Asia-Pacific Journal of Operational Research

ON THE THREE-DIMENSIONAL CONTAINER PACKING PROBLEM UNDER HOME DELIVERY SERVICE*

WAN-YU LIU[‡], CHUN-CHENG LIN[§,†], and CHANG-SUNG YU[¶]

[‡]*Dept. of Applied Natural Resources*
Aletheia University
Tainan 721, Taiwan

[§]*Dept. of Industrial Engineering and Management*
National Chiao Tung University
Hsinchu 300, Taiwan

[¶]*Dept. of Business Administration*
National Taiwan University
Taipei 106, Taiwan

[†]*Corresponding author's E-mail: cclin321@nctu.edu.tw*

Home delivery service is one of the most important cost drivers in e-commerce industry. We consider the three-dimensional container packing problem under home delivery service, where each rectangular item with its specific destination is loaded orthogonally onto a rectangular container so that the utilization rate of the container space is maximized. In our framework, we assume the routing of a consignment to be given, which turns out that there is an order of unloading items with respect to the consignment. If we load items without considering the order of unloading items, we may unload and reload other unconcerned items drastically while unloading the required item. Therefore, in this paper, the unloading costs for a consignment are precisely defined according to the *invisible and untouchable rule*, and a subvolume scheme based algorithm is proposed. Our experimental results suggest our approach to be promising.

Keywords: Container packing problem; home delivery; transportation; logistics.

1. Introduction

Since Wal-Mart and Toys'rus failed to deliver on-line Christmas orders in 1999 (*The Economist*, 2000), managing the logistics operations for electronic commerce has received more and more attention. Besides, due to the impact of Internet bubble in 2000, most of the dotcom companies confront financial problems except for some profitable ones. A study concluded that over 40% cost reduction for dotcom companies can be achieved by offering *home delivery service* (Kämäräinen et al.,

*A preliminary version of this work was presented at the 2006 IEEE Conference on Systems, Man, and Cybernetics (SMC 2006), Oct. 8-10, 2006, Taipei, Taiwan.
[†]Corresponding author.

2000). *Home delivery service* provides an efficient and simple service, whereby authorized home delivery provider delivers prescribed cargo directly to customers or the collection points closest to customers. In order to reduce operating costs of companies, the home delivery service for electronic commerce has some key elements of cost-efficiency, e.g., the product range offered, the type of reception, the length of the delivery time window, order efficiency, handling efficiency, customer density, delivery hours, and so on (Punakivi and Saranen, 2001; Punakivi, 2003).

Three-dimensional container packing problem (3DCPP) is a crucial issue among logistics operations, which is concerned with packing a number of rectangular items (cargos) orthogonally onto a rectangular container so that the utilization rate of the container space or the total value of loaded items is maximized. The 3DCPP has been known to be NP-hard (since the uni-dimensional packing problem (Garey and Johnson, 1979) can be reduced to this problem). Most of the conventional approaches to the problem are based upon meta-heuristic algorithms, e.g., genetic algorithm (Gehring and Bortfeldt, 1997), tabu search (Bortfeldt et al., 2003), and simulated annealing (Jin et al., 2004), which were tested using larger portions of running time, and did not consider the costs induced by home delivery. Therefore, they are not suitable for the home delivery service. For instance, loading cargos without considering the order of unloading cargos in a consignment would increase the difficulty of searching and unloading the required cargo in each collection point, which induces huge costs of unloading cargos. As a result, it is interesting and of urgent importance to develop an efficient and effective algorithm for the 3DCPP under home delivery service. Note that, in addition to the 3DCPP, there also exist a variety of methods for the 2D container packing problem (e.g., see a recent work in (Chen and Huang, 2007)).

When considering the 3DCPP under home delivery service, we suppose that:

- *Door-to-Door service*: A single consignment (container) is required to deliver the order directly to customers' homes. For simplicity, we assume an extreme scenario where each customer only orders an item, i.e., each item has its own destination. Therefore, when the routing of a consignment is given, an ordering of unloading items is also determined.
- *Small shipments*: Door-to-door service has tight order-to-delivery lead time and delivery time windows, and hence the items must be of small sizes (generally, the sum of three dimensions of an item is about less than 220 cm) so that a single delivery person can accomplish each delivery on time. In addition, since e-commerce companies often provide a variety of on-line products to motivate the purchase, it is reasonable to assume the types of items to be heterogeneous.
- *Unloading costs*: Given an ordering of unloading items, it is necessary to avoid unloading and reloading other unconcerned items many times. The times of unloading and reloading the unconcerned items lead to significant unloading costs.

Notice that the unloading ordering due to door-to-door service and the small-size items due to small shipments can be viewed as the given premise conditions of the problem, so it suffices to concern the computation of the unloading costs while programming the problem. In addition to maximizing the utilization ratio of the container space, minimizing the total unloading costs might be regarded as an additional objective or a part of a single objective of the problem. However, such programming problems still may lead to nonzero unloading costs, which are not suitable for practical applications because it is difficult and inconvenient for delivery people to find out and unload the required item at each customer's home (i.e., they may need to unload and reload other unconcerned items many times, and record the locations of those reloaded items in the container). Therefore, we suggest that zero unloading cost should be viewed as a constraint of the programming problem.

In this paper we have precisely defined the unloading costs according to the *invisible and untouchable rule* and modified the *iterative subvolume-based heuristic algorithm* (Jin et al., 2004) to solve the 3DCPP with zero unloading cost constraint, which is also an intractable problem like 3DCPP. As for experimental analysis, our approach is compared with the conventional approaches (Bischoff and Ratcliff, 1995; Jin et al., 2004) which solve similar problems (the 3DCPP for multi-drops situations), by using standard benchmark data set, from OR-Library (Beasley, 1990; *OR-Library*, 2007). The experimental results display that the packing patterns produced by our approach have the advantages of not only no unloading costs but also high utilization ratios.

The rest of the paper is organized as follows: Some preliminaries are given in Section 2. The definition of unloading costs and our approach to the 3DCPP with zero unloading cost are given in Section 3. Section 4 gives a number of experimental results and some discussion. Finally, a conclusion with future work is given in Section 5.

2. Preliminaries

In this section, we give the literature review on our concerned problem, a brief introduction to home delivery models, and some concerned basic settings.

2.1. *Literature Review*

This paper considers some practical requirements to solve the three-dimension container packing problem (3DCPP), which aims to orthogonally pack a subset of some given rectangular boxes into a single rectangular container of fixed dimensions. The 3DCPP is a variant of the cutting and container problems (C&P), and a complete typology for the C&P can be referred to (Wäscher et al., 2007).

Generally speaking, each solution for the 3DCPP includes two components: the *packing approach* and the *heuristic* to construct the optimal packing.

Most *packing approaches* for the 3DCPP fall into the following categories: The *wall-building* approach (e.g., see George and Robinson, 1980) constructs vertical wall

sections across the container length or width, whereas the *layer-building* approach (e.g., see Gilmore and Gomory, 1965) builds the loading plan layer by layer from the floor of the container upwards. In most of the two approaches, *ranking rules* are applied to accommodate items of large sizes in earlier stages. The *stack* or *subvolume* approach (e.g., see Bischoff and Ratcliff, 1995; Abdou and Elmasry, 1999; Abdou and Yang, 1994) considers the mixed patterns of column stacking and layer packing. Instead of requiring the packed items to form a flat layer, the *multi-faced-building* approach (e.g., see Lim et al., 2003) treats each of the four walls in the container as a base or a floor to locate items. The *block building* approach (e.g., see Mack et al., 2004) is to pack the container with mostly only boxes of a single type with the same spatial orientation. Note that the approach is related to the layer-building approach, but its main motivation is to pack the largest-possible subspaces of the container without any internal loss. The *guillotine cutting* approach (e.g., see Morabito and Arenales, 1994) represents a packing plan by a slicing tree, in which each slicing subtree corresponds to a consecutive session of the container into smaller pieces by means of guillotine cuts, whereby the leaves correspond to the packed boxes.

Some known *heuristics* to construct the optimal packing for the 3DCPP are given as follows: *Greedy* and *local search* heuristics for the 3DCPP were proposed, e.g., (Bischoff and Ratcliff, 1995; Lim et al., 2003). The local minimum problems for the 3DCPP may be solved by some *metaheuristics search strategies*: genetic algorithm (e.g., see Gehring and Bortfeldt, 1997), tabu search (e.g., see Bortfeldt et al., 2003), simulated annealing (e.g., see Jin et al., 2004), greedy randomized adaptive search procedure (e.g., see Moura and Oliveira, 2005; Parreno et al., 2008), and so on. *Tree search algorithms* were applied successfully to the 3DCPP, e.g., see (Parreno et al., 2008; Fanslau and Bortfeldt, 2009).

For most practical applications, some other constraints may be imposed on the C&P problems, and the overview of the practical requirements can be referred to the work of (Bischoff and Ratcliff, 1995). Among them, the requirements that this paper are mainly concerned with are the following:

- *Orientation constraints*: Items may be allowed to be rotated with only specific orientations. For example, the instruction 'this way up' is marked on cardboard boxes.
- *Load stability*: An easily-damaged item should not move significantly during delivery. In addition, it may not be safe for a delivery person to load and unload an unstable item. This paper is concerned about the situation where a large-size item is placed above a small-size one. In this situation, the large-size item may move significantly during delivery, or even mash the small-size one. The subvolume approach is designed for handling this situation (Abdou and Elmasry, 1999).
- *Multi-drop situations*: In the case where there are a lot of different destinations in a single consignment (container), the items for the same destination should be placed together to avoid unloading and reloading other uncon-

cerned items many times. Bischoff and Ratcliff (1995) considered that the items for the same destination form a subset, and they used an analogy of wall-building approaches (George and Robinson, 1980), which packs in a subset the items for the last destination and then those for the second to the last destination, and so on. Jin et al. (2004) applied a subvolume technique coupled with the simulated annealing algorithm to handle the 3DCPP, and their approach with a little variation (called a *subvolume-based iterative procedure* (*SBIP*), not yet simulated annealing) can be applied to multi-drop situations.

Note that, in fact, the situation that we consider in this paper is under the assumption that each item has its own specific destination, and hence it can be viewed as an extreme of the multi-drop situations. Therefore, in order to measure the performance of our approach, our approach is compared to the work of (Bischoff and Ratcliff, 1995) and (Jin et al., 2004) for multi-drop situations. From our later experimental evaluation, those approaches for multi-drop situations are not useful for handling our requirements.

In this paper, we apply the subvolume approach, for meeting the practical requirement of loading items stably. The basic idea of the subvolume approach is stated as follows: A *subvolume* is a rectangular container. As shown in Figure 1, consider a subvolume j expressed by $subv_j(x_j, y_j, z_j, L_j, W_j, H_j)$ where (x_j, y_j, z_j) is its origin (left-bottom-back corner vertex), and L_j, W_j, H_j are its length, width, and height, respectively.

Every time when an item is packed into the subvolume j, three new subvolumes, say $subv_{1'}, subv_{2'}, subv_{3'}$, at maximum are generated from the up side, right side, and front side of the item. A list of available subvolumes, denoted by $SubV$, is updated in each iteration of the algorithm. Note that in order to produce larger-size subvolumes so that larger-size items can be packed, the subvolumes are always generated along the shorter direction of L_j and W_j, as shown in Figure 1.

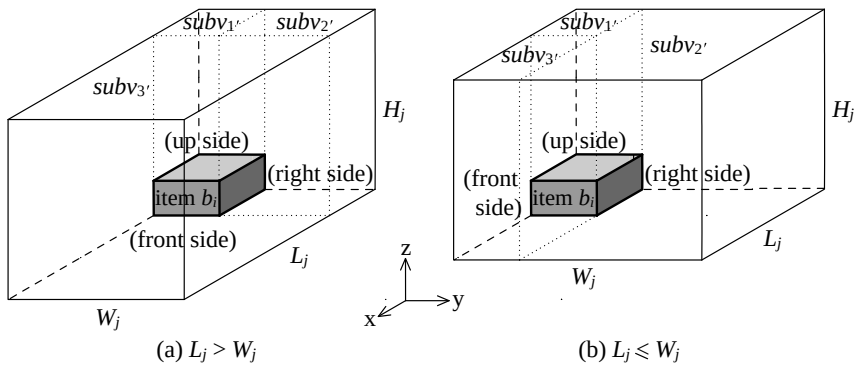


Fig. 1. Illustration of the subvolume.

In addition, there are two operations of subvolumes needed to be executed after inserting three subvolumes at maximum into $SubV$. The first is the *combining* operation, in which two adjacent subvolumes with the same height and the same length or width are combined into a larger subvolume, which allows bigger items to be packed in later iterations. The second is the *repartitioning* operation, in which two adjacent subvolumes with the same height are partitioned into two new subvolumes with dimensions different from the original ones, so that it makes the lengths of straight guillotine cuts (Beasley, 1985) be reduced.

Jin et al. (2004) proposed a *subvolume-based iterative procedure* (*SBIP*) for the multi-drops situations, but did not give any experimental results about the situations. For a given loading order, their approach is iterative to sequentially take an item in the loading order, and randomly select a subvolume from the list of subvolumes to pack the item. Since the subvolume is selected randomly, their approach leads to undeterministic results.

2.2. *Home Delivery Models*

The home delivery services with several different service levels (which is measured by the delivery time window offered for the customer, i.e., how long the customer has to stay at home waiting for the delivery) have been provided. According to the degree of service level, the service can be further classified into two models: *attended reception* and *unattended reception* (Kämäräinen and Punakivi, 2002). *Attended reception*, with one-hour to even five-hour delivery time window, claims the company or a third party to deliver the order to a given place, where the customer accepts the delivery; *unattended reception*, by which the customer is independent of delivery timetables, claims the company or a third party to deliver the order to a reception box, reception lockers, or delivery box. Intuitively, the unattended reception model delivers to fewer collecting points than the attended reception model, and it does not fail to deliver because it is not concerned with whether the customer is at home. Consequently, the unattended reception model commonly has lower operative cost level, and Punakivi and Saranen (2001) showed that the cost savings were up to 60%.

However, the unattended reception model is not universal because of the following reasons:

- The costs of investing in personal or shared reception boxes are huge. Most of the e-commerce companies are of small size or starting enterprises. It is not easy for them to raise funds for paying for the reception boxes.
- The delivery range may be widespread, and hence a huge number of reception boxes are required in order to reach the "last mile" operation.
- The sizes of the reception boxes should be considered. A customer might order an item with a size larger than the size of the reception box or an item which is packed into a nearly full reception box such that there is no room for other orders.

- The placements of reception boxes need to be charged. If a reception box is shared by a group of people, a locker would be hired.

As a result, the companies with the above consideration should be suitable to apply the attended reception model. That is, our assumption of door-to-door service would be reasonable.

2.3. *Basic Settings*

Consider a container $\mathbf{C} = \{(x, y, z) \in \mathbb{R}^3 | 0 \leq x \leq L, 0 \leq y \leq W, 0 \leq z \leq H\}$ where L, W, and H are its length, width, and height, respectively. Also consider a set of n items (rectangular boxes) expressed by $\mathbf{B} = \{b_i | b_i = (l_i, w_i, h_i) \in \mathbb{N}^3, i \in \{1, \cdots, n\}\}$, in which $l_i \leq L$, $w_i \leq W$, and $h_i \leq H$ are respectively the length, width, and height of item b_i. The *packing pattern* is a general term for the packed items and their positions and orientations in a container. Let P be a mapping from a subset $\mathbf{S}$ of $\mathbf{B}$ into $\mathbf{C}$, i.e., $P : \mathbf{S} \to \mathbb{N}^6$, such that

$$P(b_i) = \big(p^x(b_i), p^y(b_i), p^z(b_i), p^l(b_i), p^w(b_i), p^h(b_i)\big)$$

where $b_i \in \mathbf{S}$; $(p^x(b_i), p^y(b_i), p^z(b_i))$ is the coordinate of the origin of item b_i in container $\mathbf{C}$; $(p^l(b_i), p^w(b_i), p^h(b_i))$ is a permutation of (l_i, w_i, h_i), used in the case when the rotation of item b_i is allowed. Note that the coordinate of the origin of any rectangular solid is referred to its left-bottom-back vertex in the $\mathbb{N}^3$ space. The packed field occupied by item $b_i \in \mathbf{S}$ in the container is expressed as $\mathbf{R}(b_i) = [P^x(b_i), P^x(b_i) + P^l(b_i)] \times [P^y(b_i), P^y(b_i) + P^w(b_i)] \times [P^z(b_i), P^z(b_i) + P^h(b_i)]$. Consequently, the 3DCPP can be expressed as follows:

$$Maximize \quad \sum_{b_i \in \mathbf{S}} l_i w_i h_i / (LWH) \tag{2.1}$$

$$subject\ to \quad P^x(b_i) + P^l(b_i) \leq L, \forall b_i \in \mathbf{S} \tag{2.2}$$

$$P^y(b_i) + P^w(b_i) \leq W, \forall b_i \in \mathbf{S} \tag{2.3}$$

$$P^z(b_i) + P^h(b_i) \leq H, \forall b_i \in \mathbf{S} \tag{2.4}$$

$$\mathbf{R}(b_i) \cap \mathbf{R}(b_j) = \emptyset, \forall b_i, b_j \in \mathbf{S}, i \neq j \tag{2.5}$$

, where the objective function is to maximize the utilization rate of the container space; Constraints (2.2) – (2.4) mean that each loaded item in $\mathbf{S}$ must be entirely enclosed in the container, packed orthogonally, and oriented in all three dimensions; Constraint (2.5) means that there are no space overlapping between any two loaded items.

Considering an item b_i packed into subvolume $subv_j(x_j, y_j, z_j, L_j, W_j, H_j)$, the item is placed at the origin of the subvolume, i.e., $P(b_i) = (x_j, y_j, z_j, p^l(b_i), p^w(b_i), p^h(b_i))$. Then, three new subvolumes $subv_{1'}$, $subv_{2'}$, and $subv_{3'}$ at maximum are generated from the up side, right side, and front side of item b_i, in which subvolume $subv_{1'}$ is expressed as follows:

$$subv_{1'}(x_j, y_j, z_j + p^h(b_i), p^l(b_i), p^w(b_i), H_j - p^h(b_i)),$$

and subvolumes $subv_{2'}$ and $subv_{3'}$ are expressed as follows: if $L_j > W_j$ (see Figure 1(a)), then

$$subv_{2'}(x_j, y_j + p^w(b_i), z_j, p^l(b_i), W_j - p^w(b_i), H_j)$$
$$subv_{3'}(x_j + p^l(b_i), y_j, z_j, L_j - p^l(b_i), W_j, H_j);$$

otherwise (see Figure 1(b)),

$$subv_{2'}(x_j, y_j + p^w(b_i), z_j, L_j, W_j - p^w(b_i), H_j)$$
$$subv_{3'}(x_j + p^l(b_i), y_j, z_j, L_j - p^l(b_i), p^w(b_i), H_j).$$

In this paper, we assume the routing of a consignment to be given, and consequently the order of unloading items is determined with respect to the consignment. The loading order can be expressed by a permutation $\pi = (I_1, I_2, \cdots, I_n)$ of $\{1, 2, \cdots, n\}$ where $I_i \in \{1, 2, \cdots, n\}$ and $I_i \neq I_j, \forall i \neq j$. Intuitively, the unloading order is the reverse of the loading order, but if only $m \leq n$ items are loaded in the container, then the unloading order is $\pi^R = (I_m, I_{m-1}, \cdots, I_1)$. Also let $\pi(b_i)$ (resp., $\pi^R(b_i)$) denote the loading (resp., unloading) order of b_i. Note that it is reasonable that the latter $(n-m)$ items in the loading order are chosen not to be loaded in the container instead of the former $(n-m)$ items because an item with latter loading order means that its destination is not far from the starting point, and hence may have lower opportunity costs.

The *lexicographic order* for $\mathbb{N}^2$ is an ordering such that if $(x_1, y_1) < (x_2, y_2)$ iff either 1) $x_1 < x_2$ or 2) $x_1 = x_2$ and $y_1 < y_2$. The lexicographic order for $\mathbb{N}^3$ can be extended by recursively using the above definition.

3. The Packing with Zero Unloading Cost Constraint

In this section, we precisely define the costs of unloading items according to the *invisible and untouchable rule*, and then address our algorithm.

3.1. *Definition of Unloading Costs*

Recall that an ordering of loading items (whose inverse is the unloading order) with respect to a consignment is assumed to be determined, and only first m items are loaded. We define that the *unloading cost* $uc(b_i)$ of an item b_i $(i \leq m)$ with loading order $\pi(b_i) = I_k$ is directly proportional to the number $un(b_i)$ of the items with loading orders in $\{I_t | 1 \leq t < k\}$ required to be unloaded and reloaded such that item b_i can be unloaded. In this paper, number $un(b_i)$ is calculated according to the *invisible and untouchable rule* as follows: an item with a loading order in $\{I_t | 1 \leq t < k\}$ is counted as one unit of $un(b_i)$ if

- a part of the item is *invisible* to the front side of the container, or
- the item is *untouchable* by the delivery person due to the constraint of the delivery person's figure.

For simplification of analysis, while unloading item b_i, unloading an other unconcerned item and then reloading it are counted as only one unit of $un(b_i)$, because unloading an unconcerned item is always accompanied with reloading it. Note that all the items only can be taken out from the front side of the container and the delivery person may be short in figure such that the delivery person cannot touch the required item.

Consider an item b_i packed at $P(b_i) = (x_i, y_i, z_i, l'_i, w'_i, h'_i)$ in the container (see Figure 2, in which (b) illustrates three elevations of (a)). Assume that item b_i is visible to the delivery person (i.e., he/she stands at the front side of item b_i) and he/she stands at the closest position from item b_i. Let $L_{touchable}$ and $H_{touchable}$ denote the distance between item b_i and the delivery person's feet along the x-axis and the z-axis, respectively (see Figure 2(a)). If the delivery person can touch item b_i with his/her hands, $L_{touchable}$ and $H_{touchable}$ have to satisfy the following constraints:

$$L_{touchable} + H_{touchable} \leq 200 \tag{3.7}$$

$$L_{touchable} \leq \min\{200 - z_i, 60\} \tag{3.8}$$

where the constants 200 (cm) and 60 (cm) can be adjusted according to the delivery person's figure. Specifically, Inequality (3.7) means that the item is touchable if the distance from the delivery person's feet to the item is no greater than the sum of the delivery person's body height and arm length (200cm), in terms of axis-parallel distances. In Inequality (3.8), if $200 - z_i \geq 60$ (i.e., $z_i \leq 140$, which means the item to be packed at a lower location), then the the farthest distance between the delivery person's feet and the item along the x-axis is his/her arm length (60cm); otherwise (i.e., the item is packed at a higher location), he/his has to stand closer to touch the item, and the distance $L_{touchable}$ depends on how high the item is packed (i.e., $L_{touchable} \leq 200 - z_i < 60$cm).

Therefore, according to the above invisible and untouchable rule and whether the item can be taken out without unloading other unconcerned items, we define the *untakeout field* $\mathbf{F}(b_i)$ of item b_i as shown in the shaded areas of Figure 2(b), which is characterized as follows:

$$\mathbf{F}(b_i) = (\mathbf{F}_1(b_i) \cup \mathbf{F}_2(b_i) \cup \mathbf{F}_3(b_i)) \cap \mathbf{F}_y(b_i) \cap \mathbf{C}$$

where

$$\begin{aligned}
\mathbf{F}_1(b_i) &= \{(x, y, z) \in \mathbb{N}^3 | x_i < x < x_i + l'_i, z > z_i + h'_i\},\\
\mathbf{F}_2(b_i) &= \{(x, y, z) \in \mathbb{N}^3 | x_i + l'_i < x < x_i + l'_i + \min\{200 - z_i, 60\}, z > z_i\},\\
\mathbf{F}_3(b_i) &= \{(x, y, z) \in \mathbb{N}^3 | x_i + l'_i + \min\{200 - z_i, 60\} < x < L\},\\
\mathbf{F}_y(b_i) &= \{(x, y, z) \in \mathbb{N}^3 | y_i < y < y_i + w'_i\}.
\end{aligned}$$

How to define $\mathbf{F}(b_i)$ is explained as follows. Consider to pack item b_j. $\mathbf{R}(b_j)$ denotes the packed field of item b_j. The way to pack item b_j is discussed as follows:

- If $\mathbf{R}(b_j) \cap \mathbf{F}_1(b_i) \cap \mathbf{F}_y(b_i) \neq \emptyset$, which means that item b_j is packed over item b_i, then item b_i cannot be taken out before unloading item b_j.
- If $\mathbf{R}(b_j) \cap \mathbf{F}_2(b_i) \cap \mathbf{F}_y(b_i) \neq \emptyset$, which means that part of item b_i is invisible due to the coverage of item b_j or other items below item b_j, then the whole box of b_i cannot be seen before unloading item b_j.
- If $\mathbf{R}(b_j) \cap \mathbf{F}_3(b_i) \cap \mathbf{F}_y(b_i) \neq \emptyset$, which means that the delivery person stands too far to touch item b_i, or part of item b_i is invisible due to the coverage of item b_j or other items below item b_j, then item b_i cannot be taken out before unloading item b_j.

Note that item b_i still can be unloaded if item b_j is placed inside $\mathbf{F}_4(b_i)$ in Figure 2(b), i.e., the white cube in Figure 2(a).

As a result, for each item b_i with loading order $\pi(b_i) = I_k$, an item b_j with loading order $\pi(b_j) = I_l$ in $\{I_t | k \leq t < m\}$ is identified as a unit of $un(b_i)$ by checking if the packed field $\mathbf{R}(b_j)$ of item b_j overlaps the untakeout field $\mathbf{F}(b_i)$. To simplify the problem, we assume the unloading cost $uc(b_i)$ of item b_i equal to the number $un(b_i)$. Hence, the total unloading costs with respect to a packing pattern is $\sum_{b_i \in \mathbf{S}} uc(b_i) = \sum_{b_i \in \mathbf{S}} un(b_i)$. Note that we assume that the items which are not packed in the container not to induce any unloading costs, and hence only m items need to be considered when calculating the unloading costs.

3.2. *Our Algorithm*

According to our explanation in the previous sections, it is more appropriate to consider the zero unloading cost as an additional constraint (called *zero unloading cost constraint*) as follows:

$$\sum_{b_i \in \mathbf{S}} uc(b_i) = \sum_{b_i \in \mathbf{S}} un(b_i) = 0.$$

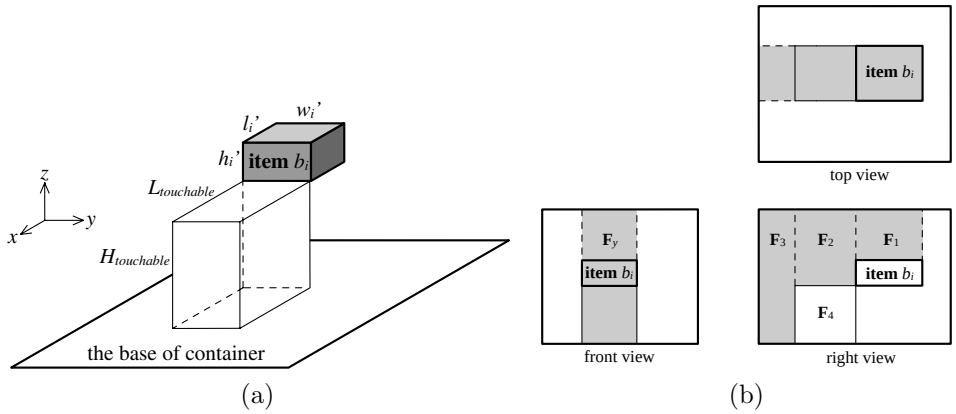


Fig. 2. (a) Illustration of an item packed in a container. (b) Three elevations of (a). An item that overlaps the shaded region leads to a unit of unloading costs.

That is to say,

$$\mathbf{F}(b_i) \cap \mathbf{R}(b_j) = \emptyset, \forall b_i, b_j \in \mathbf{S}, \pi(b_j) < \pi(b_i). \tag{3.9}$$

So the 3DCPP with zero unloading cost constraint can be characterized by the objective equation of (2.1) with the constraints of (2.2)–(2.5) and (3.9).

Our algorithm based on the subvolume scheme mentioned in Section 2 is presented in Algorithm 3.1. In Step 1, the algorithm starts with a list $SubV$ of available subvolumes which initially only involves a subvolume with the size of the container. In each iteration of the for loop in Step 2 – 10, we take out an item in the loading order of items, and under the zero unloading cost constraint select a suitable subvolume from $SubV$ according to the (x, y, z)-lexicographic order of origins of subvolumes in Step 3. Note that, if several orientations are possible in Step 5, then a random orientation is selected. In Steps 6–7, the item is inserted into the chosen subvolume, and then three new subvolumes at maximum from the up-side, right-side, and front-side of the item are generated and inserted into list $SubV$. Steps 8 and 9 are some operations for adjusting subvolumes. The combining and repartitioning operations in Step 9 are the conventional setting of the subvolume approaches.

Algorithm 3.1 3DCPP-Home-Delivery()

Input: a set of n items with a loading ordering $\{I_1, I_2, \cdots, I_n\}$, and a container.
Output: a packing pattern without unloading costs.

1: initialize a list $SubV$ of available subvolumes, which only includes a subvolume with the size of the container.
2: **for each** $k = 1$ to n **do**
3: from list $SubV$ according to the (x, y, z)-lexicographic order, select a subvolume which accommodates the item b_i of loading order I_k with an allowed orientation and does not induce any unloading costs.
4: if there is no subvolume suitable for item b_i, then break the for loop.
5: adjust (l_i', w_i', h_i') according to the determined orientation.
6: insert item b_i into the selected subvolume (i.e., set the coordinate (x_i, y_i, z_i) of the origin of item b_i).
7: generate three new subvolumes at maximum from up side, right side, and front side of item b_i, and insert them into $SubV$ so that the subvolumes in $SubV$ are maintained in the (x, y, z)-lexicographic order.
8: if the insertion of item b_i leads to any subvolume in $SubV$ whose part is invisible to the front side of the container (i.e., this subvolume must induce unloading costs while loading any later item into it), then we need to cut off the part of each subvolume which is sheltered by item b_i, as shown in Figure 3. And re-insert the remaining parts of those subvolumes into $SubV$.
9: execute the combining and the repartitioning operations of subvolumes, which are mentioned in Section 2.
10: **end for**

It is of interest in the future to investigate how much number or percentage of the iterations which experiences the two operations influences the solution quality and the running time.

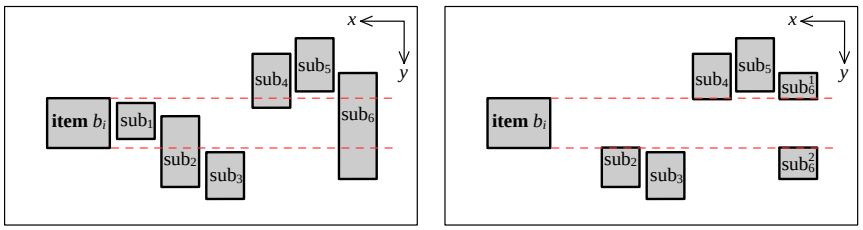
Note that list $SubV$ is sorted in the (x, y, z)-lexicographic order, and hence the selection in Step 3 of Algorithm 3.1 can be completed by reading the list sequentially in linear time. Also note that the lexicographic order can be easily maintained in linear time when we insert a new subvolume to $SubV$.

Note that Algorithm 3.1 is stochastic (in which Steps 3 and 9 are executed stochastically), so its outputs are not deterministic. Fortunately, it runs very fast to execute Algorithm 3.1 once. After a large number of trials, we observe that it takes about two minutes on average to execute 100,000 times of Algorithm 3.1 on a problem, and such running time is feasible in practice. Hence, it is suggested to execute 100,000 times of Algorithm 3.1 for practical applications.

Algorithm 3.1 follows a "strict" ordering of loading items, i.e., the algorithm stops if there is no suitable subvolume for current item (see Step 4). In fact, it would be reasonable that the algorithm is allowed to follow a "non-strict" loading ordering, i.e., in Step 4, the algorithm do not pack the current item, but continues to pack the remaining items in the loading ordering. By doing so, the final packing pattern still satisfies the constraint of zero unloading cost, but some items in the loading ordering between the first and the last packed items may not be loaded. Such a variant algorithm is stated in Algorithm 3.2. It should also be noticed that the "non-strict" loading ordering may not be universal for some small companies, which may run a small business and afford only one container consignment each day. If all the cargos must be delivered within a day and an additional consignment is not allowed, then the "non-strict" loading ordering is impossible.

4. Implementation and Experimental Results

Based on our approach detailed in the previous section, we develop a prototype system for testing the benchmark data set from OR-LIBRARY (Beasley, 1990; *OR-Library*, 2007). The implementation is in C++ using OpenGL library, and runs



(a) top view.

(b) top view after adjusting the invisible subvolumes.

Fig. 3. Illustration of adjusting the invisible subvolumes. The part of a subvolume invisible to the front side of the container needs cut off.

Algorithm 3.2 3DCPP-HOME-DELIVERY'()

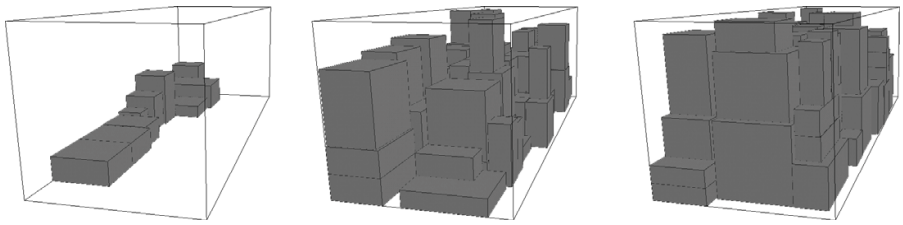
Input: a set of n items with a loading ordering $\{I_1, I_2, \cdots, I_n\}$, and a container.
Output: a packing pattern without unloading costs but following a non-strict loading ordering.

The algorithm is almost the same as Algorithm 3.1 except that Step 4 is replaced by:

4': if there is no subvolume suitable for item b_i, then go to Step 2.

on an Intel Core 2 Duo E8400 3.00 GHz PC with memory of size 2.00 GB running Windows XP. In order to compare the performance of our approach, we also implement Bischoff and Raticliff's (BR) (1995) and Jin et al.'s SBIP (2004) approaches (which, to our understanding, are the only two approaches used for solving the 3DCPP in multi-drops situations). However, since the BR approach, loading only one item per destination, is obviously not well-suited for our problem, thus only the SBIP approach is compared to our approach in the following. The statistics of our experimental results is given in Table 1. In addition, we also develop a visualization interface for packing patterns (see Figure 4 as an example). The interface allows users to observe how to construct the packing pattern step by step (see Figure 5) and interact with it, by which a more compact packing pattern might be generated according to users' hints. From Figure 4, we observe that there is some empty space in the packing pattern which seems to make some items loaded unstably, but in fact, the problem can be solved by applying empty boxes or foam rubber to filling out the empty space in practice.

Here we only give the experimental results of the benchmark problem set of the file entitled by 'thpack7.txt' (which involves 100 problems) because the types of the items provided by each problem in this problem set are the most heterogeneous among the others, so as to meet the requirement of the home delivery service applied in e-commerce industry. It also should be noticed that we assume a loading order



(a) BR (b) SBIP (c) our

Fig. 4. The visualization of the packing patterns for problem No. 7 of the *thpack7* benchmark problem set.

Table 1. Statistics of the experimental results for the *thpack7* problem set from OR-library, where the statistics of both the SBIP and our algorithms are measured by executing 100,000 times of the algorithms.

	10^5 times of SBIP				10^5 times of Algorithm 3.1				10^5 times of Algorithm 3.2			
No.	items loaded	utiliz. rate	unload. cost	run time (s)	items loaded	utiliz. rate	unload. cost	run time (s)	items loaded	utiliz. rate	unload. cost	run time (s)
1	58	56.24%	247	27.94	53	51.85%	0	27.91	69	55.90%	0	68.50
2	62	51.00%	76	59.58	67	54.74%	0	54.17	84	56.54%	0	103.70
3	60	49.67%	48	81.97	65	52.62%	0	55.25	76	56.67%	0	79.92
4	70	47.09%	131	91.84	70	47.09%	0	67.24	91	53.75%	0	118.16
5	64	41.80%	219	56.51	61	38.20%	0	51.52	74	45.84%	0	79.57
6	84	51.70%	311	142.66	80	48.94%	0	71.31	85	49.64%	0	113.62
7	55	45.32%	74	57.22	66	54.79%	0	49.76	71	56.56%	0	76.61
8	60	45.14%	145	39.12	57	43.29%	0	34.47	69	47.44%	0	62.22
9	76	56.15%	179	132.59	76	56.15%	0	70.13	81	58.13%	0	83.75
10	63	49.16%	138	58.72	63	49.16%	0	47.55	84	54.17%	0	87.59
11	71	50.92%	178	75.65	72	51.46%	0	65.74	87	57.11%	0	117.07
12	81	50.47%	207	128.56	81	50.47%	0	92.78	91	53.98%	0	118.86
13	80	48.81%	306	93.53	78	47.44%	0	67.38	86	50.15%	0	118.06
14	60	52.18%	93	59.25	61	53.92%	0	47.06	72	57.66%	0	70.44
15	66	55.96%	307	55.3	64	54.14%	0	46.01	74	57.33%	0	67.17
16	63	58.23%	230	31.38	53	52.19%	0	26.93	63	55.67%	0	55.64
17	63	51.27%	251	39.97	63	51.27%	0	36.17	73	54.64%	0	70.65
18	57	50.77%	158	34.91	56	50.23%	0	34.25	71	54.19%	0	62.50
19	52	45.24%	168	37.03	52	45.24%	0	29.91	68	50.29%	0	54.86
20	66	50.41%	290	75.91	63	47.43%	0	46.14	72	53.35%	0	83.93
21	62	44.60%	115	45.47	58	41.19%	0	42.65	70	48.13%	0	74.59
22	62	53.06%	77	81.84	69	59.29%	0	61.61	81	60.76%	0	93.25
23	74	47.31%	261	53.00	72	46.86%	0	51.09	88	51.00%	0	123.22
24	56	48.86%	200	32.2	52	45.98%	0	26.72	61	50.80%	0	42.97
25	62	48.55%	141	60.58	61	47.51%	0	43.22	68	51.09%	0	60.08
26	74	47.04%	301	45.25	65	42.92%	0	41.97	83	48.85%	0	75.50
27	62	52.59%	194	39.99	59	49.71%	0	40.56	75	54.28%	0	82.95
28	78	49.51%	419	68.94	73	47.49%	0	73.03	88	53.65%	0	108.62
29	72	58.51%	279	57.66	65	54.26%	0	47.45	75	56.19%	0	78.03
30	82	53.92%	208	120.62	82	53.92%	0	105.22	97	59.21%	0	164.50
31	69	40.10%	158	79.80	69	40.10%	0	74.87	86	49.41%	0	104.67
32	65	43.37%	181	45.03	61	41.76%	0	54.17	85	49.01%	0	109.63
33	65	50.24%	114	68.88	65	50.24%	0	61.34	77	55.36%	0	89.06
34	63	58.03%	169	49.55	62	56.78%	0	39.75	80	57.90%	0	61.68
35	68	42.26%	122	128	68	42.26%	0	80.12	87	49.66%	0	130.89
36	64	56.88%	161	82.47	65	58.48%	0	51.49	75	60.98%	0	84.11
37	55	55.31%	133	40.31	55	55.31%	0	33.55	60	57.31%	0	48.61
38	58	47.21%	118	59.52	58	47.21%	0	43.00	73	52.59%	0	69.25
39	86	50.09%	317	139.75	86	50.09%	0	105.11	102	55.88%	0	157.34
40	64	51.70%	174	49.8	63	51.13%	0	49.43	73	53.88%	0	78.90
41	72	50.40%	159	68.92	72	50.40%	0	55.83	81	54.13%	0	79.52
42	58	52.02%	96	47.44	58	52.02%	0	45.26	66	57.71%	0	66.31
43	54	55.47%	116	41.22	55	56.24%	0	30.94	63	59.87%	0	49.44
44	67	45.40%	277	45.41	62	41.70%	0	38.68	78	45.84%	0	80.00
45	76	51.25%	349	75.56	68	46.30%	0	52.82	81	51.36%	0	75.24
...	-	-	–	-	-	-	-	-	-	-	-	-
100	70	53.57%	217	83.84	70	53.57%	0	62.47	78	57.08%	0	79.47
Mean	65.45	49.90%	182.25	63.76	63.83	48.78%	0	51.98	75.83	53.18%	0	85.68

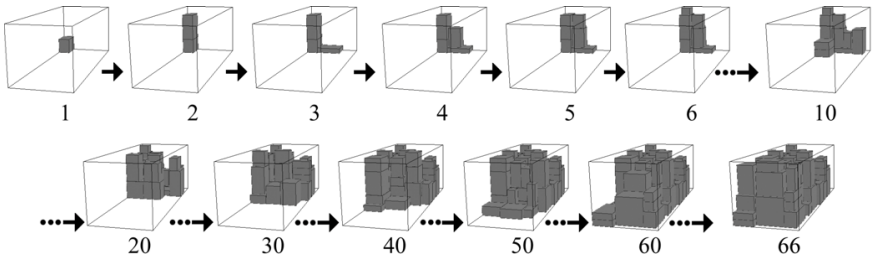


Fig. 5. Continuous process of loading from the first item to the final (66th) item by using our approach for problem No. 7 of the *thpack7* benchmark problem set.

of items to be given in our problem setting, but the benchmark data set does not give the information. Therefore, we produce a loading order for each problem in the benchmark problem set by applying the seed number given by the problem as the seed number of pseudo-random generator, and use the technique of *linear probing* (Cormen et al., 2002) to handle the collision of random numbers.

Note that both the SBIP and our algorithms are stochastic, and hence the experimental results are not deterministic. That is to say, the experimental results obtained by executing those stochastic algorithms once at different time are widely different such that we cannot give any final conclusion of their performance and quality. Fortunately, it runs very fast to execute the SBIP and our algorithms once. Consequently, what we apply to assessing the performance and quality is the experimental results with the best utilization ratios among those obtained by executing a number of times of the algorithms. By doing this, a definite conclusion can be made. After a large number of trials, we observe that it takes about two minutes on average to execute 100,000 times of our algorithms on a problem, and such running time is feasible in practice. In fact, the conclusions of executing 100,000 times and 1,000,000 times of algorithms are almost the same, so here we only give the former conclusion. Therefore, in Table 1, the running time and the final packing patterns of the SBIP and our algorithms are measured by executing 100,000 times of the algorithms. Note that the number of times of algorithms can be adjusted according to practical applications.

The experimental results are analyzed as follows. As for the running time, we observe from Table 1 that executing once of the SBIP algorithm (resp., Algorithm 3.1; Algorithm 3.2) takes about 637.6 (resp., 519.8; 856.8) microseconds on average. It implies that the running time of each of our algorithms does not differ a lot from that of the SBIP.

As for the utilization ratio, although the SBIP algorithm has higher average utilization ratio than Algorithm 3.1, Algorithm 3.1 sometimes has higher utilization ratio (see Figure 6(a)), and the gaps between the two algorithms on the number of loaded items and the utilization ratio are very small (averagely, 1.62 items and 1.12%, respectively). However, a more important feature of our algorithms over the SBIP algorithm is that our algorithms lead to zero unloading cost; instead, the SBIP algorithm leads to huge unloading costs, i.e., from Table 1, the delivery person needs to unload and reload unconcerned items 182.25 times on average in a consignment. It is concluded that for the *thpack7* benchmark problem set, the consignment using Algorithm 3.1 only sacrifices the loading of 1.62 cargos (in comparison to the SBIP approach) for avoiding the delivery person from unloading and reloading unconcerned cargos 182.25 times on average!

If a 'non-strict' unloading ordering is allowed, Algorithm 3.2, provides a packing pattern with not only higher average utilization ratio (53.18%) than the SBIP but also zero unloading cost. Note that SBIP sometimes has higher utilization ratio (see Figure 6(b)). Therefore, it is suggested to apply Algorithm 3.2 if a 'non-strict' unloading ordering is allowed.

One may wonder whether the SBIP that takes the zero unloading cost constraint into account has the ability to perform better than our algorithms. Let SBIP' denote the SBIP which considers the zero unloading cost constraint. The performance of the SBIP' is investigated as follows. The comparison of the experimental results for the *thpack7* problem set using different algorithms is given in Table 2, in which each entry is the mean performance measure evaluated according to the 100 problems in the *thpack7* problem set. From Table 2, the unloading cost of each algorithm except for the SBIP is zero, since only the SBIP does not consider the zero unloading cost constraint. As for the utilization ratio, the SBIP' performs the worst among all algorithms, and the gap is large. The main reason is that each iteration of the SBIP' selects a random subvolume to accommodate the current item, rather that in the lexicographic order. That is, a random order of packing items performs worse than the lexicographic order, at least for executing 10^5 times of the algorithms. As for the running time, since the SBIP' packs much fewer items, it spend the least time.

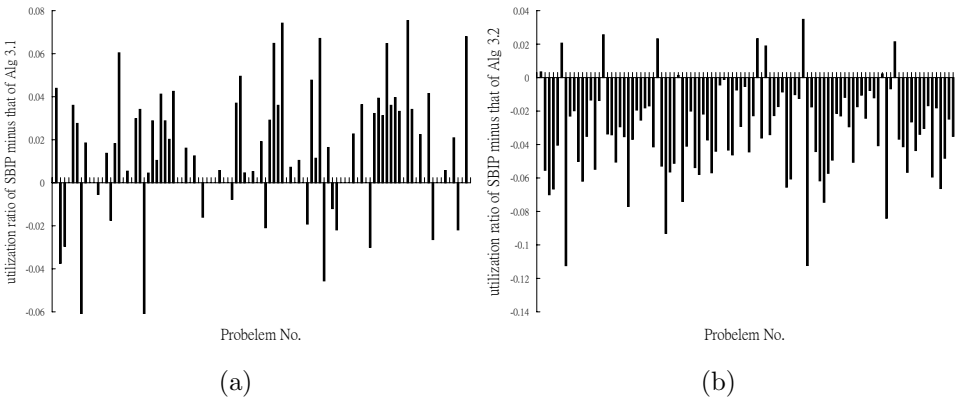


Fig. 6. With respect to every problem in the *thpack7* benchmark problem set, bar chart of the utilization ratio difference between the packing patterns by using 100,000 times of (a) the SBIP and Algorithm 3.1; (b) the SBIP and Algorithm 3.2, respectively.

Table 2. Comparison of the experimental results using different algorithms for the *thpack7* problem set, in which each entry is the mean value measured according to the 100 problems in the *thpack7* problem set.

method	items loaded	utilization rate	unloading cost	running time (s)
10^5 times of SBIP	65.45	49.90%	182.25	63.76
10^5 times of SBIP'	44.38	34.17%	0	15.81
10^5 times of Algorithm 3.1	63.83	48.78%	0	51.98
10^5 times of Algorithm 3.2	75.83	53.18%	0	85.68

Note: The SBIP' denotes the SBIP that considers the zero unloading cost constraint.

In what follows, we consider to relax the constraint of zero unloading cost. Let λ denote the unloading cost. It would be impossible to tailor our algorithm to generate a packing pattern with an arbitrary fixed λ value, because the λ value cannot be determined before the final packing pattern is generated. Therefore, we consider the algorithm $\mathcal{A}$ which executes 10^5 times of Algorithm 3.2 without the zero unloading cost constraint and maintains the best λ value so far. Then we try to find the most suitable λ value for a certain problem instance from its experimental analysis.

In our experiment, the 5th problem in the *thpack7* benchmark problem set, whose packing pattern generated by Algorithm 3.2 has the lowest utilization ratio (see Table 1), is selected as the input representative problem instance because its utilization ratio may be improved more easily. We run 100 times of algorithm $\mathcal{A}$ on this problem, record the best utilization ratio for the occurrence of each λ value, and plot them in Figure 7. Note that the maximal λ value (eleven) in Figure 7 is not a large number, which implies that the packing pattern generated by algorithm $\mathcal{A}$ does not induce too much unloading cost, even if no considering the constraint of zero unloading cost. That is, the arrangement of items in algorithm $\mathcal{A}$ has the capacity of reducing the unloading cost, but does not guarantee the satisfaction of the zero unloading cost constraint.

Notice that the algorithm $\mathcal{A}$ based on Algorithm 3.2 is stochastic, so the experimental result is not determined, as shown in Figure 7, in which, especially, the packing patterns with more than 11 units of unloading cost cannot be found in this experiment. But at least according to this experiment, if 5 units of unloading cost are allowed for the representative problem, then the utilization ratio is the highest.

5. Conclusion

By precisely defining the unloading costs according to the *invisible and untouchable rule*, an iterative subvolume-based heuristic algorithm has been proposed for

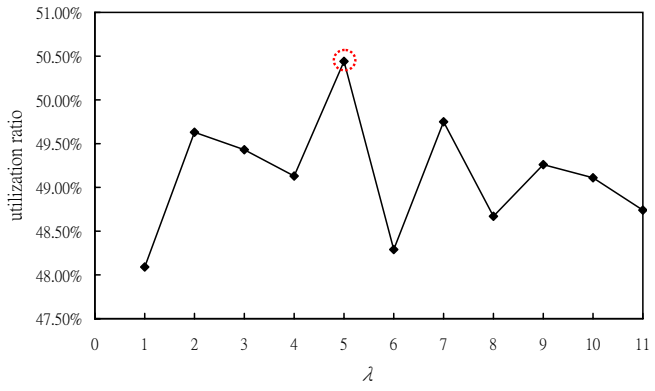


Fig. 7. Plots of utilization ratios versus unloading costs (λ).

coping with the three-dimensional container packing problem with zero unloading cost constraint, which arises from the motivations of home delivery service, and also can be viewed as an extreme of multi-drop situations. In comparison to the approaches used in multi-drop situations, the experimental results reveal that the packing patterns produced by our approach shares the merits of not only zero unloading cost but also high utilization ratios, and the benchmark problems can be executed efficiently.

A line of future work is to develop a robust heuristic algorithm for more practical situations in home delivery service, e.g., each customer may order not only an item, i.e., multi-drops situations with zero unloading costs. In addition, it should be noticed that the space of the up subvolumes is usually wasted in order to avoid producing unloading costs (see also Figures 4 and 5), and the space of the two subvolumes that are not on the same horizontal plane cannot be combined into a larger subvolume to accommodate a larger item in the subvolume scheme. Therefore, if empty boxes and foam rubber can be applied to filling out some of the wasted space so that the other wasted space can be utilized, then it is of interest to adjust the heuristic algorithm to yield better performance.

Acknowledgments

The authors thank the anonymous referees for comments that improved the content as well as the presentation of this paper. The first two authors (Wan-Yu Liu and Chun-Cheng Lin) were supported in part by National Science Council, Taiwan (NSC Grant 98-2218-E-151-004-MY3).

References

Abdou, G and M Elmasry. (1999). 3D random stacking of weakly heterogeneous palletization problems. *International Journal of Production Research*, 37, 1505–1524.

Abdou, G and M Yang. (1994). A systematic approach for the three dimensional palletization problems. *International Journal of Production Research*, 32, 2381–2394.

Beasley, J. (1985). An exact two-dimensional non-cutting tree search procedure. *Operations Research*, 33, 49–64.

Beasley, J. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41, 1069–1072.

Bischoff, E and M Ratcliff. (1995). Issues in the development of approaches to container loading. *Omega–International Journal of Management Sciences*, 23, 377–390.

Bortfeldt, A, H Gehring, and D Mack. (2003). A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29, 641–662.

Cormen, T, C Leiserson, R Rivest, and C Stein. (2002). *Introduction to algorithms* (2nd ed.). The MIT Press.

Fanslau, T and A Bortfeldt. (2009). A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing*. Available online, in press.

Garey, M and D Johnson. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco.

Gehring, H and A Bortfeldt. (1997). A genetic algorithm for solving the container loading problem. *International Transactions on Operational Research*, 4, 401–418.

George, J and D Robinson. (1980). A heuristic for packing boxes into a container. *Computers and Operations Research*, 7, 147–156.

Gilmore, P and R Gomory. (1965). Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13, 94–120.

Jin, Z, K Ohno, and J Du. (2004). An efficient approach for the three-dimensional container packing problem with practical constraints. *Asia–Pacific Journal of Operational Research*, 21, 279–295.

Kämäräinen, V and M Punakivi. (2002). Developing cost-effective operations for the e-grocery supply chain. *International Journal of Logistics: Research and Applications*, 5, 285–298.

Kämäräinen, V, S Saranen, and H Jan. (2000). How goods receipt affects e-grocery efficiently? In *Proc. of 11th international working seminar on production economics.*

Lim, A, B Rodrigues, and Y Wang. (2003). A multi-faced buildup algorithm for three-dimensional packing problems. *Omega – International Journal of Management Sciences*, 31, 471–481.

Mack, D, A Bortfeldt, and H Gehring. (2004). A parallel hybrid local search algorithm for the container loading problem. *International Transactions in Operational Research*, 11, 511–533.

Morabito, R and M Arenales. (1994). An AND/OR-graph approach to the container loading problem. *International Transactions in Operational Research*, 1, 59–74.

Moura, A and J. F Oliveira. (2005). A grasp approach to the container-loading problem. *IEEE Intelligent Systems*, 20, 50–57.

OR-Library. (2007). *OR-library.*
http://people.brunel.ac.uk/∼mastjjb/jeb/info.html

Parreno, F, R Alvarez-Valdes, J Tamarit, and J Oliveira. (2008). A maximal-space algorithm for the container loading problem. *INFORMS Journal on Computing*, 20, 412–422.

Punakivi, M. (2003). *Comparing alternative home delivery models for e-grocery home delivery.* Unpublished doctoral dissertation, Helsinki University of Technology, Finland.

Punakivi, M and J Saranen. (2001). Identifying the success factors in e-grocery home delivery. *International Journal of Retail and Distribution Management*, 29, 156–163.

The Economist. (2000). Survey: E-commerce - shopping around the web. *The Economist*, 354, s5–s6.

Wäscher, G, H Haußner, and H Schumann.(2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183, 1109–1130.

Wan-Yu Liu received her B.B.A. degree in Finance from National Chengchi University, Taipei, Taiwan, in 2002; M.S. degree and Ph.D. degree both in Agricultural Economics from National Taiwan University, Taipei, Taiwan, in 2004 and 2008, respectively. Since August 2008, she has been an assistant professor of the Department of Applied Natural Resources at Aletheia University, in Tainan, Taiwan. Her main research interests include environmental and resource economics analysis, applied econometrics, evaluation of GHG reduction policies, as well as forest economics.

Chun-Cheng Lin received his B.S. degree in Mathematics, M.B.A. degree in Business Administration, and Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 2000, 2007, and 2009, respectively. Since February 2011, he has been an Assistant Professor of the Department of Industrial Engineering and Management at National Chiao Tung University, in Hsinchu. He was an Assistant Professor of the Department of Computer Science at Taipei Municipal University of Education from February 2010 to January 2011, and an Assistant Professor of the Department of Computer Science and Information Engineering at National Kaohsiung University of Applied Sciences from February 2009 to January 2010. His main research interests include graph drawing and information visualization, computer graphics and multimedia, communications and networking, design and analysis of algorithms, as well as computational management science.

Chang-Sung Yu received his B.S. degree in Mathematics and Ph.D. degree in Industrial Management from National Taiwan University and Carnegie-Mellon University, respectively. Since 1987, he has been a professor of the Department of Business Administration, National Taiwan University, Taipei, Taiwan. From 1991 to 1994, he was the chairman and professor of the Department of Information Management, National Taiwan University. His main research interests include information management, business data communication, information technology management, as well as technology innovations and marketing.