Journal of the Korean Institute of Industrial Engineers
Vol. 37, No. 4, pp. 289-296, December 2011.

Pallet Consolidation Problems in Distribution Centers

Suk-Chul Rim[*†] · **Kyu-Seok Lim**

Department of Industrial Engineering, Ajou University, Suwon 443-749, Korea

물류센터의 팔렛합적 문제

임석철 · 임규석

아주대학교 대학원 산업공학과

In most distribution centers, products are received from the suppliers in units of pallet, stored in a rack, moved to the picking area for replenishment, picked according to customer orders, and shipped to customers. In some distribution centers, however, replenishment is made in not a whole pallet but only a portion of a pallet load, mainly due to the limited space in the order-picking area; as a result, partially loaded pallets occupy cells in the rack. As the number of slow-moving items increases, more cells are occupied by partially loaded pallets so that fewer empty cells are available for storing full, incoming pallets. This will necessitate the construction or leasing of additional storage space, which will entail significant cost. As an alternative, we propose pallet consolidation, which involves moving a partial load to another partially-loaded pallet in order to create one empty cell in the rack. In this paper, we define three pallet consolidation problems; formulate each problem as a binary integer programming model; and present heuristic algorithms for the problems. The average performance of each of the proposed heuristics is evaluated using simulation.

Keywords: Pallet Consolidation, Space Utilization, Distribution Center

1. Introduction

Distribution centers (DCs) play a core role in many supply chains by balancing demand and supply, and providing time buffers for replenishment and space buffers between manufacturers and retail stores. The typical functions of DCs include receiving goods from suppliers (usually manufacturers, in units of pallet), storing the incoming pallets in a storage rack, replenishing goods from the storage rack to the order-picking area, and order-picking and delivery to customers. Some DCs operate in cross-docking fashion, whereby incoming pallets are depalletized and distributed directly to the delivery trucks for customers without being stored in the storage facility in DC. However, most DCs currently in use have storage racks to store incoming pallets, which entail significant cost. Moreover, as the number of items grows, DC managers will be forced to expand the capacity of the storage rack, which will necessitate a substantial investment. If the investment can be postponed or even avoided by improving the storage efficiency, the savings will be substantial.

In this study, we consider a typical DC where delivery trucks leave the DC in the morning, and trucks from the suppliers arrive in the afternoon. In between, the distribution center has to prepare to receive incoming pallets, which require an adequate number of empty cells in the rack for storage. If there are no empty cells to store the incoming pallets, these will have to be placed temporarily on the floor and moved later to empty cells, which will increase the amount of material handling and possibly interrupt the flow of materials in the DC. Due to many practical restrictions such as stackability and product shape, the application of the result of this paper can be limited to a certain range of

† Corresponding author : Professor Suk-Chul Rim, Department of Industrial and Information Systems Engineering, Ajou University, Suwon 443-749, Korea, Fax : +82-31-219-1610, E-mail : scrim@ajou.ac.kr
Received August 22, 2011; Revision Received October 13, 2011; Accepted November 10, 2011.

products such as books and boxed products.

New empty cells can be created by combining partially loaded pallets. We will call this *pallet consolidation*. The space utilization of cell i, denoted as u_i, is defined as the proportion of the occupied space to the available space of cell i. The occupied space is defined as the net product volume on the pallet multiplied by $(1+\alpha)$, where α is the surplus coefficient such as 0.2 in order to accommodate many practical restrictions. The surplus coefficient can be determined by each distribution center based on the characteristics of its own products such as shape and stackability.

In this paper, we will consider the following three problems of pallet consolidation.

(1) Consolidating two pallets (C2) : by allowing only two cells to be consolidated, determine which pallet to move to which pallet in order to maximize the number of new empty cells created.
(2) Consolidating at most three pallets (C3) : by allowing at most three cells to be consolidated, determine which pallet(s) to move to which pallet in order to maximize the number of new empty cells created.
(3) Generating Q new empty cells in the shortest time (GQ) : if we need only Q additional empty cells, determine which pallets to move to which pallets so that the Q additional empty cells can be generated in the shortest time.

Note that the common objective function of (C2) and (C3) is to maximize the number of new empty cells, while that of (GQ) is to minimize the time required to generate Q new empty cells through consolidation. Also note that (GQ) may be infeasible for large Q. One may want to consider the case of consolidating more than three pallets, which we think is unrealistic in practice because too many items stored in a pallet will make it difficult to retrieve the required items in the order picking. Hence we consider consolidating only two pallets or at most three pallets as it is likely to be regulated by policy to consolidate only two pallets, or at most three pallets in the distribution centers.

We assume the following.

(1) There are N cells of identical size in a one-sided single rack.
(2) A man-on-board forklift can pick a pallet A from a cell and move to a pallet B in another cell; further, the picker on board manually can move all the items in pallet A onto pallet B in the rack such that later, any item in pallet B can be retrieved manually without being blocked by the items that were moved from pallet A in the same pallet.
(3) After consolidation, the empty pallet is moved to the input/output (I/O) point located at the lower left corner of the rack.
(4) For simplicity and practicality, a pallet is moved to only one pallet to be consolidated.

Suppliers usually ship the products to the DCs in units of pallet for efficient material handling. As the product variety increases, the DCs have to store larger number of stock keeping units (SKUs). A pallet load will take one cell of the rack, no matter how little product is left on the pallet. Therefore, as the slow-moving items grow, DCs will need additional storage space and rack facility, which incurs significant amount of financial investment. If we successfully apply the result of this paper in DCs, then the construction or lease of additional storage facility can be avoided or at least delayed, which will result in a significant amount of cost saving.

The remaining part of this paper is as follows. In Section 2, we review the relevant research results. In Section 3, we will formulate the first two problems as binary integer programming problems; present heuristic algorithms for each problem; and compare the heuristics in terms of their average performance by comparing the respective solutions with the optimal solution from the binary integer programming models. In Section 4, we will formulate the third problem as a binary integer programming problem; present two heuristic algorithms; compare them in terms of their average performance; and present a linear regression model for general use. In Section 5, we will summarize the results of the study and propose directions for further research.

2. Literature Review

There have been numerous research results in the operation of distribution centers, warehousing, storage, and picking. One of the popular research topics regarding pallets is the Pallet Loading Problem (PLP), in which rectangular boxes are placed on a rectangular pallet of specific dimensions so that the number of boxes located completely within the pallet is maximized. For the cases where the boxes are of identical dimensions, the problem is called the manufacturer's pallet loading problem (MPL) by Hodgson (1982). On the other hand, for the cases where the boxes are of different sizes, the problem is called the distributor's pallet loading problem.

For the manufacturer's pallet loading problem, Car-

penter and Dowsland (1985) develop the loading patterns into pallet stacks, and criteria which might be applied to determine the suitability of these stacks for storage and transportation. A technique is developed which allows the stability and clampability of stacks to be tested. The relative importance of different criteria is illustrated and shown to have implications for the structure of algorithms that are to provide acceptable pallet stacks. Dowsland (1987) describes a way of dealing with the problems of failing to solve the manufacturer's pallet loading problem within the given time limit. The discussion is in terms of a recent exact algorithm for the pallet-loading problem, but the methodology can be applied in conjunction with any algorithm for which the proportion of problems exceeding the time limit is small. The result is a piece of software which guarantees to solve any typical pallet-loading problem within a few minutes on an IBM-PC.

Scheithauer and Terno (1996) propose a new heuristic for the pallet loading problem, namely, G4-heuristic, which is based on the definition of the so-called G4-structure of packing patterns. The G4-structure is a generalization of the common used block structure of packing patterns which requires the same orientation of packed boxes within each block. The G4-heuristic yields in approximately 99% of the test instances an optimal solution and solves all instances exactly where at most 50 boxes are contained in an optimal packing.

Morabito and Morales (1998) present a simple and effective heuristic to solve the problem of packing the maximum number of rectangles of sizes (l, w) and (w, l) into a larger rectangle without overlapping. Using moderate computational resources, the procedure was able to find the optimal solution of 99.9% of more than 20,000 examples analysed. Alvarez-Valdes *et al.* (2005) propose a branch-and-cut algorithm for the problem. They exploit the specific structure of the PLP to define the solution graph and develop efficient separation procedures. Martins and Dell (2007) define the Minimum Size Instance (MSI) of an equivalence class of PLP and present an algorithm for MSI. Martins and Dell (2008) present new bounds, heuristics, and an exact algorithm for the MPL The set of all PLP instances with an area ratio (pallet area divided by box area) less than 101 boxes can be represented by 3,080,730 equivalent classes. They propose a heuristic, namely, G5-heuristic, which finds optimal solutions to 3,073,724 of these 3,080,730 classes and in the remaining 7006 classes only differs from the best known bound by one box.

The distributor's pallet (or multi-pallet) loading problem, on the other hand, is to load a set of distinct products with given quantities on pallets (or in containers) and to minimize the number of pallets needed. Bischoff and Ratcliff (1995) generalize the manufacturer's pallet loading problem; and determine the loading scheme for a set of items with different sizes. Terno *et al*. (2000) address the practical aspects in maximizing the space utilization, such as technological constraints, weight distribution over the pallet, stability aspects, etc. They present an efficient heuristic for the multi-pallet loading problem, where the three-dimensional (3D) solution approach uses a layer-wise loading strategy with optimal two-dimensional (2D) loading patterns.

For pallet-loading operations concerning air cargo, Lau *et al*. (2009) present a hybrid approach, using heuristic and genetic algorithms (GA), for solving a profit-based multi-pallet loading problem, which is mathematically formulated as a nonlinear integer programming problem. The simulation results for this multi-pallet loading problem show that GA can find more profitable solutions than simulated annealing, Tabu search, and branch-and-bound (see also Chan *et al*., 2006). Yaman and Sen (2008) deal with the problem of mixing multiple products with low demand into a pallet for a beverage manufacturer so as to minimize the total inventory holding and backlogging costs of its customers over a finite horizon. They formulate the problem as a mixed integer linear program and incorporate valid inequalities to strengthen the formulation. However, to the best of our knowledge, the pallet consolidation problem that we address in this study has not been reported in the literature.

3. Pallet Consolidation Models and Heuristic Algorithms

In this section, we present mathematical formulations and heuristic algorithms for the two problems C2 and C3, respectively, defined in Section 1. For cell (or equivalently, item) i, let u_i denote the space utilization of cell i, that is, the remaining items on the pallet constitute $(100u_i)\%$ of the cell volume capacity; also, let M_{jk} be 1 if item j is moved to item k for consolidation, and 0 otherwise. We will present the optimal solution procedures (termed OC2 and OC3) and heuristic algorithms (termed HC2 and HC3) for C2 and C3, respectively, in the following section.

3.1 Consolidation of Two Pallets (C2)

Problem (C2) can be formulated as a binary integer programming problem as follows.

<OC2>

$$\text{Max } Z = \sum_j \sum_{k \neq j} M_{jk} \tag{1}$$

$$\sum_{k \neq j} M_{jk} \leq 1, \text{ for all } j \quad (2)$$

$$\sum_{j \neq k} M_{jk} \leq 1, \text{ for all } k \quad (3)$$

$$\sum_{j \neq k} M_{jk} u_j \leq 1 - u_k, \text{ for all } k \quad (4)$$

$$\sum_{k} M_{jk} + \sum_{m} M_{mj} \leq 1, \text{ for all } j \quad (5)$$

$$M_{jk} = 0 \text{ or } 1, \text{ for } all\ j \neq k \quad (6)$$

Eq. (1) maximizes the number of new empty cells to be created; Eq. (2) enforces Assumption (4) in Section 1; Eq. (3) stipulates that any cell receive items from at most one other cell; Eq. (4) ensures that the total volume of items in the two consolidated cells does not exceed the space capacity of a cell; Eq. (5) prevents cyclic solutions (such as moving A to B and B to C) for consolidation; and Eq. (6) designates binary integer variables. Solving this formulation will yield the maximum number of new empty cells. However, there are two reasons why the above model hardly can be used in most DCs. First, most DCs do not have an optimizer to solve a binary integer programming problem every day. Second, as the problem grows in size, that is, for larger racks, the computational time may be prohibitive. Therefore, we present a simple heuristic algorithm for (C2) as follows. For N cells in a rack,

<HC2> Heuristic algorithm for C2.
Step 0 : $\Omega = \{1, 2, \cdots, N\}$, $m := 0$.
Step 1 : For i$\in\Omega$, select the smallest u_i.
Step 2 : Find the largest u_j, j$\in\Omega$, such that $u_i+u_j \leq 1$; (move i to j) Go to Step 3.
If no such u_j exists, stop.
Step 3 : $m := m+1$. $\Omega = \Omega$-{i, j}. Go to Step 1.

3.2 Consolidation of up to Three Pallets (C3)

As an extension of (C2), we allow up to three pallets to be consolidated into one pallet. This problem (C3) can be formulated as a binary integer programming problem as follows.

<OC3>

$$\text{Max } Z = \sum_{j} \sum_{k \neq j} M_{jk} \quad (7)$$

$$\sum_{k \neq j} M_{jk} \leq 1, \text{ for all } j \quad (8)$$

$$\sum_{j \neq k} M_{jk} \leq 2, \text{ for all } k \quad (9)$$

$$\sum_{j \neq k} M_{jk} u_j \leq 1 - u_k, \text{ for all } k \quad (10)$$

$$\sum_{k} M_{jk} + \sum_{m} M_{mj} \leq 1, \text{ for all } j \quad (11)$$

$$M_{jk} = 0 \text{ or } 1, \text{ for } all\ j \neq k \quad (12)$$

The equations are identical to those of (OC2) except for Eq. (9), which allows up to two other cells to be consolidated to a cell. A heuristic algorithm for (C3), which is a slight modification of (HC2), is as follows.

<HC3> Heuristic algorithm for C3.
Step 0 : $\Omega = \{1, 2, \cdots, N\}$, $m := 0$.
Step 1 : For i, j $\in \Omega$, select the two smallest u_i and u_j.
Step 2 : Find the largest u_k, k $\in \Omega$, such that $u_i+u_j+u_k \leq 1$; (move i and j to k) Go to Step 3.
If no such u_k exists, go to Step 4 (to try C2).
Step 3 : $m := m+2$. $\Omega = \Omega$-{i, j, k}. Go to Step 1.
Step 4 : For i $\in \Omega$, select the smallest u_i.
Step 5 : Find the largest u_j, j $\in \Omega$, such that $u_i+u_j \leq 1$; (move i to j) Go to Step 6.
If no such u_j exists, stop.
Step 6 : $m := m+1$. $\Omega = \Omega$-{i, j}. Go to Step 4.

3.3 Numerical Example

To clarify the two heuristic algorithms that we proposed in the previous sections, consider a one-sided rack with four levels and six columns. <Table 1> shows in parentheses the current space utilization of each cell. The heuristic algorithm HC2 picks $u_{13} = 0.05$ to be consolidated to $u_{14} = 0.93$; $u_6 = 0.07$ to $u_{10} = 0.83$; $u_{18} = 0.19$ to $u_5 = 0.71$; $u_{11} = 0.23$ to $u_4 = 0.66$; $u_7 = 0.25$ to $u_3 = 0.63$; $u_{24} = 0.27$ to $u_{21} = 0.60$; $u_{23} = 0.29$ to $u_{20} = 0.56$; $u_{16} = 0.34$ to $u_{20} = 0.51$; and $u_8 = 0.39$ to $u_{19} = 0.48$, thus creating nine new empty cells. Solving the BIP model (OC2) given in Section 3.1 using this data yields the same solution.

On the other hand, the heuristic algorithm HC3 picks $u_{13} = 0.05$ and $u_6 = 0.07$ to be consolidated to $u_{10} = 0.83$; $u_{18} = 0.19$ and $u_{11} = 0.23$ to $u_{15} = 0.56$; $u_7 = 0.25$ and $u_{24} = 0.27$ to $u_{19} = 0.48$; $u_{23} = 0.29$ to $u_5 = 0.71$; $u_{16} = 0.34$ to $u_4 = 0.66$; $u_8 = 0.39$ to $u_{21} = 0.60$; and $u_2 = 0.43$ to $u_{20} = 0.51$, thus creating ten new empty cells. Solving the BIP model (OC3) given in Section 3.2 using this data yields the same solution.

Table 1. An example of a rack with 24 cells (cell space utilizations)

1	2	3	4	5	6
(1.00)	(0.43)	(0.63)	(0.66)	(0.71)	(0.07)
7	8	9	10	11	12
(0.25)	(0.39)	(1.00)	(0.83)	(0.23)	(1.00)
13	14	15	16	17	18
(0.05)	(0.93)	(0.56)	(0.34)	(0.82)	(0.19)
19	20	21	22	23	24
(0.48)	(0.51)	(0.60)	(1.00)	(0.29)	(0.27)

3.4 Performance Evaluation

In this section, we will evaluate the average per-

Table 2. Average performance of heuristics for C2 and C3

	C2		C3	
	Heuristic	BIP	Heuristic	BIP
Avg. number of new cells created	145.173	146.102	145.382	146.592
Max error	1.408 %		2.113 %	
Min error	0 %		0 %	
Avg. error	0.636 %		0.825 %	

formance of the heuristic algorithms HC2 and HC3 by comparing their solutions with the optimal solutions obtained by solving the binary integer programming problems, OC2 and OC3, respectively. The binary integer programming problems are solved using CPLEX (Version 9.0), and the heuristic algorithms are coded using C++ (Version, 2005). We consider a one-sided rack of practical size, spec if., with 10 levels and 30 columns. To measure the average performance of the proposed heuristics, we generate random problems as follows. The space utilization (u_i) for each cell is randomly generated between 0 and 1. One thousand random problems are generated and solved by both binary integer programming problems and heuristic algorithms.

<Table 2> shows the average of the maximum number of new empty cells created. For (C2), the heuristic algorithm creates 145.173 new cells, while binary integer programming creates 146.102 new cells, resulting in an average error of 0.636%. The maximum error out of 1,000 replications is 1.408%, which is acceptable. As we consider that solving BIP using software such as CPLEX at a warehouse is unlikely, the heuristic algorithm that we propose seems to be a practical and useful alternative. For (C3), the heuristic algorithm creates 145.382 new cells, while BIP creates 146.592 new cells, resulting in an average error of 0.825%. The maximum error out of 1,000 random problems is 2.113%, which also is acceptable. Note that although (C3) allows three cells to be consolidated into one, (HC3) increases by only 0.14% the number of new cells compared to (HC2), which combines only pairs of cells (C2)-this is a negligible improvement. Similarly, (OC3) increases by only 0.36% the number of new cells compared to (OC2). Hence, we will consider only C2 in the following studies.

4. Generating Q new Empty Cells (GQ)

In the previous two sections, our concern has been to create as many new empty cells as possible, given the current utilization array $\mathbf{U} = (u_i)$ of the rack. This objective will be of interest when enough time is available for consolidation. However, when only a limited amount of time is available for consolidation and the number of incoming pallets is known in advance (which often arises in the context of 'advanced shipping notices' via communication networks between suppliers and DCs), one may be interested only in identifying consolidation to generate a specific number (Q) of new empty cells in the shortest possible time. We name this problem as GQ. We assume the following.

(1) The I/O point of the rack (also called 'dwell point') is located at the lower left corner of the rack.
(2) Starting from the dwell point, the fork of the truck travels along the rectilinear path between the two cells in the rack and moves at a constant speed in the horizontal and vertical directions, respectively. No acceleration or deceleration is considered.
(3) It takes a constant amount of time for the fork of the truck to pick up or deposit a pallet.

We will consider only two pallets to consolidate because the consolidation of three pallets will result in significant additional travel of the fork to move the empty pallet created at the second cell to the I/O point.

4.1 Modeling

We introduce additional notation as follows.

h_{jk}, v_{jk} : Horizontal and vertical distances, respectively, between cells j and k.
s_h, s_v : Horizontal and vertical speeds, respectively, of the folk lift.
t : Time to move the full pallet load to another pallet for consolidation.
$t_{p/d}$: Time to pick up or deposit a pallet.

Then, the problem (GQ) can be formulated as a binary integer programming problem as follows.

<OGQ>

$$\text{Max } Z = \sum_j \sum_{k \neq j} M_{jk} \times \left\{ \frac{h_{oj} + h_{jk} + h_{ko}}{s_h} + \frac{v_{oj} + v_{jk} + v_{ko}}{s_v} + t u_j + 2t_{p/d} \right\} \quad (13)$$

$$\sum_{k \neq j} M_{jk}(u_k - u_j) \geq 0, \text{ for all } j \neq k \quad (14)$$

$$\sum_{k \neq j} M_{jk} \leq 1, \text{ for all } j \quad (15)$$

$$\sum_{j \neq k} M_{jk} \leq 2, \text{ for all } k \quad (16)$$

$$\sum_{j \neq k} M_{jk} u_j \leq 1 - u_k, \text{ for all } k \quad (17)$$

$$\sum_{j} + \sum_{k} M_{jk} \geq Q \quad (18)$$

$$\sum_{k} M_{jk} + \sum_{m} M_{mj} \leq 1 \text{ for all } j \quad (19)$$

$$M_{jk} = 0 \text{ or } 1, \text{ for } all\ j \neq k \quad (20)$$

Eq. (13) represents the objective function of minimizing the time required to generate Q new empty cells. Note that the total time comprises the moving time in the horizontal and vertical directions, respectively, plus the handling time, which is proportional to the quantity of remaining items. The constraints are identical to those of (OC3) except for Eq. (18), which enforces the creation of at least Q new empty cells.

Let $P(q)$ = (i, j) denote the q^{th} consolidation, which moves the items of cell i to cell j; let x_i and y_i denote the column number (from the dwell point) and the level of cell i, respectively; and let $t(i, j)$ denote the time to move from cell i to cell j. Now, we will propose two heuristic algorithms for this problem as follows.

<HGQ_a>
Step 0 : $\Omega = \{1, 2, \cdots, N\}$, $q := 0$, $w := 1$.
Step 1 : For $i \in \Omega$, select the smallest u_i.
Step 2 : Among the cells within the rectangle defined by cell i and the dwell point, find the largest u_j, $j \in \Omega$, such that $u_i+u_j \leq 1$; (move i to j) Go to Step 3.
If no such u_j exists, go to Step 4.
Step 3 : $q := q+1$. $\Omega = \Omega \backslash \{i, j\}$. $P(q) := (i, j)$, $T(q) := t(o, i)+t(i, j)+t(j, o)+tu_i+2t_{p/d}$.
If $q < Q$, go to Step 1; otherwise stop.
Step 4 : Among the cells k such that $1 \leq x_k \leq x_{i+w}$, $y_k := y_{i+w}$ or $x_k := x_{i+w}$, $1 \leq y_k \leq y_{i+w}$, find the largest u_j, $j \in \Omega$, such that $u_i+u_j \leq 1$; (move i to j) Go to Step 3.
If no such u_j exists, $w := w+1$.
If there are no more cells to search, stop; otherwise go to Step 4.

<HGQ_b>
Step 0 : $\Omega = \{1, 2, \cdots, N\}$, $q := 0$.
Step 1 : For $i \in \Omega$, select the smallest u_i.
Step 2 : Find the largest u_j, $j \in \Omega$, such that $u_i+u_j \leq 1$; (move i to j) Go to Step 3.
If no such u_j exists, stop (no feasible solution can be found).
Step 3 : $q := q+1$. $\Omega = \Omega \backslash \{i, j\}$. $P(q) := (i, j)$, $T(q) := t(o, i)+t(i, j)+t(j, o)+tu_i+2t_{p/d}$.
If $q < Q$, go to Step 1; otherwise stop.

4.2 Performance Evaluation

In this section, using simulation, we will compare the two heuristic algorithms described in the previous section, namely, HGQ_a and HGQ_b, in terms of the average performance. The total time taken for generating Q new empty cells using the heuristics can be expressed as $T_Q = \sum_{1 \leq q \leq Q} P(q)$. For 1,000 random problems, we compare T_Q with the optimal objective function value of (OGQ) for Q = 20, 40, and 60 out of 300 cells in the rack. <Table 3> and <Figure 1> show the results. The computation is finished within a few seconds for all the cases in <Table 3> using a Pentium PC.

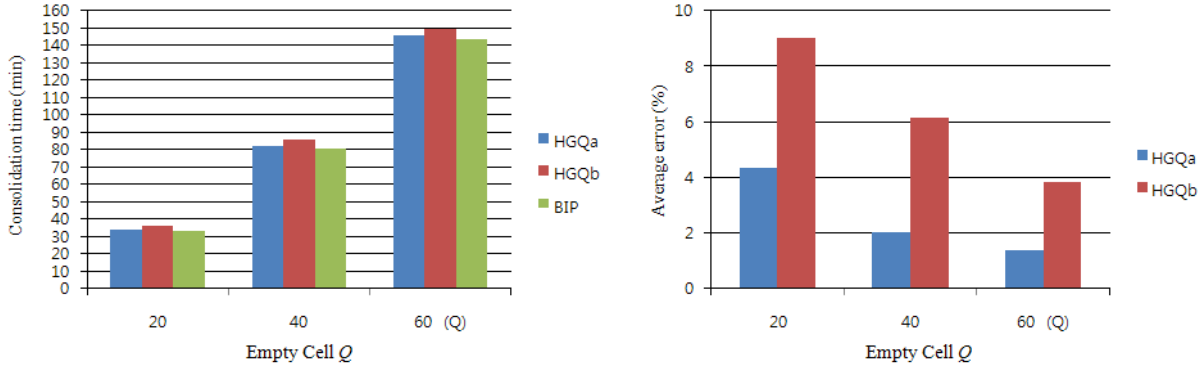
4.3 Regression Model for the Total Consolidation Time

In the previous section, we examined the total time to generate Q new empty cells (T_Q) for three values of Q. Obviously, T_Q is a function of a set of variables including Q. For operating a distribution center, it will be very useful if one can quickly estimate how long it will take to generate Q new empty cells through consolidation in the DC for a given utilization array $\mathbf{U} = (u_i)$ of the rack and various parametric values. To do

Table 3. Average performance of HGQ_a and HGQ_b

	HGQ_a			HGQ_b			OGQ		
	Q = 20	40	60	20	40	60	20	40	60
Average time (min)	34.29	82.42	145.79	35.82	85.73	149.35	32.86	80.76	143.83

	HGQ_a			HGQ_b		
	Q = 20	40	60	20	40	60
Max error	5.21%	3.27%	1.63%	9.56%	6.68%	4.21%
Min error	3.82%	1.98%	1.27%	8.86%	6.02%	3.46%
Average error	4.34%	2.05%	1.37%	9.01%	6.15%	3.83%



(a) Average time spent for generating Q cells (b) Average error (additional time)

Figure 1. Average performance of HGQ_a and HGQ_b

this, in this section, we will identify the relationship between T_Q and a set of parameters using regression.

Among many relevant parameters, we will consider X (time to move horizontally by one cell), Y (time to move vertically by one cell), and t (handling time to move a full pallet load of product to another pallet) as the independent variables, and T_Q (time to generate Q new empty cells) as the dependent variable. Since HGQ_a outperforms HGQ_b as shown in the previous section, we will obtain T_Q using HGQ_a in developing the regression model.

Again, we consider the same configuration of the one-sided rack with 10 levels and 30 columns. For Q = 5, 10, 15, ⋯, 60, 100 random problems are generated by choosing a random value of X from the ranges of $10 \leq X \leq 20$, $Y = 2X$, and $180 \leq t \leq 360$ seconds. For a significance level of $p = 0.05$, it turns out that the regression equation satisfies the basic assumptions of regression, such as normality, independence, and linearity. However, equality of variance is not satisfied; so, we replace T_Q with $\sqrt{T_Q}$. Using SPSS version 12.0, 77 outliers are screened for normality out of 1,200 random datasets; thereafter, we obtain X, t, and Q as significant independent variables with $p = 0.00$. However, Y is eliminated from the regression equation due to multi-collinearity. In the resulting regression equation (21),

R_Q denotes the estimated time to generate Q new empty cells for given values of X and t. The results of the analysis of variance indicate that the regression equation supports the validity of the regression model with an adjusted R^2 of 0.98.

$$\sqrt{R_Q} = -2.5627+1.4170X+0.001t+1.2907Q. \quad (21)$$

For example, if the fork of the truck moves at a speed (X) of 18 seconds per cell horizontally, and the time to move a full pallet load to another pallet (t) is 239 seconds, then for Q = 25, we obtain R_Q = 3,074 seconds from Eq. (21); also, T_Q = 3,198 seconds from HGQ_a, yielding an error of 3.88% relative to the solution from HGQ_a, which is acceptable from a practical point of view. Conversely, using Eq. (21), we can also quickly calculate how many new empty cells can be generated by consolidation in a certain length of time in the DC. For example, suppose the same DC has only 30 minutes; then the number of new empty cells that can be generated during these 30 minutes can be estimated as Q = 14.91.

5. Conclusions

In this paper, we study the pallet consolidation problem for distribution centers. We formulate three problems : creating as many new empty cells as possible by (i) consolidating only two cells (C2) and (ii) consolidating up to three cells (C3); and (iii) generating a specific number (Q) of new empty cells in the shortest time (GQ). For the three problems, we present mathematical formulations and heuristic algorithms. For C2 and C3, the heuristic algorithms yield good solutions with an average error of less than 1% relative to the optimal solution. For GQ, the average error decreases to below 2% as Q exceeds 13% of the total number of cells in the rack, which is practically useful.

To generalize the results, we develop a regression model to estimate the time to generate Q new empty cells with different parametric values such as the fork speed in the horizontal direction and the time to move a full pallet load from one pallet to another. Using this regression equation, one can not only quickly estimate the time for generating Q new empty cells but also estimate the number of new empty cells that can be generated within a specific length of time.

Considering that it is very hard to run any optimization tools such as CPLEX in most distribution

centers, the heuristic algorithms that can be implemented easily on personal computers, besides the regression equation, can be very useful tools for operating distribution centers. Substantial economic benefits from pallet consolidation in DCs can be realized by postponing the construction or leasing of additional storage space. Possible further research includes the case where the fork of the truck moves in both horizontal and vertical directions at the same time, viz., the case of Chebyshev travel.

References

Alvarez-Valdes, R., Parreno, F., and Tamarit, J. M. (2005), A branch-and-cut algorithm for the pallet loading problem, *Computers and Operation Research*, **32**(11), 3007-3029.

Bischoff, E. E. and Ratcliff, M. S. W. (1995), Loading multiple pallet, *Journal of the Operational Research Society*, **44**(11), 1322-1336.

Carpenter, H. and Dowsland, W. B. (1985), Practical considerations of the pallet-loading problem, *Journal of the Operational Research Society*, **36**(6), 489-497.

Chan, F. T. S., Bhagwat, R., Kumar, N., Tiwari, M. K., and Lam, P. (2006), Development of decision support system for air-cargo pallets loading problem : a case study, *Expert Systems with Applications*, **31**(3), 472-485.

Dowsland, K. A. (1987), A Combined data-base and algorithmic approach to the pallet-loading problem, *Journal of the Operational Research Society*, **38**(4), 341-345.

Hodgson, T. J. (1982), A combined approach to the pallet loading problem, *IIE Transactions*, **14**(3), 175-182.

Lau, H. C. W., Chan, T. M., Tsui, W. T., Ho, G. T. S., and Choy, K. L. (2009), An AI approach for optimizing multi-pallet loading operations, *Expert Systems with Applications*, **36**(3), 4296-4312.

Martins, G. H. A. and Dell, R. F. (2007), The minimum size instance of a pallet loading problem equivalence class, *European Journal of Operational Research*, **179**(1), 17-26.

Martins, G. H. A. and Dell, R. F. (2008), Solving the pallet loading problem, *European Journal of Operational Research*, **184**(2), 429-440.

Morabito, R. and Morales, S. (1998), A simple and effective recursive procedure for the manufacturer's pallet loading problem, *Journal of the Operational Research Society*, **49**(8), 819-828.

Scheithauer, G. and Terno, J. (1996), The G4-heuristic for the pallet loading problem, *Journal of the Operational Research Society*, **47**(4), 511-522.

Terno, J., Scheithauer, G., Sommerweiû, U., and Riehme, J. (2000). An efficient approach for the multi-pallet loading problem, *European Journal of Operational Research*, **123** (2), 372-381.

Yaman, H. and Sen, A. (2008), Manufacturer's mixed pallet design problem. *European Journal of Operation Research*, **186**(2), 826-840.