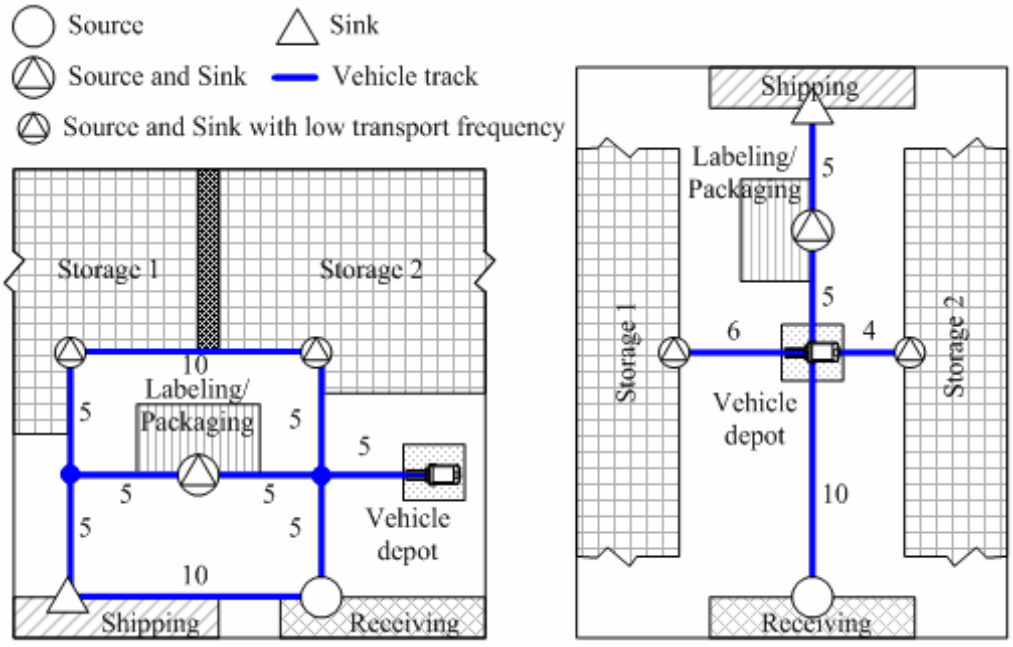


Figure 2. U-layout (left) and I-layout (right) used in experiments

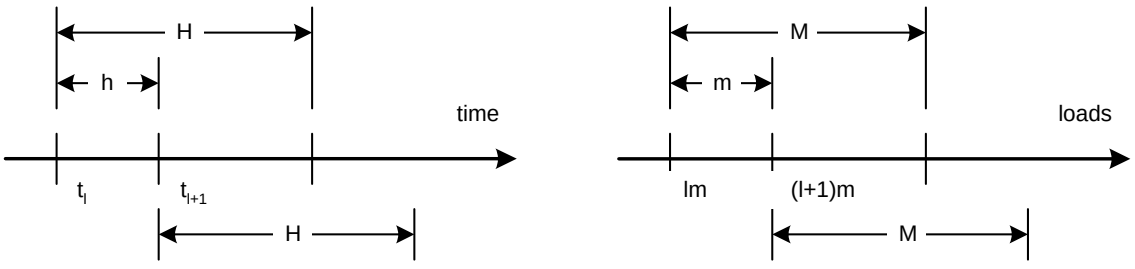
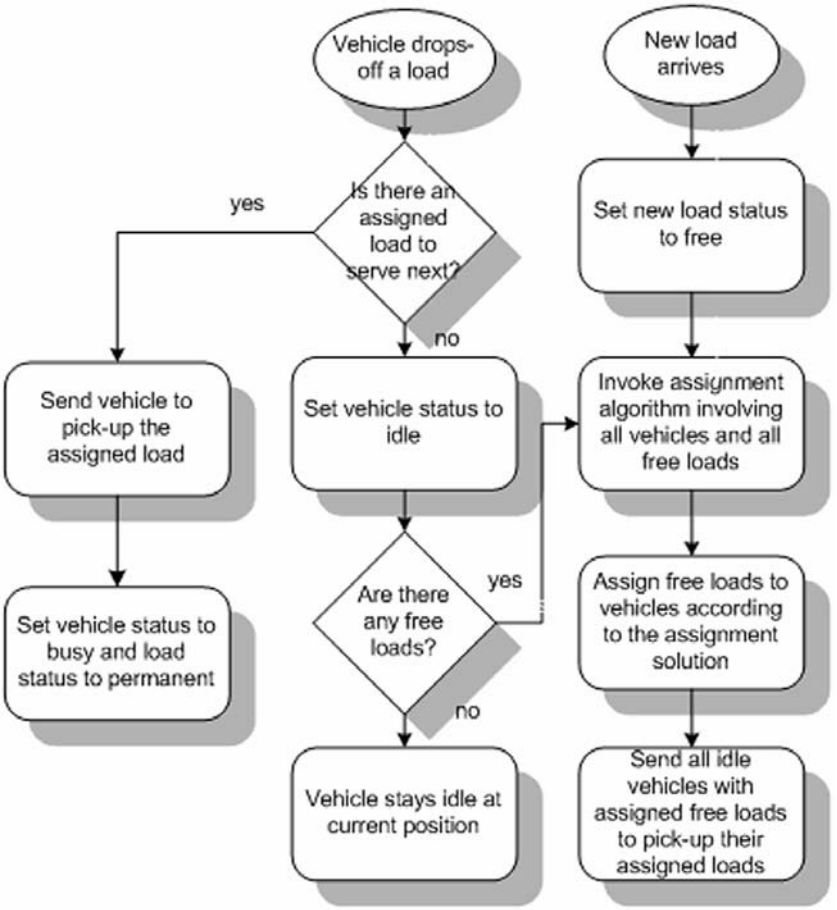


Figure 3. Rolling horizon illustration (by time - left and by the number of loads - right)



Free load: a load already arrived but not assigned to any vehicle or the assigned vehicle is still busy serving another load. A *busy vehicle* will be available at its current load drop-off location at drop-off time.

Figure 4. The general framework for the dynamic assignment algorithm

Table 1. Load flow matrices of the two layouts (in percentage %)

U layout									**I layout**						
Location		*0*	*1*	*2*	*3*	*4*	*5*			*0*	*1*	*2*	*3*	*4*	*5*
Depot	*0*	0	0	0	0	0	0		*0*	0	0	0	0	0	0
Receiving	*1*	0	0	50%	50%	0	0		*1*	0	0	50	50	0	0
Storage 1	*2*	0	0	0	0	50	0		*2*	0	0	0	0	50	0
Storage 2	*3*	0	0	0	0	50	0		*3*	0	0	0	0	50	0
Labeling	*4*	0	0	0	0	0	100		*4*	0	0	0	0	0	100
Shipping	*5*	0	0	0	0	0	0		*5*	0	0	0	0	0	0

Note: The real load flows are the percentages in the table multiplied by the load arrival rate ($1/\tau$) at the receiving area.

Table 2. Computational results (total waiting times) for the static case

		U-layout						I-layout					
		2 vehicles, 12 loads			*6 vehicles, 36 loads*			*2 vehicles, 12 loads*			*6 vehicles, 36 loads*		
IA		8			3			8			3		
Dist	*Alg*	*avg*	*gap%*	*RT(s)*	*avg*	*gap%*	*RT(s)*	*avg*	*gap%*	*RT(s)*	*avg*	*gap%*	*RT(s)*
uni	*ins*	**98.9**	**13.7**	< 0.1	**193.0**	**37.3**	< 0.1	**119.1**	**23.8**	< 0.1	**228.2**	**44.2**	< 0.1
	com	**92.5**	**7.7**	0.1	**152.4**	**20.6**	0.2	**97.7**	**7.2**	0.1	**167.2**	**23.8**	0.2
	col	**85.7**	**0.4**	1.5	**130.6**	**7.3**	45.2	**90.9**	**0.2**	1.3	**140.2**	**9.1**	55.9
	LB	**85.4**			**121.1**			**90.7**			**127.4**		
exp	*ins*	**132.4**	**20.6**	< 0.1	**240.6**	**33.9**	< 0.1	**134.5**	**17.2**	< 0.1	**239.9**	**32.4**	< 0.1
	com	**111.3**	**5.5**	0.1	**188.4**	**15.6**	0.2	**122.1**	**8.8**	0.1	**183.6**	**11.6**	0.2
	col	**106.1**	**0.9**	1.2	**166.7**	**4.6**	35	**112.7**	**1.2**	1.6	**171.6**	**5.5**	48.7
	LB	**105.2**			**159.0**			**111.4**			**162.2**		

IA, Dist: load inter-arrival time mean value (time units) and distribution; *Uni, Exp*: uniform, exponential distributions; *Alg*: algorithm; *ins, com, col*: insertion, combined and column generation heuristics; *LB*: lower bound originated from the column-generation algorithm; *avg*: average of total waiting time (time units) of ten problems corresponding to ten input data ; *gap%*: gap calculated by (current objective solution-lower bound)/ current objective solution ×100%; *RT*: running time (CPU time - seconds).

Table 3. Experimental results for the U-layout

			Disp. Rules		**Scheduling algorithms**							
					Assign. Algs		*Insertion*		*Com_Heur*		*Column_Heur*	
Dist	**t**	**perfor. measure**	*NVF*	*NVF_LA*	*DAS*	*LAS*	*T*	*M*	*T*	*M*	*T*	*M*
Uni	3	***Avg_wait***	**15.7**	**12.25**	**15.36**	**8.09**	**11.96**	**10.66**	**6.33**	**6.16**	**4.74**	**4.91**
		Max_wait	49.3	52.7	38.5	30.6	45.9	45.8	39.7	39.4	41	41.8
		Max_inQ	7	8	6	8	6	6	5	5	4	4
		Util%	95.99	92.19	92.65	98.68	94.74	94.86	93.08	93.09	91.23	92.04
		Imp%	-	21.97	2.17	48.47	23.82	32.10	59.68	60.76	69.81	68.73
	3.6	***Avg_wait***	**10.74**	**4.42**	**9.42**	**2.14**	**2.96**	**2.79**	**1.99**	**1.89**	**1.49**	**1.48**
		Max_wait	32.6	31.5	25.7	17.3	21.2	20.9	27.8	20	24.5	23.3
		Max_inQ	5	5	5	7	4	3	3	3	3	3
		Util%	86.65	86.21	79.22	96.83	84.25	84.25	82.63	82.83	81.91	81.93
		Imp%	-	58.85	12.29	80.07	72.44	74.02	81.47	82.40	86.13	86.22
Exp	3	***Avg_wait***	**19.51**	**16.48**	**22.52**	**14.58**	**14.98**	**14.55**	**10.7**	**10.37**	**8.17**	**9.14**
		Max_wait	68.2	68.7	53	43.7	47.4	48.7	46.9	46.3	47.4	46.9
		Max_inQ	9	10	8	9	7	8	7	6	6	6
		Util%	93.81	91.24	91.69	97.33	93.27	93.28	91.57	91.52	86.83	90.84
		Imp%	-	15.53	-15.43	25.27	23.22	25.42	45.16	46.85	58.12	53.15
	3.6	***Avg_wait***	**12.72**	**7.34**	**12.39**	**5.2**	**6.18**	**5.97**	**4.17**	**4.12**	**3.46**	**3.57**
		Max_wait	43.5	46.8	35.9	27.4	37.5	36.4	37.6	34.8	35.9	37.8
		Max_inQ	6	7	6	8	5	5	4	4	4	4
		Util%	83.18	82.55	78.75	94.44	82.84	83.03	81.26	80.92	78.7	80.32
		Imp%	-	42.30	2.59	59.12	51.42	53.07	67.22	67.61	72.80	71.93

Dist: the load generation distribution; *τ*: the load inter-arrival time; *Avg_wait*, *Max_wait*: the average and max load waiting time (time units); *Max_inQ:* the maximum number of loads in queues; *Util%*: the vehicle utilization; *Imp%*: (current Avg_wait-Avg_wait of NVF)/ Avg_wait of NVF×100%*; NVF*, *NVF_LA*: the nearest-vehicle-first rules without and with look-ahead; *DAS*, *LAS*: the dynamic assignment algorithms without and with look-ahead; Insertion: the (dynamic) insertion algorithm; *Com_Heur*, *Column_Heur*: the (dynamic) combined and column-generation heuristics; *T*, *M*: the two rolling schemes (by time and by the number of loads).

Table 4. Ranking of different scheduling policies for the U-layout (Tukey test 95 % CL)

Dist	*Uniform*							*Exponential*						
τ	3			3.6				3			3.6			
Column generation	1			1				1			1			
Combined heuristic	1				2			1				2		
LAS		3			2				3			2		
Insertion		3				4			3			2		
NVF_ LA		3					5		3			2		
DAS			6				6			6			6	
NVF			6				7			6			6	

Scheduling approaches are ranked from high to low according to the average load waiting time. The average load waiting times of scheduling approaches in the same number block are not significantly different.

Table 5. Experimental results for the I-layout

Dist	**t**	**perfor. measure**	**Disp. Rules**		**Scheduling algorithms**							
					Assign. Algs		*Insertion*		*Com_Heur*		*Column_Heur*	
			NVF	*NVF_LA*	*DAS*	*LAS*	*T*	*M*	*T*	*M*	*T*	*M*
Uni	3	***Avg_wait***	**40.1**	**36.11**	**27.71**	**17.73**	**19.2**	**18.47**	**12.8**	**12.45**	**10.57**	**10.4**
		Max_wait	204.2	189.4	59.3	49.1	49.3	49.3	49.2	49.5	49.2	49.5
		Max_inQ	19	18	9	10	8	8	7	7	6	7
		Util%	96.74	96.43	94.89	97.94	95.98	96.05	95.69	95.65	93.73	95.02
		Imp%	-	9.95	30.90	55.79	52.12	53.94	68.08	68.95	73.64	74.06
	3.6	***Avg_wait***	**14.73**	**10.64**	**13.27**	**3.29**	**4.87**	**4.91**	**3.04**	**3.04**	**2.46**	**2.46**
		Max_wait	66.5	70.1	34	22	32.5	28.8	35	33	34.4	33.4
		Max_inQ	7	8	6	7	4	4	4	4	4	4
		Util%	89.05	87.98	82.61	95.24	86.4	86.23	84.95	85.25	84.45	84.56
		Imp%	-	27.77	9.91	77.66	66.94	66.67	79.36	79.36	83.30	83.30
Exp	3	***Avg_wait***	**44.19**	**42.25**	**34.76**	**25.42**	**19.45**	**18.73**	**14.14**	**14.4**	**13.81**	**12.66**
		Max_wait	214	213.5	74.4	66.1	50	49.8	49.5	49.7	51.7	48.6
		Max_inQ	21	20	10	11	8	8	7	8	7	7
		Util%	95.89	95.68	93.48	96.89	94.31	93.93	94.04	94.06	93.57	93.51
		Imp%	-	4.39	21.34	42.48	55.99	57.61	68.00	67.41	68.75	71.35
	3.6	***Avg_wait***	**18.73**	**16.05**	**17.02**	**7.33**	**8.74**	**8.57**	**6.07**	**6.08**	**5.5**	**5.55**
		Max_wait	93.9	91.1	48.7	38.4	43.5	43.5	44.5	44.5	43.3	43.4
		Max_inQ	10	10	7	8	6	6	5	5	5	5
		Util%	87.03	86.73	81.74	93.4	85.32	84.73	83.5	83.81	83.1	83.29
		Imp%	-	14.31	9.13	60.86	53.34	54.24	67.59	67.54	70.64	70.37

Table 6. Ranking of different scheduling policies for the I-layout (Tukey test 95 % CL)

Dist	*Uniform*						*Exponential*					
τ	3			3.6			3			3.6		
Column generation	1			1			1			1		
Combined heuristic	1			1				2		1		
LAS		3		1				2		1		
Insertion		3		1				2		1		
NVF_ LA			5		5				5		5	
DAS			6			6			5		5	
NVF			6			6			5		5	

Table 7. The average load waiting times resulting of the three heuristics that use a look-ahead period (U-layout)

	LA_per	*Util%*	*0τ*	*0.5τ*	*τ*	*2τ*	*3τ*	*4τ*	*6τ*	*8τ*	*10τ*	*24τ*	*36τ*
NVF_LA	*Uni3*	96.0	15.7	14.6	13.3	12.5	12.3	14.3	17.9				
	Exp3	93.8	19.5	18.4	17.1	16.5	16.5	17.2	20.3		**		
	Uni3.6	86.7	10.7	9.4	7.9	5.5	4.4	5.1	7.3				
	Exp3.6	83.2	12.7	11.5	10.0	8.0	7.3	8.2	10.1				
LAS	*Uni3*	92.7	15.4	14.0	12.9	10.5	9.5	8.2	8.1	9.1	9.3		
	Exp3	91.7	22.5	22.8	19.7	17.3	15.6	14.4	14.6	14.1	14.8	**	
	Uni3.6	79.2	9.4	7.7	6.3	3.7	2.5	2.3	2.1	2.2	2.2		
	Exp3.6	78.8	12.4	10.9	9.3	6.5	5.4	5.1	5.2	5.0	5.2		
Com_Heur.	*Uni3*	93.1	*	*	10.4	10.0	9.3	9.4	8.0	7.7	7.4	6.2	6.5
	Exp3	91.6	*	*	15.2	14.1	13.9	13.9	12.7	12.5	11.4	10.4	10.4
	Uni3.6	82.6	*	*	2.2	2.1	2.0	2.1	2.0	2.1	2.0	1.9	1.9
	Exp3.6	81.3	*	*	5.3	4.9	4.7	4.8	4.7	4.6	4.4	4.1	4.1

Com-Heur: the combined heuristic; *LA_per*: length of the look-ahead time; *τ*: load inter-arrival time; *Uni3*: uniform load inter-arrival time ($\tau = 3$); (*) the combined heuristic does not work if loads' pre-arrival information is not available; (**): no further improvements found.

Table 8. The average load waiting times resulting of the three scheduling approaches with a look-ahead period (I-layout)

	LA_per	*Util%*	*0τ*	*0.5τ*	*τ*	*2τ*	*3τ*	*4τ*	*6τ*	*8τ*	*10τ*	*24τ*	*36τ*
NVF_LA	*Uni3*	96.7	40.1	36.1	39.6	44.7	55.6	72.1	78.2				
	Exp3	95.9	44.2	42.3	45.9	49.0	57.4	70.8	74.2		**		
	Uni3.6	89.1	14.7	12.9	11.7	10.6	14.2	29.3	40.9				
	Exp3.6	87.0	18.7	16.7	16.1	17.0	19.7	33.8	38.9				
LAS	*Uni3*	94.9	27.7	25.4	24.3	22.7	20.0	18.6	17.7	17.2	17.2		
	Exp3	93.5	34.8	33.1	31.5	29.9	28.3	27.0	25.4	25.6	25.2	**	
	Uni3.6	82.6	13.3	11.6	9.9	7.3	5.7	4.3	3.3	3.2	3.2		
	Exp3.6	81.7	17.0	15.3	13.8	11.4	9.7	8.2	7.3	7.2	7.4		
Com_Heur.	*Uni3*	95.7	*	*	17.7	17.6	16.3	16.2	16.0	15.1	14.4	12.4	12.4
	Exp3	94.0	*	*	18.3	18.1	17.1	17.1	17.0	16.4	15.6	14.4	14.0
	Uni3.6	85.0	*	*	4.1	4.1	3.5	3.5	3.5	3.3	3.3	3.0	3.0
	Exp3.6	83.5	*	*	7.4	7.3	7.1	7.1	7.0	6.5	6.5	6.1	6.2

Table 9. The impacts of main factors on vehicle control (dispatching and scheduling) policy performance

Factors	**Impacts**
Load arrival rate ↑ (*Vehicle utilization* ↑)	- The performance gaps between the dispatching rules and the scheduling approaches ↓
Load arrival rate's variance ↑	- All vehicle control policies' performances ↓
Layouts with remote areas	- The performance of *NVF*-based rules ↓
Horizon of pre-arrival information ↑	- all rules' performances ↑ initially. Too long look-ahead horizons do not help or even deteriorate performance.

Publications in the Report Series Research* in Management

ERIM Research Program: "Business Processes, Logistics and Information Systems"

2006

Smart Business Networks Design and Business Genetics
L-F Pau
ERS-2006-002-LIS
http://hdl.handle.net/1765/7319

Designing and Evaluating Sustainable Logistics Networks
J. Quariguasi Frota Neto, J.M. Bloemhof-Ruwaard, J.A.E.E. van Nunen and H.W.G.M. van Heck
ERS-2006-003-LIS
http://hdl.handle.net/1765/7320

Design and Control of Warehouse Order Picking: a literature review
René de Koster, Tho Le-Duc and Kees Jan Roodbergen
ERS-2006-005-LIS
http://hdl.handle.net/1765/7322

A Theoretical Analysis of Cooperative Behavior in Multi-Agent Q-learning
Ludo Waltman and Uzay Kaymak
ERS-2006-006-LIS
http://hdl.handle.net/1765/7323

Supply-Chain Culture Clashes in Europe. Pitfalls in Japanese Service Operations
M.B.M. de Koster and M. Shinohara
ERS-2006-007-LIS
http://hdl.handle.net/1765/7330

From Discrete-Time Models to Continuous-Time, Asynchronous Models of Financial Markets
Katalin Boer, Uzay Kaymak and Jaap Spiering
ERS-2006-009-LIS
http://hdl.handle.net/1765/7546

Mobile Payments in the Netherlands: Adoption Bottlenecks and Opportunities, or… Throw Out Your Wallets
Farhat Shaista Waris, Fatma Maqsoom Mubarik and L-F Pau
ERS-2006-012-LIS
http://hdl.handle.net/1765/7593

Hybrid Meta-Heuristics for Robust Scheduling
M. Surico, U. Kaymak, D. Naso and R. Dekker
ERS-2006-018-LIS
http://hdl.handle.net/1765/7644

VOS: A New Method for Visualizing Similarities between Objects
Nees Jan van Eck and Ludo Waltman
ERS-2006-020-LIS
http://hdl.handle.net/1765/7654

On Noncooperative Games, Minimax Theorems and Equilibrium Problems
J.B.G. Frenk and G. Kassay
ERS-2006-022-LIS
http://hdl.handle.net/1765/7809

An Integrated Approach to Single-Leg Airline Revenue Management: The Role of Robust Optimization
S. Ilker Birbil, J.B.G. Frenk, Joaquim A.S. Gromicho and Shuzhong Zhang
ERS-2006-023-LIS
http://hdl.handle.net/1765/7808

Optimal Storage Rack Design for a 3D Compact AS/RS with Full Turnover-Based Storage
Yu Yugang and M.B.M. de Koster
ERS-2006-026-LIS
http://hdl.handle.net/1765/7831

Optimal Storage Rack Design for a 3-dimensional Compact AS/RS
Tho Le-Duc, M.B.M. de Koster and Yu Yugang
ERS-2006-027-LIS
http://hdl.handle.net/1765/7839

E-Fulfillment and Multi-Channel Distribution – A Review
Niels Agatz, Moritz Fleischmann and Jo van Nunen
ERS-2006-042-LIS
http://hdl.handle.net/1765/7901

Leveraging Offshore IT Outsoutcing by SMEs through Online Marketplaces
Uladzimir Radkevitch, Eric van Heck and Otto Koppius
ERS-2006-045-LIS
http://hdl.handle.net/1765/7902

Buyer Commitment and Opportunism in the Online Market for IT Services
Uladzimir Radkevitch, Eric van Heck and Otto Koppius
ERS-2006-046-LIS
http://hdl.handle.net/1765/7903

Managing Supplier Involvement in New Product Development: A Multiple-Case Study
Ferrie E.A. van Echtelt, Finn Wynstra, Arjan J. van Weele and Geert Duysters
ERS-2006-047-LIS
http://hdl.handle.net/1765/7949

The Multi-Location Transshipment Problem with Positive Replenishment Lead Times
Yeming Gong and Enver Yucesan
ERS-2006-048-LIS
http://hdl.handle.net/1765/7947

Solving Lotsizing Problems on Parallel Identical Machines Using Symmetry Breaking Constraints
Raf Jans
ERS-2006-051-LIS
http://hdl.handle.net/1765/7985

Urban Distribution: The Impacts of Different Governmental Time-Window Schemes
H.J. Quak and M.B.M. de Koster
ERS-2006-053-LIS
http://hdl.handle.net/1765/8020

Leader-follower Game in VMI System with Limited Production Capacity Considering Wholesale and Retail Prices
Yu Yugang, Liang Liang, and George Q. Huang
ERS-2006-054-LIS
http://hdl.handle.net/1765/8194

Privacy Management Service Contracts as a New Business Opportunity for Operators
L-F Pau
ERS-2006-060-LIS
http://hdl.handle.net/1765/8110

Performance Evaluation of Real-time Scheduling Approaches in Vehicle-based Internal Transport Systems
Tuan Le-Anh, M.B.M de Koster and Yu Yugang
ERS-2006-063-LIS
http://hdl.handle.net/1765/8129