Les Cahiers du GERAD ISSN: 0711–2440

A Heuristic Approach for the Shared Storage Based on the Duration-of-Stay of Unit Loads

L. Chen, A. Langevin,
D. Riopel, P. Montulet

G–2008–37

May 2008

A Heuristic Approach for the Shared Storage Based on the Duration-of-Stay of Unit Loads

Lu Chen

School of Mechanical Engineering
Shanghai Jiao Tong University
Shanghai, China, 200240
and Département de mathématiques et génie industriel
École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7
lu.chen@polymtl.ca

André Langevin
Diane Riopel

GERAD and Département de mathématiques et génie industriel
École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7
{andre.langevin, diane.riopel}@polymtl.ca

Pierre Montulet

Département de mathématiques et génie industriel
École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7
pierre.montulet@polymtl.ca

May 2008

Les Cahiers du GERAD
G–2008–37

Abstract

Shared storage policy allows more flexible use of space than that allowed by the dedicated storage policy. This paper addresses duration-of-stay based shared storage in an automated storage/retrieval system. An integer programming model is formulated to obtain optimal solution for small and medium sized problems. A graph based heuristic approach is developed to solve large scale problems in reasonable time. The computational results indicate that the approach is very effective in finding high quality solutions, in terms of both total travel time and space requirements.

Key Words: Shared storage; Duration-of-stay; Heuristics.

Résumé

L'entreposage partagé permet une utilisation plus flexible de l'espace que l'entreposage dédié. Cet article traite de l'entreposage partagé basé sur la durée de séjour dans un système d'entreposage automatisé. Un modèle de programmation linéaire en nombres entiers permet d'obtenir une solution optimale pour les problèmes de petite et moyenne taille. Une méthode heuristique est développée pour résoudre les problèmes de grande taille. Les tests indiquent que l'heuristique est très efficace pour trouver des solutions de haute qualité par rapport au temps de manutention et à l'espace requis.

1 Introduction

Warehouses are an essential component of any supply chain. Products are stored temporarily in warehouses and retrieving products from storage can fill customer orders. The research on warehousing systems gained interest in 1970s (Shouman et al., 2005). A number of warehouse operation decision support models have been proposed in the literature. They are extensively discussed in de Koster et al. (2007), Gu et al. (2007) and Cormier (2005).

The storage location assignment problem is to assign incoming products to storage locations in order to reduce material handling cost and improve space utilization. Most of the research has focused on unit-load warehouse. With regard to unit load storage, there are four main storage policies (Francis et al., 1992). They are: dedicated storage policy, randomized storage policy, class-based storage policy and shared storage policy. *Dedicated storage* involves the assignment of specific storage locations for each product stored. Heskett (1963, 1964) considers the location assignment in a dedicated storage warehouse. He defines the cube-per-order index (COI) of an item to be the ration of the item's total required space to the number of trips required to satisfy its demand per period. The reciprocal of the COI is called the turnover rate of that item, therefore the COI policy is also referred to as the turnover-based storage policy. The results on COI-based policy and its optimality in different systems are discussed widely. Malmborg and Krishnakumar (1989) investigate the optimal storage assignment policy for a multi-address warehousing system employing a dedicated storage policy. The authors demonstrate that the COI-based assignment policy is optimal with respect to the order picking cost for fixed inventory levels and a fixed assignment of items to order pickers. Malmborg and Altassan (1998) extend the analysis for item assignment policies in unit load warehousing systems to less than unit load warehousing systems. This study provides an analytical tool for assessing the impact of alternative storage policies in less than unit load storage systems, such as COI-based dedicated storage and random storage with "closest open location" rule. Montulet et al. (1998) study the problem of minimizing the peak load with single command and dedicated storage policy. The peak load is the maximum value of the daily loads over a fixed planning horizon. The daily load is the expected total time of the command cycles. A mathematical model is formulated and a branch-and-bound algorithm is developed to solve the problem.

Randomized storage allows the incoming item to be stored in any of the open locations. The closest open location (COL) rule is often used in practice. The random assignment method requires less space at the expense of increased travel distance (Francis et al., 1992). Hausman et al. (1976) argue that the COL storage and random storage have a similar performance if items are moved by full pallets only. Malmborg (1996) provides a model to analyze tradeoffs in space requirements and retrieval efficiency associated with dedicated and randomized storage policies.

As a compromise between dedicated storage and randomized storage, *class-based storage* is frequently used, in which the items are partitioned into a small number of classes based on their turnover rates. Most research on class-based storage has been performed in the context of an automated storage and retrieval system (AS/RS). Normally, the class of items having the highest turnover rate is assigned to the "better" locations. Graves et al. (1977) observe that in order to store an incoming item in the correct class region, empty slots must be available. This increases space requirements with the number of classes. Accordingly, class-based storage requires more space than randomized storage. Hausman et al. (1976) use continuous representation of the traveling distance and the turnover to quantify the reduction in machine travel times of the class-based assignment over random assignment. Significant potential reductions in machine travel times in automatic warehousing seem to be possible

based on class-based storage policy. Lai et al. (2002) deal with the problem of paper reel assignment where different classes of paper reels need to be placed in the cells of a warehouse. Here paper reels are divided into classes based on their diameters instead of turnover rate. A mathematical model is formulated to assign the paper reels into the cell space so that the total transportation cost is minimized. A two-stage simulated annealing approach is used to solve the problem. Zhang et al. (2000) extend the mathematical model to the paper reel assignment problem with adjacency constraints imposed on certain items.

Shared storage policy based on duration-of-stay (DOS) is another storage policy that is from industry practice but has received relatively little attention from the research community. DOS-based shared storage recognizes and takes advantage of the inherent differences in lengths of time that individual items remain in storage. It can provide substantial benefits when precise information is available concerning the timing of storages and retrievals for individual items of products. However, as stated by Francis et al. (1992), it would be incorrect to assume that we cannot benefit from using DOS-based shared storage in the absence of such precise information.

This paper deals with the storage location assignment problem (SLAP) in an AS/RS with DOS-based shared storage policy. The single command mode is assumed where there is only one storage or one retrieval on each round trip of the storage/retrieval (S/R) machine. The objective of the problem is to improve the storage efficiency based on the travel time of the S/R machine.

This paper is organized as follows. In Section 2, we introduce the problem and present the related work. In Section 3, an integer programming formulation for the SLAP is proposed. A heuristic procedure is developed in Section 4. Computational experiments are conducted and the results are reported in Section 5. Finally, in Section 6 we present our conclusions and indicate some possible extensions.

2 DOS-based shared storage

In DOS-based shared storage policy, the expected DOS of the x^{th} item of a product with replenishment lot size Q is x/λ for $x = 1, 2, \ldots, Q$, where λ is the demand rate of that product. Then the items of all the different products are assigned to the locations according to their DOSs. DOS-based shared storage policy allows more flexible use of space than that allowed by the dedicated storage policy. This provides the potential to better utilize the more desirable storage locations, and increases the efficiency of the storage system.

Storage policy based on the DOS of each item is more complex and few research results are available, although it is often found in practice. Goetschalckx and Ratliff (1990) appear to be the first to study formally the shared storage policy based on the DOS of each item. Two heuristics are developed for static and dynamic unit load warehousing context. Simulation results are provided which compare travel times for dedicated storage, random storage, turnover-based storage and DOS-based storage. The authors show that, for single command storage and retrieval, DOS-based shared storage policy has the potential to significantly decrease travel time.

Montulet et al. (1997) study the DOS-based storage policy in a single command, non-interleaving, and static warehousing system. A column generation method is applied to solve medium size problems optimally. A heuristic approach is proposed for large scale problems. Kulturel et al. (1999) compare the turnover-based and DOS-based storage assignment policies

in an AS/RS, which operates based on the continuous review inventory policy. The average travel time of the handling machine is used as the main performance measure.

Goetschalckx and Ratliff (1990) propose a heuristics, called GReedy Heuristics (GRH), based on the following Lemma:

Lemma 1 *For all shared storage systems that satisfy the travel independence condition, any storage policy that relies only on the ranking of the travel times to the locations is optimal if and only if it simultaneously maximizes the number of unit loads stored in the first, first two, . . . , first K locations (locations are ordered according to there accessibilities).*

The condition in the Lemma is very restrictive. Trying to maximize the number of items stored in the most accessible locations, GRH assigns the items with early departure times to the closest open locations in each step. Ties are broken by ordering the items by non-decreasing arrival times.

3 Problem Formulation

We address the storage location assignment problem in an AS/RS that operates under DOS-based shared storage policy. The AS/RS under consideration in this paper is a storage system with multiple aisles. Each aisle is served by a dedicated S/R machine that can reach a pick face on each side of the aisle. All items in the system are stored and moved in unit loads. The S/R machine can carry one unit load at a time and can either store an incoming load in an empty location in the rack or pick up a load from a location in the rack and deliver it to the I/O station of the system. Each aisle has one I/O station for incoming and outgoing loads. An effective storage location assignment can reduce the mean travel times of the S/R machine.

The single command mode is assumed where there is only one deposit or one retrieval on each round trip. The time to deposit or pickup a unit load is assumed to be independent of the sequence of the requests. We minimize the total travel time for performing all the storage and retrieval operations. The total travel time consists of the following two components: the deposit time for all the storage unit loads; and the retrieval time for all the retrieval unit loads.

3.1 Notations

The following notations are defined for the mathematical formulation:

N**:** number of unit loads
K**:** number of locations
i, j**:** index of unit load, i, $j = 1, 2, \ldots, N$
k**:** index of location, $k = 1, 2, \ldots, K$

Each unit load i, $i = 1, 2, \ldots, N$, has a duration-of-stay DOS_i. It is an interval $[a_i, d_i]$, where a_i is the arrival time of unit load i, and d_i is its departure time. A location may be empty before a_i, but cannot be assigned to i until i arrives at a_i. i is a storage unit load at time a_i, and becomes a retrieval unit load at time d_i. There is a one-way traveling time c_k to each location k, $k = 1, 2, \ldots, K$. This time is independent of which unit load is stored in that location. Also, c_k is assumed to be a small value, and not comparable to the DOSs of the unit loads.

An $N \times N$ matrix $W, W = \{w_{ij} | i, j = 1, 2, \ldots, N\}$, is defined to indicate whether any two unit loads could be assigned to the same location. w_{ij} equals to 1 if the DOSs of i and j are not overlapped between each other. Thus, i and j can be assigned to the same location in the planning horizon. w_{ij} is defined as:

$$w_{ij} = \begin{cases} 1, a_i \geq d_j \text{ or } a_j \geq d_i, i, j = 1, \ldots, N, i \neq j \\ 0, \text{ otherwise} \end{cases}$$

The objective is to minimize the total travel time required to process the storage and retrieval operations of the N unit loads in the current time horizon.

Decision variables:

$x_{ik} = 1$, if i is stored in location k, 0 otherwise;
$y_{ijk} = 1$, if i and j are stored in the same location k, 0, otherwise.

3.2 Mathematical model

In this section, we formulate the SLAP as an integer programming problem.

$$\text{Min} \sum_{i=1}^{N} \sum_{k=1}^{K} 4 c_k x_{ik} \tag{1}$$

subject to:

$$\sum_{k=1}^{K} x_{ik} = 1, \quad i = 1, \ldots, N \tag{2}$$

$$y_{ijk} + 0.5 \geq 0.5(x_{ik} + x_{jk}), \quad i, j = 1, \ldots, N, \quad k = 1, \ldots, K \tag{3}$$

$$(1 - w_{ij}) y_{ijk} = 0, \quad i, j = 1, \ldots, N, \quad k = 1, \ldots, K \tag{4}$$

$$x_{ik}, y_{ijk} \in \{0, 1\} \tag{5}$$

For each unit load there are four one-way trips, therefore the objective function (1) is to minimize the total travel time. Constraints (2) ensure that each unit load is assigned to only one location. Constraints (3) ensure that $y_{ijk} = 1$ when i and j are assigned to the same location k. When $w_{ij} = 1$, constraints (4) are not binding. When $w_{ij} = 0$, constraints (4) ensure that i and j are not assigned to the same location. Constraints (5) specify the binary property of the decision variables.

Solving the problem for the optimum will require a prohibitive computational effort. For example, the solution process terminates because too much memory is used for a problem with 120 unit loads and 40 locations using CPLEX 10.1.1. This encourages the use of heuristic procedures that can find a good solution reliably in a reasonable amount of time.

4 A heuristic approach

GRH is myopic in that it assigns the items in the current step without considering their influence on the future decisions. It is because that it maximizes the number of items stored

in the most accessible locations separately in each step. Some items which are not assigned too early could improve the results of the subsequent steps. It is on this observation that our Graph Based Heuristic approach (GBH) is inspired.

In order to describe the procedure, a storage graph is introduced that represents all the storage activities in a planning horizon. A storage activity is defined here as a temporary stay of an incoming unit load i that arrives at time a_i and departs at time d_i. For a given time horizon T, the storage graph $\boldsymbol{G = (V, C, A)}$ is defined as follows:

- $\boldsymbol{V}$ is the set of nodes corresponding to all the time periods in T. $V = \{1, 2, \ldots, T+1\}$. The node of time period 1 is denoted as the source node, and the node of $T+1$ is denoted as the sink node.
- $\boldsymbol{C}$ is the set of directed connection arcs. There is a connection arc between each pair of consecutive nodes.
- $\boldsymbol{A}$ is the set of directed storage arcs. Each storage arc represents a storage activity. The node of arrival time period and the node of departure time period of one storage activity are connected by a storage arc. A weight defined later is associated with each storage arc.

This storage graph $\boldsymbol{G}$ contains the DOS information of all the unit loads to be stored. Moreover, a path from the source node to the sink node is a combination of unit loads whose DOSs are not overlapped and could be assigned to the same location.

Example 1. Consider an AS/RS system consisting of four unit loads to be stored in a four-day horizon and three storage locations. Here $N = 4$, $K = 3$. Table 1 shows the arrival and departure time of the four unit loads. The traveling time for the S/R machine from the I/O point to each location is 1, 2, and 3 respectively.

Table 1: DOS of storage unit loads

i	arrival time	departure time	DOS
1	1	2	1
2	1	3	2
3	3	4	1
4	2	5	3

Figure 1 illustrates a storage graph of the system for the instance. Each storage arc has a label indicating the associated storage activity.

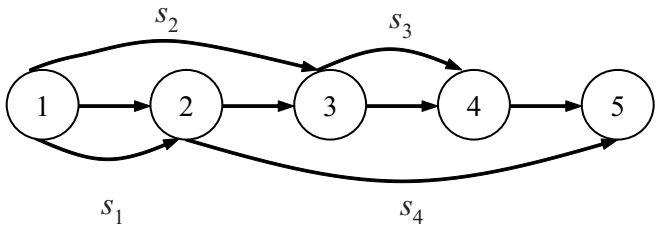


Figure 1: A storage graph with 4 unit loads

Each storage arc s_i is associated with a weight of $B + DOS_i$, where B is a constant bigger than T, and DOS_i is the duration of stay of the corresponding unit load i. Thus, the GBH

tries to satisfy at best the condition of optimality in the Lemma by finding, in the assignment of each step, the longest path in the graph. Obviously, with this definition of weight, the path that contains the largest number of storage arcs will be the longest.

Example 2. The longest path of the graph in Figure 1 is shown in Figure 2. The length of the path equals to $2B + 4$.

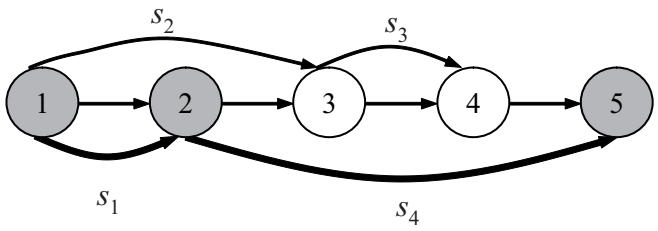


Figure 2: The longest path of the storage graph (in shadow)

The GBH consists of searching the longest path in the storage graph, and follows the steps as:

Step 1: Find the longest path p in $\boldsymbol{G}$ from the source node to the sink node;

Step 2: Assign the unit loads corresponding to the storage arcs in p to the most accessible open location;

Step 3: Delete the storage arcs in p from $\boldsymbol{G}$;

Step 4: If $\boldsymbol{A}$ is empty, then stop; otherwise, go to step 1.

Let us see how the GRH and the GBH do the assignment by using the small instance of Example 1. By GRH, the unit loads are ordered by non-decreasing departure times. It begins with the assignment of unit load 1 to location 1, which is the most accessible open location. When unit load 2 enters, it is assigned to location 2 because location 1 is occupied with unit load 1. The same rules apply to unit load 3 and 4. The whole solution is illustrated in Figure 3(a). The total travel time is $4\times(1\times2+2\times1+3\times1) = 28$. Applying the GBH for the system in the instance, we get an assignment solution that is illustrated in Figure 3(b). That is, unit load 1 and 4 are assigned to location 1; unit load 2 and 3 are assigned to location 2. The total traveling time in this case is $4 \times (1 \times 2 + 2 \times 2) = 24$. And only two locations are needed.

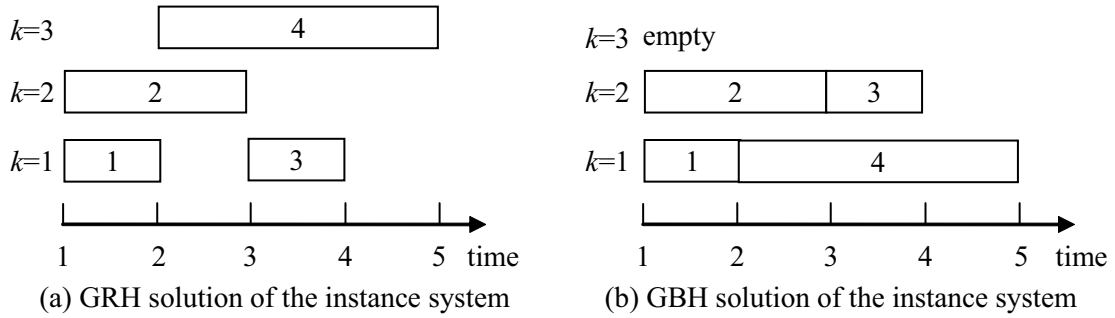


Figure 3: Two solutions of the instance system

5 Experiments

The experiments were conducted to determine how well the proposed approach GBH performs as compared with the GRH. A single aisle consisting of two racks is considered in our computational experiments. Let L be the length and H be the height of the AS/RS rack, and V_h and V_v the horizontal and vertical travel speed of the S/R machine. Then, the times required to travel the full length and height, respectively, are given by $t_h = L/V_h$ and $t_v = H/V_h$. Let $Z = \max\{t_h, t_v\}$, $b = \min\{t_h/Z, t_v/Z\}$, which are referred to as the maximum travel time and shape factor, respectively. We consider a "square-in-time" system, where $b = 1$.

Each location is associated with a pair of travel times in horizontal and vertical directions (h_k, v_k) that were randomly generated from uniform distributions U(0,1). It is assumed that the S/R machine travels simultaneously in the horizontal and vertical directions, i.e., Tchebychev travel. Thus, the traveling time c_k of storing/retrieval a unit load in/from location $k, k = 1, \ldots, K$, is

$$c_k = \max(h_k, v_k) \tag{6}$$

The experiments are conducted in the same way as they are in Goetschalckx and Ratliff (1990). The information on the DOS of each unit load is generated randomly. It is controlled by parameters that allow us to produce instance sets with specific characteristics. These parameters are: the number of products P, the reorder quantity Q, and the average demand inter-arrival time DIT. Various test problems are generated by varying P from 5 to 80 and Q from 1 to 40.

All the products in a single experiment have the same Q. For each product, replenishment occurs immediately after the last unit of inventory is retrieved. The DIT_p of product p, $p = 1, 2, \ldots, P$, is sampled from a uniform distribution of (1,4) (numbers are rounded). The first replenishment date for product p, FRD_p, is a random variable, and is generated from a uniform distribution of $(1, Q^*DIT_p)$. The length of planning horizon is set to $T = \max\{Q^*DIT_p | p = 1, \ldots, P\}$. Therefore, during a planning horizon the times of replenishment for product p equals to $\left\lfloor \frac{T - FRD_p}{DIT_p * Q} \right\rfloor$.

We used Visual C++ programming language to code the algorithms, and perform the computational test on a personal computer with a Pentium 930 MHz processor and 384 MB RAM. CPLEX 10.1.1 integer programming software is used to find the optimal solutions for small sized problems.

To assess the quality of the heuristic methods, we compare the solution values from the heuristics with the optimal solution for small and medium sized problems. We compare the average travel time performance of both heuristics over various values of problem size (P and Q). The average travel time is the total travel time divided by the total number of storage/retrieval operations. For each scenario, we randomly generated 10 instances. Table 2 gives the number of times both methods find the optimum out of 10 instances (see column '#OPT'), the average relative gap from the optimum, and the CPU seconds for both methods. The table also gives the average CPU time of the CPLEX solver.

Experimental results show that the GBH performs very effectively and is superior over the GRH for all the testing problems. For 149 testing problems out of 150 in total, the GBH found the optimum. The GRH found the optimum for 91 testing problems out of 150. The average gap of the GRH ranges from 0% to 1.21%, while the average gap of the GBH ranges

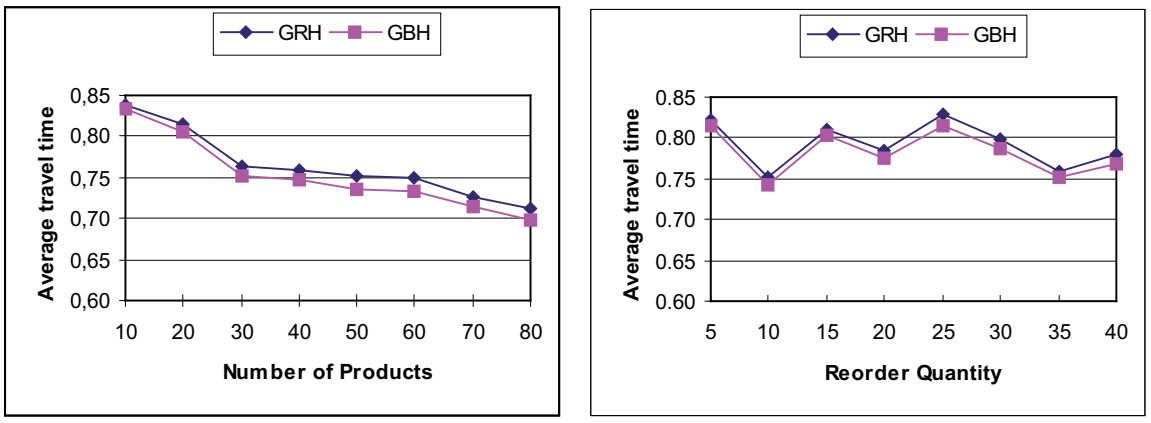
Table 2: Performance comparison for small and medium sized problems

		GRH			GBH			solver
P	*Q*	#OPT	GAP	CPU	#OPT	GAP	CPU	CPU
5	1	10	0%	0.152	10	0%	0.154	0.046
5	2	9	0.22%	0.173	10	0%	0.174	0.255
5	3	8	0.57%	0.216	10	0%	0.211	1.104
5	4	6	0.45%	0.265	10	0%	0.262	2.772
5	5	5	0.77%	0.289	10	0%	0.295	2.090
8	1	10	0%	0.163	10	0%	0.162	0.085
8	2	8	0.23%	0.222	10	0%	0.223	0.088
8	3	7	0.34%	0.285	10	0%	0.286	4.478
8	4	4	0.50%	0.443	10	0%	0.435	1700
8	5	3	0.78%	0.596	10	0%	0.574	2040
10	1	10	0%	0.171	10	0%	0.184	0.144
10	2	6	0.31%	0.242	10	0%	0.241	1.324
10	3	1	0.77%	0.378	9	0.02%	0.365	21.53
10	4	3	0.40%	0.631	10	0%	0.602	1322
10	5	1	1.21%	0.854	10	0%	0.836	2777

from 0% to 0.02%. The results also suggest that as the problem size increases the quality of the GBH solutions is still very good, while the quality of the GRH solutions deteriorates.

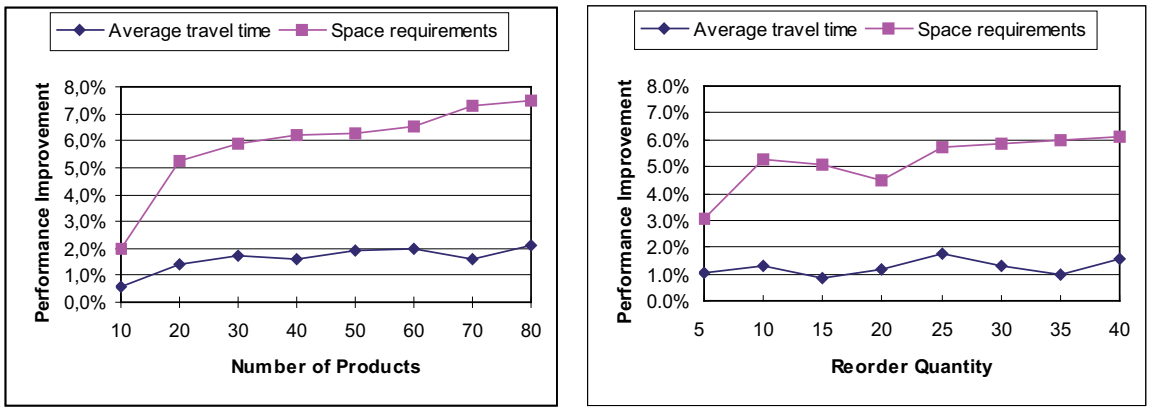
The performances of the GBH and GRH are compared as a function of different parameters for large scale problems, whose sizes range from several items to several thousands (more than three thousands).

Both heuristics have comparable computing times, ranging from a split second to less than fifteen minutes for the problems of different sizes. Figure 4 compares the performance on the average travel time of both heuristics. The GBH results in shorter travel times than the GRH for all the instances.



(a) Average travel time versus number of products (b) Average travel time versus reorder quantity

Figure 4: Performance comparison on the average travel time



(a) Average improvements versus number of products (b) Average improvements versus reorder quantity

Figure 5: Average performance improvements of the GBH over the GRH

Figure 5 illustrates the average performance improvements of the GBH over the GRH. Figure 5(a) shows the performance improvements as a function of number of products. The travel time savings of the GBH is from 0.56% to 2.13%, with an average of 1.61%. The savings of the space requirements is from 2.01% to 7.50%, with an average of 5.87%. The improvements by the GBH increase with increasing number of products. Figure 5(b) shows the performance improvements as a function of reorder quantity. The travel time savings of the GBH are from 0.95 to 1.75%, with an average of 1.25%. The savings of the space requirements are from 3.08% to 6.09%, with an average of 5.20%. Also, the improvements by the GBH increase with increasing reorder quantity. It is observed from the results that the performances of the GBH show much improvement over the GRH as the problem size increases.

6 Conclusions

The study of Goetschalckx and Ratliff (1990) shows that the DOS-based shared storage has an average travel time saving of 25% when compared with the turnover-based dedicated storage. The storage location assignment problem addressed in this paper aims at minimizing the total travel time by an S/R machine in an AS/RS with DOS-based shared storage policy. An integer programming model is provided to solve the problem optimally. A graph based heuristic GBH is developed to solve large scale problems in reasonable time.

It is found that the GBH has a better performance than the GRH by Goetschalckx and Ratliff (1990), in terms of both total travel time and space requirements. The GBH gives optimal solutions for all the small and medium sized test problems with only one exception. Compared with the GRH for large sized problems, the GBH has an improvement of from 0.56% to 2.13% in terms of total travel time, and an improvement of from 2.01% to 7.50% in terms of space requirements.

Future work could focus on extending the model to integrate the interleaving rules. Another interesting direction is to consider storage location assignment with limit processing capacity of the system. A question arises in this context, that is, how to distribute the workload in order to avoid the phenomena of queues.

References

Cormier, G., 2005. Operational research methods for efficient warehousing. In: Langevin, A. and Riopel, D. (eds.), Logistics Systems: Design and Optimization, pp. 93–122. Springer, New York.

De Koster, R., Le-Duc, T., and Roodbergen, K.J., 2007. Design and control of warehouse order picking: A literature review. European Journal of Operational Research, 182: 481–501.

Eynan, A. and Rosenblatt, M.J., 1993. An interleaving policy in automated storage/retrieval systems. International Journal of Production Research, 31(1): 1–18.

Francis, R.L., McGinnis, L.F., and White, J.A., 1992. Facility layout and location: an analytical approach, Prentice-Hall, Englewood Cliffs, N.J.

Geotschalckx, M. and Ratliff, H.D., 1990. Shared storage policies based on the duration stay of unit loads. Management Science, 36 (9): 1120–1132.

Gu, J., Goetschalckx, M., and McGinnis, L.F., 2007. Research on warehouse operation: a comprehensive review. European Jounal of Operational Research, 177: 1–21.

Hausman W.H., Schwarz L.B., and Graves S.C., 1976. Optimal storage assignment in automatic warehousing systems. Management Science, 22(6): 629–638.

Heskett, J.L., 1963. Cube-per-order index – A key to warehouse stock location. Transport and Distribution Management 3, 27–31.

Heskett, J.L., 1964. Putting the cube-per-order index to work in warehouse layout. Transport and Distribution Management 4, 23–30.

Lai, K.K., Xue, J., and Zhang, G.Q., 2002. Layout design for a paper reel warehouse: a two-stage heuristic approach. International Journal of Production Economics, 75: 231–243.

Malmborg, C.J., 1996. Storage assignment policy tradeoffs. International Journal of Production Research, 34(2): 363–378.

Malmborg, C.J. and Altassan, K.M., 1998. Analysis of storage assignment policies in less than unit load warehousing systems. International Journal of Production Research, 36(12): 3459–3475.

Malmborg, C.J. and Krishnakumar, B., 1989. Optimal storage assignment policies for multiaddress warehouse systems. IEEE Transaction on Systems, Man, and Cybernetics, 19(1): 197–204.

Montulet, P., Langevin, A., and Riopel, D., 1997. Le problème de l'optimisation de lentreposage partagé : méthodes exacte et heuristique. INFOR, 35(2): 138–153.

Montulet, P., Langevin, A., and Riopel, D., 1998. Minimizing the peak load: an alternate objective for dedicated storage policies. International Journal of Production Research, 36(5): 1369–1385.

Shouman, M.A., Khater, M., and Boushaala, A.A., 2005. Comprehensive survey and classification scheme of warehousing systems. Proceedings of the 2005 International Conference on Simulation and Modeling. Thailand.

Zhang, G.Q., Xue, J., and Lai, K.K., 2000. A genetic algorithm based heuristic for adjacent paper-reel layout problem. International Journal of Production Research, 38(4): 3343–3356.