Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems

Tuan Le Anh, René B.M. de Koster, Yugang Yu

Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-0962
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	21-Oct-2009
Complete List of Authors:	Le Anh, Tuan; Electric Power University, Faculty of Management De Koster, René; RSM Erasmus University, Rotterdam School of Management Yu, Yugang; RSM Erasmus University, Dept of Management of Technology and Innovation
Keywords:	MATERIALS HANDLING, DISPATCHING RULES, FACILITY DESIGN, TRANSPORTATION, LOGISTICS
Keywords (user):	vehicle-based internal transport

Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems

Tuan Le-Anh[1], René B.M. de Koster[*2], Yugang Yu[2]
[1]*Faculty of Management, Electric Power University, Hanoi, Vietnam*
[2]*Rotterdam School of Management, Erasmus University, the Netherlands*

* Corresponding author. Email: rkoster@rsm.nl

Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems

This paper studies the performance of static and dynamic scheduling approaches in vehicle-based internal transport (VBIT) systems and is one of the first to systematically investigate under which circumstances, which scheduling method helps in improving performance.. In practice, usually myopic dispatching heuristics are used, often using look-ahead information. We argue more advanced scheduling methods can help, depending on circumstances. We introduce three basic scheduling approaches (insertion, combined and column generation) for the static problem. We then extend these to a dynamic, real-time setting with rolling horizons. We propose two further real-time scheduling approaches: dynamic assignment with and without look-ahead, respectively. The performances of the above five scheduling approaches are compared with two of the best performing look-ahead dispatching rules known from literature. The performance of the various approaches depends on the facility layout and work distribution. However, column generation, the combined heuristic, and the assignment approach with look-ahead consistently outperform dispatching rules. Column generation can require substantial calculation time but delivers very good performance if sufficient look-ahead information is available. For large scale systems, the combined heuristic and the dynamic assignment approach with look ahead are recommended and have acceptable calculation times.

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) based rule (De Koster et al., 2004; Egbelu and Tanchoco, 1984; Kim et al., 2007). 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 often very 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 for transport problems have been widely studied in external transport. Comparable studies on developing scheduling-based approaches for VBITSs are far less abundant. Some scheduling approaches, used in external transport, can be adapted for VBITSs. However, 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 usually 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 objectives usually chosen for external transport systems (Laporte et al., 2000; Savelsbergh and Sol, 1998); (2) travel times in VBIT environments are much shorter (this leaves little opportunity for (re)scheduling vehicles); (3) prior information on 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) in Section 3); (5) vehicles in most VBITSs can transport only one (pallet) load at once. Because of these differences, there are no guarantees that dynamic scheduling approaches, successful in external transport, also perform well for VBIT systems. Also, it is not clear under which circumstances these approaches can improve system performance, compared with existing dispatching rules for VBIT systems.

We adapt some well-known scheduling approaches, such as insertion and column generation appended with local search methods, to fit VBIT systems, and evaluate their performance, depending on the amount and certainty of prior information. In those internal environments, the requests for transporting loads are very stochastic. Still, in some cases, prior information on load releases is available, of which use can be made by look-ahead dispatching, and dynamic scheduling. We investigate the impact of some look-ahead information on performance (primarily defined as average load waiting time for pick-up) for different dispatching and scheduling approaches. The main contribution of the paper is to systematically investigate under which circumstances, which real-time dynamic scheduling method helps in improving performance in VBIT systems.

The studied 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-windows. We reformulate the VBIT scheduling problem as a multiple traveling salesman problem with time windows (*m*-TSPTW) (a special case of PDPTW) in section 3.

The *m*-TSPTW is an NP-hard problem (Desrochers and Soumis, 1988). Depending on the load arrival rate, even a small instance of the *m*-TSPTW can be very difficult to solve optimally 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 scheduling heuristics, and then apply them into instances with rolling horizons. We also propose a look-ahead dynamic assignment algorithm for the real-time VBIT scheduling problem which is based on Fleischmann et al. (2004).

In the static case, we numerically compare the performance (measured by average waiting time) of the three heuristics. In the real-time case, we systematically compare the performance of the above scheduling approaches with that 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. The results show that although dispatching is the dominant approach in practice, scheduling approaches can bring substantial improvements, even with little pre-arrival information. However, the performance gaps between different scheduling approaches vary with different circumstances. We recommend different approaches for different circumstances (particularly problem size and amount of look-ahead information).

The rest of paper is organized as follows: the next section reviews related literature; section 3 formulates the mathematical models for the static and real-time VBIT scheduling problems; section 4 describes three heuristics for static scheduling problems; section 5 provides a performance evaluation of the proposed static scheduling approaches with experimentation; section 6 describes the dynamic scheduling approaches; section 7 provides a performance evaluation of the proposed dynamic scheduling approaches with experimentation; section 8 summarizes the paper's results.

2. Literature overview on transport scheduling

We first discuss myopic dispatching methods for vehicles in VBIT systems, then we consider static and dynamic scheduling approaches used in external transport. We finish by discussing literature using such scheduling methods in internal transport.

A wide variety of dispatching rules are used for internal transport scheduling, including nearest-workstation-first (NWF), nearest vehicle first (*NVF*), modified first come first served

(MODFCFS), nearest vehicle first with time priority (NVFTP), shortest-travel-distance first, and reassignment based dispatching rules (De Koster et al., 2004; Kim et al., 2007; Le-Anh and De Koster, 2006). De Koster et al. (2004) test most of the above dispatching rules using layouts and data from three real-world settings including a distribution center, a production plant, and a container transshipment terminal. Their results show *NVF* and *NVF* with look-ahead (*NVF_LA*) are consistently among the best: According to the *NVF* rule, 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. *NVF_LA* additionally uses some information about future load arrivals to send vehicles to loads that still have to arrive. According to De Koster et al. (2004) significant improvements can be obtained using look-ahead information of arriving loads. 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 methods outperform dispatching rules, including shortest-travel-distance first. More studies considering look-ahead information can be found in Jang et al. (2001) and Mes et al. (2007).

The internal vehicle scheduling problem can be formulated as an *m*-TSPTW (and a special case of the PDPTW). In the literature, similar types of problems have been studied extensively (Desrochers and Soumis, 1988; Ohlmann and Thomas, 2007; Ropke et al., 2007; Savelsbergh, 1995; Thomas, 2002) for external transport systems. Several heuristics have been widely used,

including insertion heuristics (Bent and Van Hentenryck, 2006; Laporte et al., 2000; Morihiro et al., 2007) and some improvement heuristics: *Re-insertion*, *Exchange* and *Relocation* (Kindervater and Savelsbergh, 1992; Pisinger and Ropke, 2007; Xiang et al., 2006). Kindervater and Savelsbergh (1992) show that the complexity of the three improvement algorithms is $O(m^2)$ (m is the number of loads). The main advantages of these heuristics are simplicity and relatively fast calculation speed. Desrochers et al. (1988) distinguish two main optimization approaches for the PDPTW: dynamic programming and branch-and-bound. Both methods are very time consuming and cannot solve practical VBIT problems within an acceptable computation time. Dumas et al.(1991) introduce an exact algorithm to solve the PDPTW, using a column-generation scheme to decompose the problem into a series of subproblems. Each 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-sized problems, they additionally propose a heuristic to speed up the optimization process. Savelsbergh and Sol (1998) also 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. Several authors apply scheduling methods to external transport in a dynamic context. Savelsbergh and Sol(1998) use a rolling horizon approach to solve a dynamic PDPTW. Fleischmann et al. (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 some assignment rules and insertion algorithms. More examples can be found in literature (Chen and Xu, 2006; Powell, 1996).

There is some literature on scheduling methods for VBIT systems, including vehicle assignment (Bilge and Ulusoy, 1995; Vis et al., 2005). Besides minimizing load waiting time also other objectives are used, for example minimizing empty vehicle trip distances under a fixed block layout (Asef-Vaziri et al., 2008), and developing conflict-free routes (Krishnamurthy et al., 1993; Singh and Tiwari, 2002). Krishnamurthy et al. (1993) introduce a column-generation based heuristic for a VBIT system. Their research differs from ours. They focus on static problems; however, we focus on dynamic problems and consider the ranking of approaches.

Our brief survey shows most real-time scheduling studies concern external transport. Few authors use methods successfully deployed for dynamic external transport problems for internal transport. It is possible though to adapt such methods for internal transport and fit them to a dynamic context. This paper adapts successful scheduling approaches for VBITSs, and systematically compares their performance with dispatching rules for two layouts under various working conditions.

3. Mathematical formulation

We make the following assumptions for studying VBIT systems: (*a*) vehicles operate continuously without breakdown; (*b*) there are no traffic problems (like congestion or deadlocks). This is a common and reasonable assumption (see, Bilge and Ulusoy, 1995; De Koster et al., 2004; Kim and Bae, 2004), since vehicles travel along wide aisles in the two warehouse layouts we study, the number of vehicles is fairly low, and the vehicles are usually manned in this environment (forklift trucks with truck-mounted terminals), implying the drivers can avoid deadlocks; (*c*) all vehicles have unit-load capacity; (*d*) vehicles choose the shortest path to pick up 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; and (*h*) vehicle loading

and unloading times are fixed and considered in travel times between loading and unloading locations.

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. We formulate the scheduling problem for VBIT systems 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 $e_n = e_p$, $l_n = \min(l_p, l_d - t_{pd})$. 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 feasible ($[e_n, l_n] \neq \varnothing$). The travel time from job-node i to 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, $t_{i^-j^+}$.

Similar to the m-TSPTW, the VBIT problem can be seen as a graph $G = (V, A)$, in which V is a set of vertices and A is a set of arcs. We distinguish two fixed depot vertices one starting node $\{0\}$ and one ending node$\{n+1\}$. The vertex set $V = \{0\} \cup N \cup \{n+1\}$, with $N = \{1,...,n\}$ is the set of (job-)nodes. $A = \{0\} \times N \cup I \cup N \times \{n+1\}$, where $I \subseteq N \times N$ 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 the end depot. 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 $[e_i, l_i]$. 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 scheduling model, denoted by Model SP, becomes:

Model SP:

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

subject to:

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

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

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

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

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

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

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

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

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

In this formulation, B represents a sufficiently large positive number. 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 (7)-(8) ensure feasibility of the schedule. Equations (9) and (10) are time-window and binary constraints.

During the implementation 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 above model 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 constraints (4)

and (7) by $\sum_{j\in N} x^k_{0_k j} = 1 \quad \forall k \in K$ 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 only solve small instances in reasonable time. We used CPLEX 7.1 to solve small instances of our problems (2 vehicles, 12 loads). In some of them, CPLEX 7.1 took from 30 minutes to a few hours to solve, requiring much computer memory (>128 MB). 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.

4. The static scheduling problem

For the static (or offline) scheduling problem of the VBIT problem, we introduce three heuristic approaches. The first one, insertion, is mainly used as a benchmark. The other two are a column-generation and a combined heuristic (a combination of existing heuristics designed to suit our problems). The *cost* of a vehicle tour indicated below represents the average load waiting time of the loads served in the tour.

4.1 Insertion heuristic

The insertion heuristic (Laporte et al., 2000; Morihiro et al., 2007; Van der Meer, 2000) is frequently used for vehicle scheduling problems (Psaraftis, 1988). The insertion heuristic works as follows: (1) Initialize all vehicle routes by locating them at the depot node {0}, let the set, S contain all (job-) nodes arranged in an increasing order of the load (job) release times ($S \neq \varnothing$), and set all tour costs to zero. (2) Remove the first node from S and insert it into a specific tour

with least cost, satisfying the time-window constraints (6)-(9). By doing this, we expand vehicle routes gradually. (3) Repeat the above process until $S = \varnothing$, compute the total cost, and then stop. The complexity of the *Insertion* algorithm is $O(n^2)$ (Van der Meer, 2000; n is the total number of loads). Therefore, it is simple and has fast computational time.

4.2 Combined heuristic

This heuristic starts with an initial solution created by the insertion heuristic and sequentially applies three well-known improvement algorithms to improve the solution. We use *Re-insertion*, *Exchange* and *Relocation* (Kindervater and Savelsbergh, 1992; Laporte et al., 2000)

Re-insertion changes the sequence of job-nodes within one route. Every node in the route is taken from its original position, and reinserted between two other consecutive nodes in the same route if this reduces the cost of the route. If multiple reinsertions bring a cost reduction, the best insertion position, bringing the largest cost reduction, is accepted.

Relocation takes a node from one route, and relocates it to another route if this relocation reduces the summed cost of the two routes. Similar to Re-insertion, the algorithm selects the relocation position bringing the largest cost reduction.

Exchange swaps two nodes between a pair of routes if the swap (exchange) reduces the summed cost of the two routes. Similar to Re-insertion, the algorithm selects the exchange position bringing the largest cost reduction.

The Combined heuristic used here sequentially uses *Insertion, Re-insertion, Exchange, Relocation*, and *Re-insertion. Insertion* creates initial (vehicle) routes. The second *Re-insertion* is used to improve individual vehicle routes in the end. *Exchange* is used before *Relocation* as this sequence provides a better solution (on average) than the reversed sequence.

The complexities of *Re-insertion, Exchange, Relocation* are $O(km^2)$, $O(k^2m^2)$, and $O(k^2m^2)$, respectively (Kindervater and Savelsbergh, 1992); Therefore, the overall complexity of the combined algorithm is $O(k^2m^2)$ which is $O(k^2n^2)$ in the worst case ($m \le n$, k is the number of vehicles, and n is the maximum number of loads served by any vehicle route). Therefore, the complexity of the combined heuristic does not increase much in comparison with the insertion heuristic.

4.3 Column generation heuristic

The number of columns (or feasible vehicle tours) for Model SP can be very large ($O(k \times n!)$). It is impossible to enumerate all columns in an acceptable computational time. We here use the column generation approach to obtain only 'good' columns. The column-generation approach has been used by many authors for solving the PDPTW (Bronmo et al., 2007; Dumas et al., 1991; Savelsbergh and Sol, 1998) and has proven to be a very promising approach. In this study, we apply this approach to solve Model SP. We re-formulate Model SP as a set-partitioning problem. This heuristic includes two steps; Step 1 generates columns for the master problem and Step 2 obtains an integer solution.

Step 1: the master problem is the set-partitioning problem, which can be formulated as follows:

Model *MP*:

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

subject to:

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

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

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 is the cost of route r served by vehicle k; S_k is the set of routes for vehicle k; K is the 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.

Model *MP* selects routes covering all nodes, each node exactly once, with minimal cost. To solve the model, we relax constraint (14) to be $z_r^k \geq 0$, and refer to the corresponding model as the *relaxed master problem* (*RMP*), denoted by **Model *RMP***. The optimal solution of the relaxed model provides a lower bound for Model *MP*.

We now have to generate feasible columns. This can be done by solving a *pricing problem (shortest-path problem with time-windows)*._Let u_i ($i \in N$) be dual variables corresponding to constraint (12), and v_k ($k \in K$) be dual variables corresponding to constraint (13). According to linear programming duality (Ahuja et al., 1993), z (a feasible solution of Model *RMP*) is optimal for Model *RMP* if 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 $c_r^k = \sum_{i \in N} (D_{ir} - e_i)\delta_{ir}^k$ (D_{ir} is 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). The SPPTW is solved by the *generalized permanent labeling* (*GPL*) algorithm (Desrochers and Soumis, 1988). This is a dynamic programming shortest path algorithm with a single resource constraint. If the solution of the pricing problem (z) results in $\min d_r^k \geq 0$, z is an optimal

solution to Model *RMP* and we are done. If z results in $d_r^k < 0$, we add the current solution z into the master problem, Model *MP*.

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.

In sum, the column-generation algorithm works as follows: (1) solve Model *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 Model *RMP* and go to Step 1. When the column-generation algorithm stops, we also get a good lower bound for Model *MP* (the optimal solution of Model *RMP*).

Step 2: *obtaining an integer solution.* The algorithm in the previous column-generation step provides a set of columns for Model *RMP*, which is now used to calculate an integer solution. We can obtain a good solution by solving Model *MP* 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 the set of set-covering constraints leads to better overall solutions within the same computational times. We denote this new model as Model *RMP'*.

Framework for the column-generation heuristic. Solve Model *RMP'* by the column-generation approach. The optimal value of this problem is a lower bound for Model *MP*. Next, solve Model *MP* with the columns obtained in the previous step using CPLEX. If the objective value equals

the lower bound, stop and output the final results. Otherwise improve the solution using the combined heuristic, and output the final results.

5. Performance evaluation for the static case

5.1 Experimental conditions

We selected two basic warehouse layouts for experimentation, U- and I-layout type warehouses. Both are very common in practice (Tompkins et al., 2003; Van der Meer, 2000).

<Insert Figure 1 here>
<Insert Table 1 here>

In the U-layout, locations with transportation requests are more concentrated than in the I-layout (see Figure 1). In the latter layout, the receiving area is located further from the other areas. The distances between different areas are given in Figure 1.

Table 1 shows the load flow matrices of the two layouts by 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 values (6 and 2 vehicles).
- Load inter-arrival distribution (*Dist*): 2 types (uniform and exponential),
- Load inter-arrival time (mean value τ): 2 values ($\tau = 3, 8$). This implies a variance of τ^2 for exponential and $\tau^2/3$ for uniform distributions. We combine $\tau = 3$ with $|K|=6$ and $\tau = 8$ with

|*K*|=2. This leads to rather high vehicle utilization, which is typical in practice (fork lifts need to be manned).

- Time windows: 50 seconds.

All approaches have been coded in C++. We use CPLEX 7.1 from ILOG for solving set-covering problems in the subproblems of the column generation. All experiments are 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

With the experimental conditions in section 5.1, we obtain the results in Table 2, and draw the following conclusions:

<Insert Table 2 here>

The column-generation heuristic obtains the best overall results at the expense of computational time. Its application in static settings is promising with the average gaps less than 10% in any case. However, if 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 does not perform as well as the column-generation heuristic, but significantly outperforms the insertion heuristic without greatly increasing computational -times. Potentially, the heuristic can be used for large scale VBIT problems when computational time is critical. Based on these results, the column-generation heuristic is preferred in static cases, while the combined heuristic is recommended for large-scale internal transport systems if the computational time is critical.

6. The real-time scheduling problem

6.1 Dynamic scheduling using three static heuristics

In VBITSs, we may have a priori information about load arrivals during a time period T (this information changes over time). Based on this information we propose two rolling-horizon strategies, rolling by time and rolling by the number of loads, when the above three heuristics are used in the dynamic scheduling case. We assume before a vehicle can start to serve a load, it has to finish its current job. Cancellation of jobs is not allowed.

Rolling by time horizon *(see Figure 2(a)).* We schedule all (known) loads during a time period H ($0 < H \leq T$) using the three heuristics proposed in 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, and the more 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.

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 transit and those already assigned to vehicles) in the period $[h, h + H]$. The process stops when all loads have been transported.

<Insert Figure 2 here>

Implementation in Model MP

Every time $t_{l+1} = t_l + h$ Model MP has to be solved, the set N contains loads with release times before $t_l + H$ which have not yet been served or scheduled and loads that have release times satisfying: $t_l + H < e_j \leq t_{l+1} + h$. A vehicle k becomes available at its last drop-off location at time

$D_{0_k}^k$, 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 2(b)).* Supposing that during time period T, we know at least L loads in advance. This policy schedules M loads which are known in advance ($0 < M \leq L$) using all three proposed heuristics (insertion, combined, and column generation heuristics). Then 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.

Implementation in model MP

The implementation is 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.

6.2 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 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 $C_{empt} \times Travtime + C_{wait} \times (Lwaittime)^{\alpha}$, 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 for the corresponding load.

- The cost of assigning a real vehicle to a dummy load is the unattractiveness cost of a location (vehicle waits at its 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 $C_{urg}/(\textit{load release time} + \textit{time window size} - \textit{current time})^{\beta}$ if $(\textit{load release time} + \textit{time window size}) > (\textit{current time})$ and equals ∞ otherwise.

The values of the cost coefficients in our implementation are $C_{empt} = 10$, $C_{wait} = 2$, $C_{loc} = 5\times10^3$, $C_{urg} = 2\times10^7$, $\alpha = 2$, $\beta = 1$ or 2 (for I- and U-layout respectively). $\alpha>1$ and $\beta\geq1$ 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. The general operating framework for the scheduling approach using the *DAS* algorithm is illustrated in Figure 3.

<Insert Figure 3 here>

Look-ahead dynamic assignment algorithm (LAS). The assignment algorithm works best for the case where we may assign about one load to each vehicle; however, with the implementation of Figure 3, we do not have enough loads to assign to all vehicles. In VBITSs, we may know some information about future load arrivals, which can be used 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 approximately $|K|$ (the number of vehicles) loads are

known to arrive or are waiting ($T_L = |K| \times \tau$, τ is the load inter-arrival time). We may consider *LAS* as 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} starting from the current time.

6.3 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 myopic 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 Experimental conditions

The experimental conditions related to system input parameters and computational environment are similar to those in subsection 5.1. There are three 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. Next, we have two values for the load inter-arrival time ($\tau = 3, 3.6$). These values lead to vehicle utilizations of about 80% or more. Finally, we use one-sided time-windows only (the lower bound being the load generation time and the maximum waiting time is unrestricted). However, since the cost function f_{main} in rules like *DAS* and *LAS* is in favor of loads with smaller waiting times, this may lead to a high value of the maximum waiting time or event some load might be ignored. We therefore use an artificial time window approximately equal to the maximum load waiting time when the NVF rule is used (determined in a pre-run) to guarantee an acceptable value of the maximum load waiting time.

Performance criteria. The main performance criterion is the average load waiting time (*Avg_wait*), in line with Model SP. To obtain more information about the approaches, the maximum load waiting time (*Max_wait*), vehicle utilization (*Util%*), and the maximum number of loads in queues (*Max_inQ*) are added as side criteria. We use Tukey's test (Hsu, 1996) with a 95% confidence level (CL), using SPSS 11.0 to rank the performance of the approaches, based on average load waiting .times.

Experimental approach parameters. In total seven methods are compared: two dispatching rules (*NVF* and *NVF_LA)* and five scheduling algorithms. The length of the look-ahead period (T_L) of *NVF_LA* is set during the experimentation by selecting the one with the best performance. The five scheduling approaches are *DAS, LAS* ($T_L = |K| \times \tau$), insertion (*Insertion*), combined (*Com-Heur*) and column-generation (*Column-Heur*) under two rolling horizon policies: by time (T) and by the number of loads (M). For rolling by the number of loads, $M = |K| \times 4$, $m = |K| \times 2$ ($M = 24$, $m = 12$) initially. Scheduling 4 loads per vehicle on average leads to good scheduling results for the column generation within a short computational time (less than 10 seconds). For rolling by

time we choose $H = |K| \times 4 \times \tau$, $h = |K| \times 2 \times \tau$ (H = 72 and 86.4, h = 36 and 43.2 corresponding to τ = 3 and 3.6, respectively). The lengths of the planning horizons (simulation periods) are 900 (τ = 3) and 1080 (τ = 3.6) time units (seconds).

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

7.2 Performance evaluation

The results for U and I-layout are given in Tables 3 and 5 respectively for different parameter combinations. The ranking results can be found in Tables 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 Tables 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>

When we schedule vehicles using two dynamic scheduling strategies, *Com_Heur* and *Column_Heur,* the average load waiting time reduces dramatically compared to dispatching in Tables 3&4. The 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 86.2% (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 heuristic performs much better than *Insertion* with the largest improvement 42.2%. Both *NVF_LA* and *LAS* perform significantly better than *NVF* and *DAS*, respectively since they use more information about future load arrivals.

DAS performs slightly better than *NVF* in general. *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 the larger the load inter-arrival time τ) are, the bigger the improvements of the two scheduling approaches (*Com_Heur*, and *Comlum_Heur*), 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 better for the uniform load inter-arrival distribution relative to the exponential distribution. This can be explained by the fact that with the same mean inter-arrival 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 when computational 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 for different approaches in two layouts, Tables 7 and 8 give some selected results corresponding to selected load look-ahead times, scheduling approaches, and system layouts. Since the dynamic scheduling heuristics behave similarly, only the combined heuristic is selected for experimentation. 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 make several important observations.

Value of look-ahead information.

- 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. Apparently, the best value for the look-ahead period is fairly small (less than 3τ); it can be determined by experimentation.
- 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 logical to assign only one load for each vehicle. Looking 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. Similarly to *LAS*, looking 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: (1) *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. (2) *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.

As a conclusion to this section, the influences of main factors are summarized in

Table 9.

8. Concluding remarks

This paper studies real-time vehicle scheduling approaches in internal transport systems. These systems can be characterized by a high degree of uncertainty, short travel times, stopping or parking positions spread around the building, and, in many cases, high vehicle utilization rates. In practice, myopic dispatching (for example with *NVF*) is the common method. This paper is one of the first to systematically investigate under which circumstances real-time dynamic scheduling helps in improving performance in vehicle-based internal transport systems

We propose a mathematical model for the VBIT problem and introduce three heuristics for the static vehicle scheduling problem. We apply these static 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 the best-performing dispatching rules *NVF* and *NVF_LA* known from De Koster et al. (2004). Using Monte-Carlo simulation, we systematically compare and rank the performances (primarily measured by the average waiting time) of these seven approaches (two dispatching and five scheduling approaches), by varying load inter-arrival distributions, load arrival variances, and layouts (U and I-layout).

Our results show that (*a*) the dynamic scheduling approaches of *Com-Heur, Column-Heur*, and *LAS* perform significantly better than the dispatching rules. Depending on layouts and working conditions, the waiting time reduction can be as much as 85%. (*b*) When sufficient load pre-arrival information is available (see Tables 7 and 8), the scheduling approaches perform significantly better than dispatching with *NVF_LA*. (*c*) For a given level of pre-arrival information, *Com-Heur* and *Column-Heur* perform (slightly) better than *LAS* for the U-layout warehouse. However, *LAS* has the advantage of easy implementation and shorter computational time. (*d*) The performance of the approaches is highly impacted by load inter-arrival distributions, load arrival variances, and layouts (U and I-layout).

In sum, we recommend *Com-Heur, Column-Heur*, and *LAS* for dynamic VBIT. *Column-Heur* has the shortest average time of loads, but only suits small or medium size problems due to its computational complexity. If no or only little ($< 0.5\tau$) prior information is available, LAS is the recommended approach.

The results obtained in this paper suggest some future research topics: (1) developing better (particularly faster) static and dynamic-heuristics; (2) taking the vehicle congestion problem into account. It is not easy to model this, but it might imply all rules have to be adapted to consider

potential congestion. Moreover, severe congestions (De Koster and Yu, 2008) may not be solved by mere scheduling, and even lead to a redesign of facilities; (3) adapting the heuristics to scheduling load transports in other warehousing systems, such as compact automated storage and retrieval systems (as discussed in, for example, De Koster et al., 2008, and Yu and De Koster, 2009a; 2009b).

Acknowledgement

The research is supported by VENI grant (#016.075.154) of NWO (the Netherlands Organization for Scientific Research).

References

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

Asef-Vaziri, A., Hall, N.G. and George, R., 2008. The significance of deterministic empty vehicle trips in the design of a unidirectional loop flow path. *Computers & Operations Research*, 35(5), 1546-1561.

Bent, R. and Van Hentenryck, P., 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33(4), 875-893.

Bilge, U. and Ulusoy, G., 1995. A time window approach to simultaneous scheduling of machines and material handling system in an FMS. *Operations Research*, 43(6), 1058-1070.

Bronmo, G., Christiansen, M. and Nygreen, B., 2007. Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society*, 58(9), 1167-1177.

Chen, Z.L. and Xu, H., 2006. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1), 74-88.

De Koster, M.B.M., 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.

De Koster, R. and Yu, M., 2008. Minimizing makespan and throughput times at Aalsmeer flower auction. *Journal of the Operational Research Society*, 59(9), 1182-1190.

De Koster, M.B.M., Le-Duc, T. and Yu, Y., 2008. Optimal storage rack design for a 3-dimensional compact AS/RS. *International Journal of Production Research*, 46(6), 1495-1514.

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(3), 479-494.

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), 420-433.

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

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

Jang, J., Suh, J. and Ferreira, P.M., 2001. An AGV routing policy reflecting the current and future state of semiconductor and LCD production lines. *International Journal of Production Research*, 39(17), 3901-3921.

Kim, B.I., Oh, S., Shin, J., Jung, M.Y., Chae, J. and Lee, S., 2007. Effectiveness of vehicle reassignment in a large-scale overhead hoist transport system. *International Journal of Production Research*, 45(4), 789-802.

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. Local search in physical distribution management, 1992 (Erasmus University Rotterdam: Rotterdam).

Krishnamurthy, N.N., Batta, R. and Karwan, M.H., 1993. Developing Conflict-Free Routes for Automated Guided Vehicles. *Operations Research*, 41(6), 1077-1090.

Laporte, G., Gendreau, M., Potvin, J.-Y. and Semet, F., 2000. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operations Research*, 7, 285-300.

Le-Anh, T. and De Koster, M.B.M., 2006. A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, 171(1), 1-23.

Mes, M., van der Heijden, M. and van Harten, A., 2007. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1), 59-75.

Morihiro, Y., T., M. and S., K., 2007. An initial assignment method for tasks assignment and routing problem of autonomous distributed AGVs. *Ieice Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E90A (11), 2465-2471.

Ohlmann, J.W. and Thomas, B.W., 2007. A compressed-annealing heuristic for the traveling salesman problem with time windows. *Informs Journal on Computing*, 19(1), 80-90.

Pisinger, D. and Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8), 2403-2435.

Powell, W.B., 1996. A stochastic formulation of the dnamic assignment problem, with an application to truckload motor carriers. *Transportation Science*, 30(3), 195-219.

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

Ropke, S., Cordeau, J.F. and Laporte, G., 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4), 258-272.

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

Savelsbergh, M.W.P., 1995. The General Pickup and Delivery Problem. *Transportation Science*, 29(1), 17-29.

Singh, S.P. and Tiwari, M.K., 2002. Intelligent agent framework to determine the optimal conflict-free path for an automated guided vehicles system. *International Journal of Production Research*, 40(16), 4195-4223.

Thomas, B.W., Anticipatory Route Selection Problems. University of Michigan, 2002.

Tompkins, J.A., White, J.A., Bozer, Y.A. and Tanchoco, J.M.A., 2003. *Facilities Planning*, pp. (New York: John Wiley and Sons).

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

Vis, I.F.A., De Koster, M.B.M. and Savelsbergh, M.W.P., 2005. Minimum vehicle fleet size under time-window constraints at a container terminal. *Transportation Science*, 39(2), 249-260.

Xiang, Z.H., Chu, C.B. and Chen, H.X., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, 174(2), 1117-1139.

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

Yu, Y. and De Koster, M.B.M., 2009a. Designing an optimal turnover-based storage rack for a 3D compact automated storage and retrieval system. *International Journal of Production Research*, 47(6), 1551-1571.

Yu, Y. and De Koster, M.B.M., 2009b. Optimal zone boundaries for two class-based compact 3D automated storage and retrieval systems. *IIE Transactions*, 41(3), 194 - 208.

List of Figures and Tables

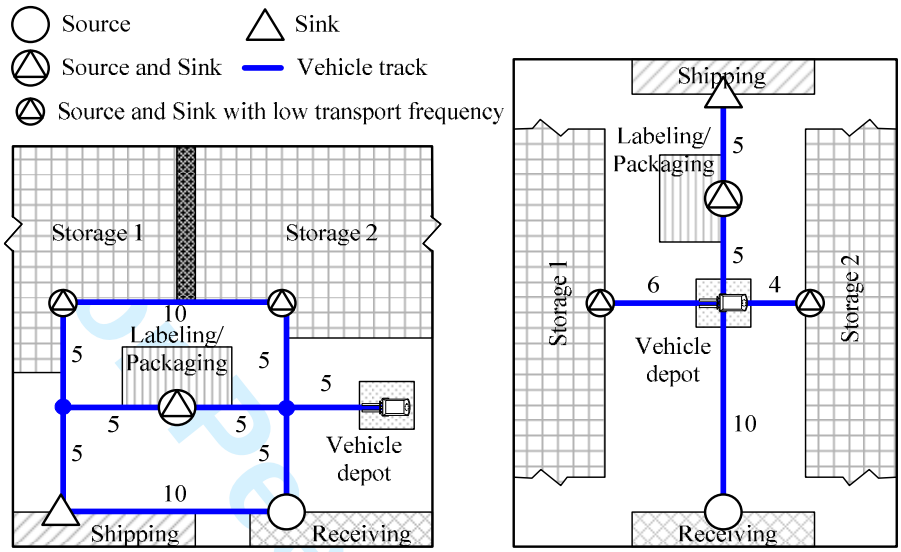
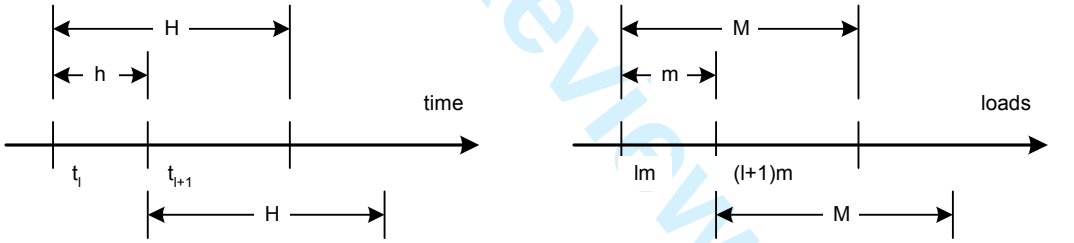
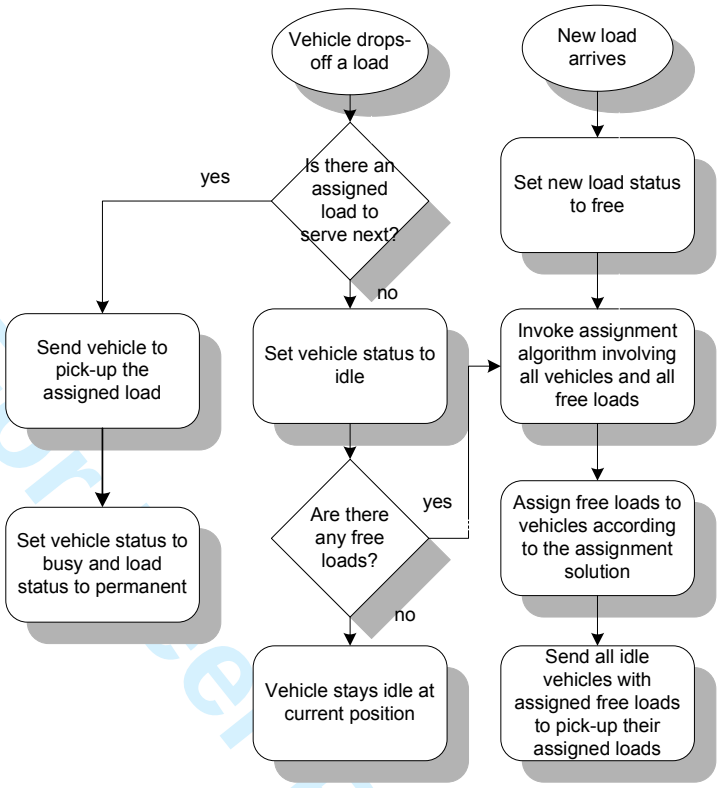


Figure 1. U-layout (left) and I-layout (right) used in experiments



(a) Rolling by time (b) Rolling by the number of loads

Figure 2. Rolling horizon policies



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 3. 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

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 originating 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 for the real time cases

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***	**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 approaches for the U-layout (Tukey test with a 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
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 for the real-time cases

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 approaches for the I-layout (Tukey test with a 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 of three selected approaches with different lengths of look-ahead periods (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 of the three approaches with different lengths of look-ahead periods (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 the performances of different approaches

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.