Computers & Industrial Engineering 97 (2016) 212–221

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

# An effective placement method for the single container loading problem

Yao-Huei Huang [a], F.J. Hwang [b], Hao-Chun Lu [c,*]

[a] *School of Economics and Management, Southwest Jiaotong University, Chengdu 610031, China*
[b] *School of Mathematical and Physical Sciences, University of Technology Sydney, Ultimo 2007, Australia*
[c] *Department of Information Management, Fu Jen Catholic University, New Taipei City 24205, Taiwan*

ARTICLE INFO

*Article history:*
Received 9 August 2015
Received in revised form 14 March 2016
Accepted 2 May 2016
Available online 4 May 2016

*Keywords:*
Container loading problem
Mixed integer linear program
Heuristic
Loading placement

ABSTRACT

This study investigates a three-dimensional single container loading problem, which aims to pack a given set of unequal-size rectangular boxes into a single container such that the length of the occupied space in the container is minimized. Motivated by the practical logistics instances in literature, the problem under study is formulated as a zero-one mixed integer linear programming model. Due to the NP-hardness of the studied problem, a simple but effective loading placement heuristic is proposed for solving large-size instances. The experimental results demonstrate that the developed heuristic is capable of solving the instances with more than two hundred boxes and more efficient than the state-of-the-art mixed integer linear program and existing heuristic methods.

## 1. Introduction

The container loading problem (CLP), which is also referred to as the packing problem, plays a crucial role in logistics planning and scheduling, and widely arises in various real-world applications, such as electronic, steel, paper, textiles, manufacturing and transportation industries. In this paper, we consider the three-dimensional container loading problem (3DCLP), where a given set of nonuniform-size rectangular boxes is to be packed within an enveloping rectangular container such that the length of the occupied space in the container is minimized. All the boxes are assumed to be orthogonally positioned, i.e. each edge of the box is parallel to one axis of the container. The objective function addressed in this study is different from the criterion of minimizing the volume of the occupied space in that an occupied space with the minimum length unnecessarily retains the minimum volume. For example, a container with a length of 6-m, a width of 3-m, and a height of 2-m is given for packing six cube boxes with a 1-m side length and one rectangular box with a 1-m length, 1-m width, and 0.5-m height. If minimizing the volume of the occupied space is considered, an optimal solution with the length equaling 3.5 m and the volume equaling $(3.5 \times 2 \times 1) = 7\ \text{m}^3$ is shown in Fig. 1(a). If the criterion is the minimization of the length, an optimal solution with the length equaling 1.5 m and the volume equaling $(1.5 \times 3 \times 2) = 9\ \text{m}^3$ is depicted in Fig. 1(b). Motivated by the practical instances in logistics (cf. Chen, Lee, & Shen, 1995; Hu, Li, & Tsai, 2012), the studied problem could arise from the container loading for a truck, which delivers goods or parcels of various types to several different shops or transshipment points along a route. To process the unloading efficiently, the container is divided into several sections, each of which is stacked with the goods ordered by an individual shop or the parcels to be transferred to a transshipment station. As shown in Fig. 2, minimizing the length of the section required to pack goods for each shop yields the maximal number of shops or transshipment points to which a truck can make deliveries with a single trip, and thus reduces inventory and transportation costs. For each shop or transshipment station, i.e. each section in the container, the 3DCLP under study is considered. Furthermore, the generalized assumption that the sizes of rectangular boxes are various and even customer-defined is not uncommon and actually practical in Taiwan's logistics industries, such as Hsin-Chu Transportation Logistics, Kerry TJ Logistics, Taiwan Pelican Express, T-cat Takkyubin, and Taiwan Post (cf. Chou & Lu, 2009).

In the past decades, the solution approaches proposed to cope with variants of CLPs can be divided into two classes: heuristic algorithms and deterministic methods. Owing to the great complexity of the CLPs in nature, most research has focused on the development of heuristic methods. Considering the two-dimensional pallet loading problem, Hodgson (1982) developed an integrated approach of dynamic programming and heuristic procedure to minimize the number of pallets for loading a given set of boxes. George and Robinson (1980) investigated a problem of packing a given set of boxes into a container with a fixed volume

* Corresponding author.
*E-mail addresses:* yaohuei.huang@gmail.com (Y.-H. Huang), feng-jang.hwang@uts.edu.au (F.J. Hwang), haoclu@gmail.com (H.-C. Lu).

http://dx.doi.org/10.1016/j.cie.2016.05.008
0360-8352/© 2016 Elsevier Ltd. All rights reserved.

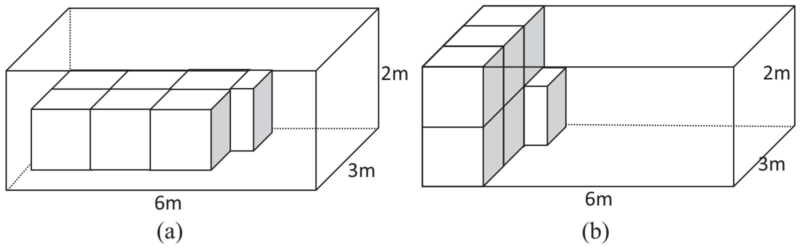

**Fig. 1.** Optimal solutions for the volume minimization (a) and the length minimization (b).

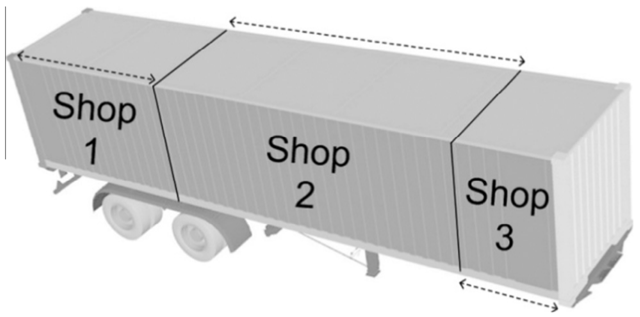

**Fig. 2.** Three container sections for three individual shops.

and proposed a heuristic for the sequencing and positioning such that all the boxes can be fitted in. Bischoff and Marriott (1990) presented a comparative analysis of fourteen heuristic rules for the CLP and examined the impact of the number of different item types in a load on the loading efficiency. Considering the weight and the load bearing abilities of the boxes in the CLP, Ratcliff and Bischoff (1998) proposed an approach to adapt the solution procedures in the literature to deal with fragility consideration. Faina (2000) developed a geometrical model that reduces the CLP to a finite enumeration scheme, and presented a statistical optimization algorithm based on simulated annealing. Eley (2002) designed a greedy heuristic coupled with a tree search procedure for the single-container and multiple-container packing problems. Pisinger (2002) presented a heuristic algorithm to decompose the CLP into a number of layers, which are divided into a number of strips. Then the decomposed sub-problem of packing strips can be formulated as the well-known knapsack problem. Investigating the arbitrary *m*-dimensional CLP, Lins, Lins, and Morabito (2002) proposed a recursive uniform algorithm to partition the container into various sub-containers, each of which can be recursively partitioned into further sub-containers. A parallel tabu search algorithm based on the multi-search threads for the CLP was presented by Bortfeldt, Gehring, and Mack (2003). Then Hifi (2004) developed an integrated algorithm of dynamic programming and graph search procedure with depth-first search strategy for solving the weighted three-dimensional cutting problem, which is a variant of CLP. Considering the three-dimensional strip packing problem, Bortfeldt and Mack (2007) designed a heuristic algorithm which was derived from the layer-building approach proposed by Pisinger (2002). Almeida and Figueiredo (2010) presented an alternative non-linear formulation containing additional restrictions on the placement and designed a heuristic algorithm for the CLP. Fanslau and Bortfeldt (2010) presented a tree search algorithm for the 3DCLP, where a special form of tree search keeps the search effort low and ensures a suitable balance between search diversity and foresight during the search. Zhang, Peng, and Leung (2012) proposed a block-loading algorithm based on multi-layer search for the 3DCLP with depth-first and multi-layer search for determining the selected bock in each packing phase. The computational results in Zhang et al. (2012) showed that their developed algorithm outperforms the method presented by Fanslau and Bortfeldt (2010). Zhu, Huang, and Lim (2012) addressed the cost minimization in the multiple-container packing problem formulated as a set cover problem, and presented a prototype column generation scheme.

In the relevant literature on deterministic methods, the CLPs are generally modeled as a mathematical program. Chen et al. (1995) considered the 3DCLP, which aims to select a number of containers to pack a given set of boxes. They proposed a mixed integer linear programming (MILP) formulation and solved the small-size problems by the LINGO package. Then Tsai and Li (2006) first adapted the MILP model proposed by Chen et al. (1995) to present a mixed integer nonlinear programming (MINLP) model for the 3DCLP which aims to pack a given set of boxes into a container with the minimum volume. They modified the MINLP by reducing the number of binary variables and utilizing a piecewise linearization technique to find a global optimum within a tolerable error. Tsai, Wang, and Lin (2014), subsequently, converted the original MINLP into an MILP with an improved piecewise linearization technique using fewer extra zero-one variables and constraints to enhance the computational efficiency. On the other hand, Hu et al. (2012) developed a transformation method to convert the nonlinear objective function in the 3DCLP into an increasing function with single variable and two fixed parameters. A transformed MILP was then decomposed into several sub-problems, which were solved in parallel by a proposed distributed genetic algorithm.

The NP-hardness of the studied 3DCLP, which is a generalization of the bin-packing problem, indicates that it is very unlikely to devise a polynomial-time exact algorithm for yielding an optimal solution to the 3DCLP unless P = NP. To cope with the medium- or large-scale 3DCLP, developing an effective heuristic is relatively applicable. We observe that most existing heuristics for the 3DCLP are based on the wall-building approach, which suffers from the following deficiencies in efficiency and effectiveness: (*i*) A backtracking probability is necessary for solution evolution, which could incur a considerable computation time; (*ii*) The generated solution would have numerous unutilized separated spaces and thus a low container space utilization. In this paper, we proposed a simple but effective loading placement heuristic for the studied 3DCLP. By comparison with the state-of-the-art deterministic MILP model and existing heuristic techniques, the developed heuristic algorithm is capable of generating quality solutions to the benchmark dataset efficiently.

The rest of the paper is organized as follows. In Section 2, a reference MILP of the 3DCLP is introduced. A loading placement heuristic algorithm is proposed in Section 3. Numerical experiments are conducted in Section 4. Finally, some concluding remarks are made in Section 5.

## 2. Reference MILP model

Assume that $n$ nonuniform-size rectangular boxes, each of which has a specific length, width and height, are given. Denote the length, width, and height of the space required to pack all the $n$ boxes into a container by $x$, $y$ and $z$, respectively. Referring to Chen et al. (1995), Tsai and Li (2006), and Tsai et al. (2014), the 3DCLP model studied in this paper can be stated as follows:

Min $x$

s.t.

(i) All of the boxes are positioned without overlap;
(ii) All of the boxes are loaded within the range of $x$, $y$, and $z$;
(iii) The dimensions of the occupied space abide by the given upper bounds, i.e. the dimensions of the container.

The notation used in the 3DCLP model (cf. Chen et al., 1995; Tsai & Li, 2006; Tsai et al., 2014) is described below:

| *Notation* | |
|---|---|
| $n$ | Total number of the given set of boxes to be loaded. |
| $N$ | $N = \{1, 2, \ldots, n\}$. |
| $M$ | $M = \max\{\bar{x}, \bar{y}, \bar{z}\}$, where $\bar{x}$, $\bar{y}$, and $\bar{z}$ are the upper bounds of $x$, $y$, and $z$, i.e. the length, width, and height of the container, respectively. |
| $(p_i, q_i, r_i)$ | Parameters indicating the length, width and height of box $i$, respectively. |
| $(x_i, y_i, z_i)$ | Variables indicating the coordinates of the left-front-bottom corner of box $i$. |
| $(l_{xi}, l_{yi}, l_{zi})$ | Binary variables indicating whether the length of box $i$ is parallel to the $x$-axis, $y$-axis or $z$-axis. For example, $l_{xi} = 1$ if the length of box $i$ is parallel to the $x$-axis; otherwise, $l_{xi} = 0$. |
| $(w_{xi}, w_{yi}, w_{zi})$ | Binary variables indicating whether the width of box $i$ is parallel to the $x$-axis, $y$-axis or $z$-axis. For example, $w_{xi} = 1$ if the width of box $i$ is parallel to the $x$-axis; otherwise, $w_{xi} = 0$. |
| $(h_{xi}, h_{yi}, h_{zi})$ | Binary variables indicating whether the height of box $i$ is parallel to the $x$-axis, $y$-axis or $z$-axis. For example, $h_{xi} = 1$ if the height of box $i$ is parallel to the $x$-axis; otherwise, $h_{xi} = 0$. |
| $(\alpha_{ij}, \beta_{ij}, \delta_{ij})$ | Binary variables indicating the relative positions of box $i$ and box $j$, such as (a) $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (0,0,1)$ if box $i$ is on the left-hand side of box $j$; (b) $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (0,1,0)$ if box $i$ is on the right-hand side of box $j$; (c) $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (1,0,0)$ if box $i$ is behind box $j$; (d) $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (0,1,1)$ if box $i$ is in front of box $j$; (e) $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (1,0,1)$ if box $i$ is below box $j$; (f) $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (1,1,0)$ if box $i$ is above box $j$. |

By simplifying the Model 2 in Tsai et al. (2014), we adopt the following MILP as the reference model.

*Reference MILP Model*

$$\text{Min } x \tag{1}$$

s.t.

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leqslant x_j + M(1 + \alpha_{ij} + \beta_{ij} - \delta_{ij}), \quad \forall i, j \in N, \ i < j, \tag{2}$$

$$x_j + p_j l_{xj} + q_j w_{xj} + r_j h_{xj} \leqslant x_i + M(1 + \alpha_{ij} - \beta_{ij} + \delta_{ij}), \quad \forall i, j \in N, \ i < j, \tag{3}$$

$$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \leqslant y_j + M(1 - \alpha_{ij} + \beta_{ij} + \delta_{ij}), \quad \forall i, j \in N, \ i < j, \tag{4}$$

$$y_j + p_j l_{yj} + q_j w_{yj} + r_j h_{yj} \leqslant y_i + M(2 + \alpha_{ij} - \beta_{ij} - \delta_{ij}), \quad \forall i, j \in N, \ i < j, \tag{5}$$

$$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{zi} \leqslant z_j + M(2 - \alpha_{ij} + \beta_{ij} - \delta_{ij}), \quad \forall i, j \in N, \ i < j, \tag{6}$$

$$z_j + p_j l_{zj} + q_j w_{zj} + r_j h_{zj} \leqslant z_i + M(2 - \alpha_{ij} - \beta_{ij} + \delta_{ij}), \quad \forall i, j \in N, \ i < j, \tag{7}$$

$$1 \leqslant \alpha_{ij} + \beta_{ij} + \delta_{ij} \leqslant 2, \quad \forall i, j \in N, \ i < j, \tag{8}$$

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leqslant x, \quad \forall i \in N, \tag{9}$$

$$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \leqslant y, \quad \forall i \in N, \tag{10}$$

$$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{zi} \leqslant z, \quad \forall i \in N, \tag{11}$$

$$l_{xi} + l_{yi} + l_{zi} = 1, \quad \forall i \in N, \tag{12}$$

$$w_{xi} + w_{yi} + w_{zi} = 1, \quad \forall i \in N, \tag{13}$$

$$h_{xi} + h_{yi} + h_{zi} = 1, \quad \forall i \in N, \tag{14}$$

$$l_{xi} + w_{xi} + h_{xi} = 1, \quad \forall i \in N, \tag{15}$$

$$l_{yi} + w_{yi} + h_{yi} = 1, \quad \forall i \in N, \tag{16}$$

$$l_{zi} + w_{zi} + h_{zi} = 1, \quad \forall i \in N, \tag{17}$$

where $M = \max\{\bar{x}, \bar{y}, \bar{z}\}$, $x_i, y_i, z_i \geqslant 0$, $0 < x \leqslant \bar{x}$, $0 < y \leqslant \bar{y}$, $0 < z \leqslant \bar{z}$, and $l_{xi}$, $l_{yi}$, $l_{zi}$, $w_{xi}$, $w_{yi}$, $w_{zi}$, $h_{xi}$, $h_{yi}$ and $h_{zi}$ are 0–1 variables.

The objective function (1) of the reference MILP is to minimize the length of the required space in the container for packing all boxes. Inequalities (2)–(8) are the constraints used to ensure that all these boxes are positioned without overlap. Constraints (9)–(11) make all boxes be packed within the required space in the container. Constraints (14) ensure that the length (width, height) of box $i$ is parallel to one of $x$-axis, $y$-axis and $z$-axis for all $i \in N$. Constraints (17) enforce that only one of the length, width and height of box $i$ is parallel to $x$-axis ($y$-axis, $z$-axis) for all $i \in N$. To the best of our knowledge, the above reference MILP model (1–17), is the state-of-the-art mathematical programming model for the studied 3DCLP.

## 3. Proposed loading placement heuristic

In the development of the loading placement heuristic, we generate loading patterns through an iterative process and propose a placement procedure to load all these boxes. The notions of the loading pattern representation, iteration, and evaluation stem from the fundamentals of evolutionary computation, such as genetic algorithm (GA) (Holland, 1975). The iterative loading placement procedure is comprised of the following seven components.

(i) Coordinate expression is used to record the relative position of boxes in the container.
(ii) Non-overlapping and non-overstepping conditions are utilized to avoid any overlap between boxes.
(iii) Use a loading pattern to present a sequence for packing all boxes and the corresponding data structure.
(iv) Evaluation function measures the solution qualities of loading patterns.
(v) Load all boxes into the container by the loading process.
(vi) Generate different loading patterns using two specific operators.
(vii) A history bucket aims to avoid generating the same loading pattern.

### 3.1. Coordinate expression

As the definition in the reference MILP model, the position of any box $i$ in the container is recorded with its coordinates of the left-front-bottom corner. Take Fig. 3 for an example. Parameter vector (0,0,0) indicates the coordinates of the left-front-bottom corner of box $i$ in the container.

### 3.2. Non-overlapping and non-overstepping conditions

Referring to Constraints (2)–(8), the proposed heuristic adopts the following six conditions to check whether any pair of boxes $i$ and $j$, $i,\ j \in N$, $i \neq j$ overlap:

$$x_i \leqslant x_j - p_i; \quad x_j \leqslant x_i - p_j; \quad y_i \leqslant y_j - q_i; \quad y_j \leqslant y_i - q_j;$$
$$z_i \leqslant z_j - r_i; \quad z_j \leqslant z_i - r_j \tag{18}$$

If and only if all the conditions in Eq. (18) are violated, the box $i$ and box $j$ overlap. The non-overstepping conditions ensuring that each box $i \in N$ is positioned inside the container are

$$x_i + p_i \leqslant \bar{x}; \quad y_i + q_i \leqslant \bar{y}; \quad z_i + r_i \leqslant \bar{z}. \tag{19}$$

### 3.3. Loading pattern

Assume that at each iteration $r$ of the heuristic the boxes are loaded into the container in accordance with a packing sequence $([1],[2],\ldots,[n])$, the loading pattern of which can be represented by an array

$$B_r = (b_{[1]}, b_{[2]}, \ldots, b_{[n]}) \tag{20}$$

where

$$b_{[h]} = (p_{[h]}, q_{[h]}, r_{[h]}, x_{[h]}, y_{[h]}, z_{[h]}), \quad \forall h \in N. \tag{21}$$

Each element $b_{[h]}$ in the loading pattern $B_r$ corresponds to the box which is arranged in the $h$-th position in the packing sequence, and denotes the parameter vector of box $[h]$.

### 3.4. Evaluation function

The evaluation function shown in Eq. (22) is used to measure the solution quality of the loading pattern $B_r$:

$$f_r = \frac{\max_{i \in N} \{x_i + p_i\}}{\bar{x}}. \tag{22}$$

The loading pattern $B_r$ is feasible if $0 < f_r \leqslant 1$. Otherwise, $B_r$ is infeasible.

### 3.5. Loading process

For each iteration $r$ of the heuristic procedure, the loading process is performed for all boxes in $N$ according to the sequence ([1], [2],…,[$n$]). Denote the set of loaded boxes by $N_l \subseteq N$. The loading process is initialized by positioning box [1] in the left-front-bottom corner $(0,0,0)$ (cf. Fig. 3 for $i$ = [1]) and setting $N_l = \{[1]\}$ and $h$ = 2. Then the loading process is run with the following steps.

Step 1: Load box $i = [h]$ by attaching the left-front-bottom corner of box $i$ to the right-front-bottom corner of some loaded box $k$ such that $x_k + p_k$ is the smallest among all loaded boxes and box $i$ is non-overlapping with any loaded box and does not exceed the dimension of the container (cf. Fig. 4).

(1a) Define $x_i := x_{i'} + p_{i'}$, $y_i := y_{i'}$ and $z_i := z_{i'}$.
(1b) Find $k_1 = \arg\min_{i' \in N_l} \{x_i | \text{Eq.}(18)\ \forall j \in N_l \setminus \{i'\};\ \text{Eq.}(19)\}$.
(1c) If $k_1 \neq null$, set $x_i = x_{k_1} + p_{k_1}$, $y_i = y_{k_1}$ and $z_i = z_{k_1}$.

Step 2: Move box $i$ by attaching the left-front-bottom corner of box $i$ to the left-behind-bottom corner of some loaded box $k$ such that $y_k + q_k$ is the smallest among all loaded boxes and box $i$ is non-overlapping with any loaded box and does not exceed the dimension of the container.

(2a) Define $x_i := x_{i'}$, $y_i := y_{i'} + q_{i'}$ and $z_i := z_{i'}$.
(2b) Find $k_2 = \arg\min_{i' \in N_l} \{y_i | \text{Eq.}(18)\ \forall j \in N_l \setminus \{i'\};\ \text{Eq.}(19)\}$.
(2c) If $k_2 \neq null$, set $x_i = x_{k_2}$, $y_i = y_{k_2} + q_{k_2}$ and $z_i = z_{k_2}$.

Step 3: Move box $i$ by attaching the left-front-bottom corner of box $i$ to the left-front-top corner of some loaded box $k$ such that $z_k + r_k$ is the smallest among all loaded boxes and box $i$ is non-overlapping with any loaded box and does not exceed the dimension of the container.

(3a) Define $x_i := x_{i'}$, $y_i := y_{i'}$ and $z_i := z_{i'} + r_{i'}$.
(3b) Find $k_3 = \arg\min_{i' \in N_l} \{z_i | \text{Eq.}(18)\ \forall j \in N_l \setminus \{i'\};\ \text{Eq.}(19)\}$.
(3c) If $k_3 \neq null$, set $x_i = x_{k_3}$, $y_i = y_{k_3}$ and $z_i = z_{k_3} + r_{k_3}$.

Step 4: If $k_1 = k_2 = k_3 = null$, return $f_r = \infty$.

Otherwise, set $N_l = N_l \cup \{i\}$, $h = h + 1$ and check if $h \leqslant n$. If yes, go to Step 1. If no, return $f_r = \frac{\max_{i \in N}\{x_i + p_i\}}{\bar{x}}$.

### 3.6. Two specific operators

There are two specific operators, the position-swap operator and rotation operator, which perform the data alteration for a loading pattern to generate a new loading pattern. The position-swap operator is equipped with two random number generators. One is used to produce a random number $\rho$ in $(0,1]$, and the other can produce the random 2-tuple $(\kappa,\ \lambda)$ with $1 \leqslant \kappa < \lambda \leqslant n$. If $\rho > \bar{\rho} = 0.5$, then elements $b_{[\kappa]}$ and $b_{[\lambda]}$ in a loading pattern swap positions to produce a new loading pattern as shown in Fig. 5, where $\kappa = 2$ and $\lambda = 4$. The value $\bar{\rho} = 0.5$ is determined based on a hundred rounds of pre-testing.

The rotation operator is employed to rotate the boxes in a loading pattern, each of which obviously has six types of rotations. The rotation operator is also equipped with a random number generator which produces a probability $\alpha$, $0 < \alpha \leqslant 1$, How each box in a loading pattern is rotated depends on the generated probability as follows (cf. Fig. 6):

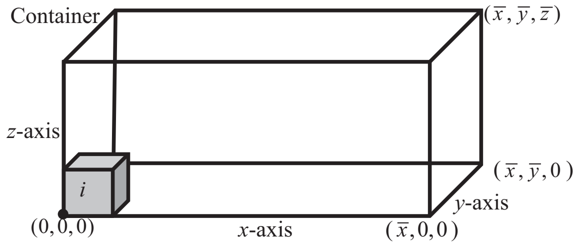

**Fig. 3.** The coordinates of the left-front-bottom corner of box $i$.

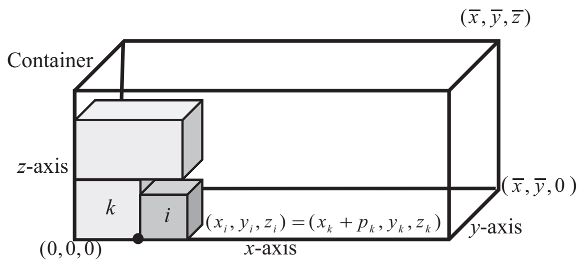

**Fig. 4.** Box $i$ adjoining box $k$.

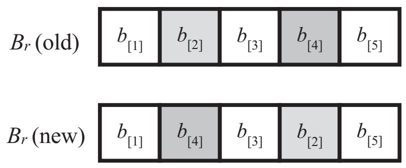
$B_r$ (old)
$b_{[1]}$ $b_{[2]}$ $b_{[3]}$ $b_{[4]}$ $b_{[5]}$
$B_r$ (new)
$b_{[1]}$ $b_{[4]}$ $b_{[3]}$ $b_{[2]}$ $b_{[5]}$

**Fig. 5.** A position swap occurring in position 2 and 4.

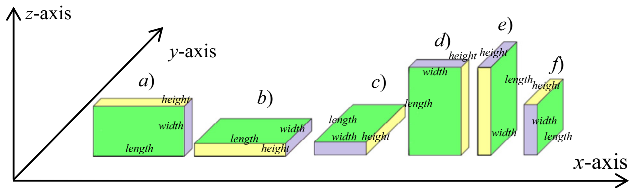

**Fig. 6.** Six types of rotations for a rectangular box.

(a) $0 < \alpha \leqslant 0.167$: The box is not rotated.
(b) $0.167 < \alpha \leqslant 0.333$: The length, width and height of box $i$ are parallel to $x$-axis, $z$-axis and $y$-axis, respectively.
(c) $0.333 < \alpha \leqslant 0.5$: The length, width and height of box $i$ are parallel to $z$-axis, $x$-axis and $y$-axis, respectively.
(d) $0.5 < \alpha \leqslant 0.667$: The length, width and height of box $i$ are parallel to $y$-axis, $x$-axis and $z$-axis, respectively.
(e) $0.667 < \alpha \leqslant 0.833$: The length, width and height of box $i$ are parallel to $y$-axis, $z$-axis and $x$-axis, respectively.
(f) $0.833 < \alpha \leqslant 1$: The length, width and height of box $i$ are parallel to $z$-axis, $y$-axis and $x$-axis, respectively.

### 3.7. History bucket

The history bucket denoted by set $H$ records the loading structure in each iteration so that the loading patterns generated in the previous iterations can be avoided. Given the current loading pattern $B_r$, the previous loading patterns can be expressed as $B_t$ for $t = 1, \ldots, r-1$. Initially we have $H = \emptyset$. In each iteration $r$, the loading pattern $B_r$ is included in $H = \{B_1, B_2, \ldots, B_{r-1}\}$.

### 3.8. Algorithm procedure

After introducing all the algorithm components, we propose the algorithm procedure as follows. First, to initiate the algorithm procedure all the given data, including the dimensions of $n$ boxes and the container, and threshold values $T_1$ and $T_2$ respectively for limiting the number of iterations and the run time are input. Subsequently, a loading pattern $B_r$ for $r = 1$ is generated in the preprocessing step. Then the algorithm performs the iterative procedure by checking whether the same loading pattern $B_r$ exists in history bucket $H$, running the loading process, updating the incumbent solution $B^*$, and then generating a new loading pattern $B_r$ for $r = r + 1$. The detail steps of the designed heuristic algorithm are expressed below:

**Initialization**

**Step 1:** Input $n$, $T_1$, $T_2$, $(\bar{x}, \bar{y}, \bar{z})$, and $(p_i, q_i, r_i)$, $\forall i \in N$.

**Preprocessing**

**Step 2:** Set $H = \emptyset$, $B^* = \emptyset$, $f^* = 1$, $r = 1$, $(x_i, y_i, z_i) = (\bar{x}, \bar{y} - q_i, \bar{z} - r_i)$, $\forall i \in N$, and $B_r = (b_{[1]}, b_{[2]}, \ldots, b_{[n]})$, $b_{[h]} = (p_{[h]}, q_{[h]}, r_{[h]}, x_{[h]}, y_{[h]}, z_{[h]})$, $\forall h \in N$.

**Iterative Procedure**

**Step 3** (Checking the history bucket):

If $B_r \notin H$, set $H = H \cup \{B_r\}$ and run the loading process to generate the evaluation value $f_r$.
Otherwise, go to **Step 5**.

**Step 4** (Updating the incumbent solution):

If $f_r \leqslant f^*$, then $B^* \leftarrow B_r$.

**Step 5** (Checking the stop condition):

If $r \leqslant T_1$ and the run time is no greater than $T_2$, then set $r = r + 1$ and $B_r \leftarrow B_{r-1}$.
Otherwise, go to the **Output** phase.

**Step 6** (Altering the loading pattern):

Run the position-swap operator and rotation operator for $B_r$, and go to **Step 3**.

**Output:** If $B^* \neq \emptyset$, the best solution $B^*$ is yielded. Otherwise, no feasible solution is found.

The flowchart of the proposed loading placement heuristic is depicted in Fig. 7.

### 3.9. Computational complexity analysis

For each iteration, the proposed algorithm procedure needs $\frac{3}{2}n(n-1)$ steps for checking all the considered locations for positioning boxes. Since the threshold of the iteration number is $T_1$, the total steps in the worst case are $\frac{3}{2}T_1 n(n-1)$. Therefore the run time of the proposed heuristic algorithm is $O(T_1 n^2)$, which is polynomial-time if $T_1$ is not a part of input, i.e. a constant.

## 4. Numerical experiments

To demonstrate the effectiveness and efficiency of proposed heuristic, we implemented it on two classes of experiments, which were conducted on the benchmark problem datasets in Tsai et al. (2014) and a real-world instance of packing different types of desktop equipment. Then a comprehensive comparison between the proposed loading placement heuristic and the following three solution techniques was presented:

(i) *Integer programming method* (*Reference MILP*): Utilizing the reference MILP model introduced in Section 2, we adopted GUROBI (2014) as the mathematical programming solver, and the instance run times solved by MILP were limited to 6000 s.
(ii) *The strip-layer placement method* (*SL*): The strip-layer placement procedure referring to Pisinger (2002) and Bortfeldt and Mack (2007) was utilized to solve the 3DCLP through an iterative process. To make the computational comparison fair, a similar evolutionary process is set for SL. We designed the random position-swap and rotation mutations as the recombination operators to produce offspring for mating in the next generation and to enlarge the diversity of new chromosomes, where the population size was set to be 50. Additionally, the probability of introducing the random position-swap and rotating mutations in the evolutionary process was set as 0.5, which is based on a hundred rounds of iterative pre-testing.
(iii) *The block-loading placement method* (*BL*): A block-loading placement procedure based on multi-layer search (Zhang et al., 2012) was utilized to solve the 3DCLP through the same evolutionary process as SL.

The heuristics SL, BL, and the proposed algorithm were implemented in Java version 6.0. All the experiments were run on a PC equipped with Intel Core i5 CPU, 8 GB RAM and Windows 7 64-bit operating system.

### 4.1. Experiment 1

In Experiment 1, we used the benchmark dataset in Tsai et al. (2014) as small-scale instances with $n < 10$ (cf. Tables 1–3) to

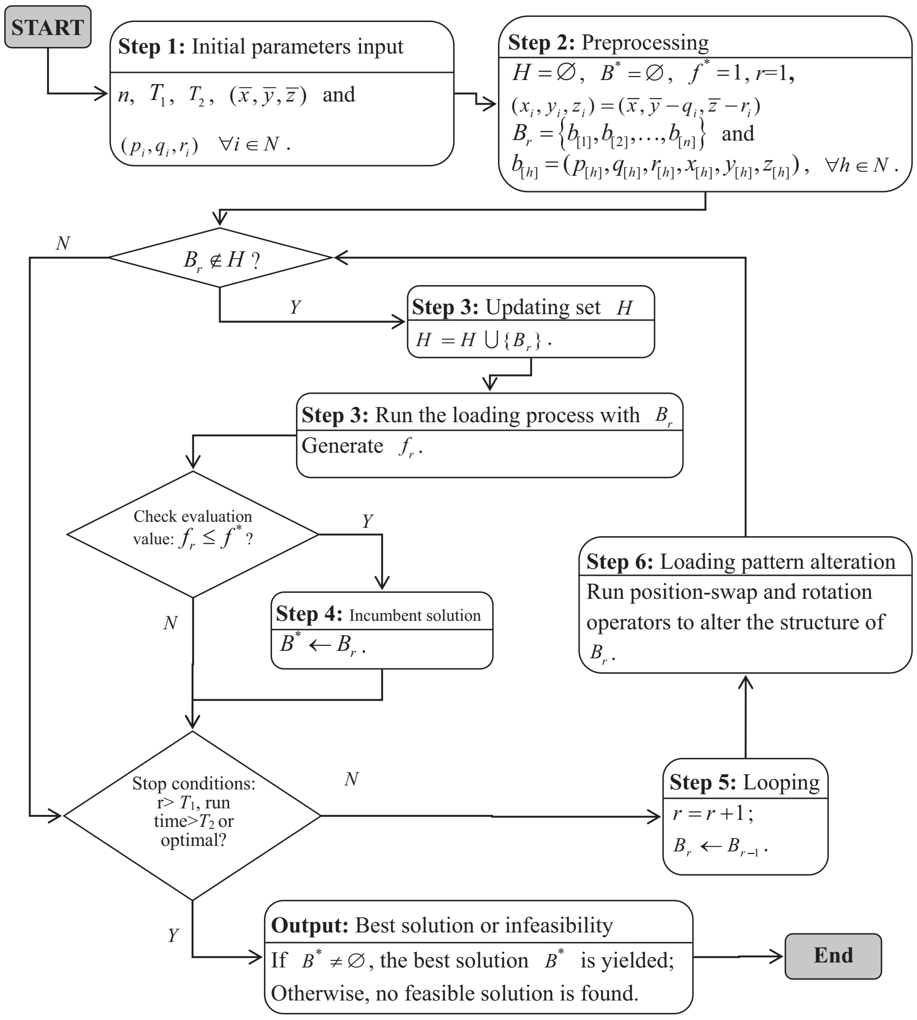
START
Step 1: Initial parameters input
$n,\ T_1,\ T_2,\ (\bar{x},\bar{y},\bar{z})$ and
$(p_i,q_i,r_i)\ \ \forall i\in N$.
Step 2: Preprocessing
$H=\emptyset,\ B^*=\emptyset,\ f^*=1, r=1,$
$(x_i,y_i,z_i)=(\bar{x},\bar{y}-q_i,\bar{z}-r_i)$
$B_r=\{b_{[1]},b_{[2]},\ldots,b_{[n]}\}$ and
$b_{[h]}=(p_{[h]},q_{[h]},r_{[h]},x_{[h]},y_{[h]},z_{[h]}),\ \forall h\in N$.
$B_r\notin H$ ?
N
Y
Step 3: Updating set $H$
$H=H\cup\{B_r\}$.
Step 3: Run the loading process with $B_r$
Generate $f_r$.
Check evaluation value: $f_r\le f^*$ ?
Y
N
Step 4: Incumbent solution
$B^*\leftarrow B_r$.
Step 6: Loading pattern alteration
Run position-swap and rotation operators to alter the structure of $B_r$.
Stop conditions: r> $T_1$, run time>$T_2$ or optimal?
N
Y
Step 5: Looping
$r=r+1$;
$B_r\leftarrow B_{r-1}$.
Output: Best solution or infeasibility
If $B^*\neq\emptyset$, the best solution $B^*$ is yielded;
Otherwise, no feasible solution is found.
End

**Fig. 7.** Flowchart of the proposed loading placement heuristic.

compare the computational times and solution qualities required by the reference MILP model, SL, BL and the proposed loading placement heuristic.

The results of Experiment 1 solved with the reference MILP, SL, BL and the proposed heuristic were reported in Tables 4 and 5. Table 4 shows that for small-scale 3DCLPs ($n < 10$) optimal solutions can be obtained in a reasonable time (less than 360 s) by the reference MILP. Subsequently, in order to compare the efficiencies of SL, BL and the proposed heuristic, the stopping criterion of all the three methods is set as "if the objective value of the yielded solution is equal to that of the optimal solution obtained by MILP". All three heuristic algorithms were run 50 times for each benchmark instance, and the best, worst, and median results among these 50 runs are tabulated in Table 4. Compared with the reference MILP, SL and BL, the proposed heuristic needs much less CPU time (in seconds) to reach the optimal objective values for all Instances 1–10.

On the other hand, the stopping criterion for comparing the effectiveness of all the three methods is set as "if the run time ($T_2$) reaches 10 s". Table 5 shows the computational results of SL, BL and the proposed heuristic in Experiment 1. For Instance 1, the solutions of all the best, worst, and median cases yielded by BL and the proposed heuristic are optimal, and SL obtains an optimal solution in the best case while it fails to obtain an optimal solution within the time limit in the median and worst cases. SL even fails to obtain optimal solutions for Instances 8, 9 and 10, and BL also fails to obtain optimal solutions for Instance 9 and 10. The proposed heuristic only fails to get an optimum in Instance

**Table 1**
Dataset in Instances 1–4 (cf. Problem 1–4 in Tsai et al., 2014).

| Instance # | Box sizes | | | | | | $n$ | $(\bar{x},\bar{y},\bar{z})$ |
|---|---|---|---|---|---|---|---|---|
| | (25, 8, 6) | (20, 10, 5) | (16, 7, 3) | (15, 12, 6) | (22, 8, 3) | (20, 10, 4) | | |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | (35, 26, 6) |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 | 5 | (35, 28, 6) |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | (50, 28, 6) |
| 4 | 2 | 1 | 1 | 1 | 1 | 1 | 7 | (50, 22, 14) |

**Table 2**
Dataset in Instances 5–6 (cf. Problem 5–6 in Tsai et al., 2014).

| Instance # | Box sizes | | | | $n$ | $(\bar{x}, \bar{y}, \bar{z})$ |
|---|---|---|---|---|---|---|
| | (2, 2, 2) | (3, 3, 3) | (4, 4, 4) | (5, 5, 5) | | |
| 5 | 3 | 3 | 1 | 1 | 8 | (20, 8, 5) |
| 6 | 3 | 3 | 2 | 1 | 9 | (20, 8, 6) |

10. The computational results reveal that for small-size 3DCLPs the proposed heuristic is not only effective but also efficient.

## 4.2. Experiment 2

Experiment 2 was conducted on the large-scale instances with $n$ up to 240, where the datasets were generated by referring to an online desktop computer retailer (cf. Tables 6 and 7). The following data are investigated from an online desktop computer retailer in Taiwan (PChome, http://shopping.pchome.com.tw/). There are eight types of boxes used for loading different types of PC equipment. Table 6 lists the size of each type of boxes and Table 7 shows the arrangement of boxes to be allocated in Instances 11–16. In each instance, the listed boxes are to be shipped by a container with a single trip. For example, in Instance 11 there are five sets of identical host cases, mice and scatters with wires and drivers, keyboards, 22″ LCD monitors, 24″ LCD monitors, 27″ LCD monitors, 29″ LCD monitors and 32″ LCD monitors. A container with the volume $(500, 150, 150)$ as listed in Table 7 is used by Hsin-Chu Transportation Logistics in Taiwan for shipment. In order to evaluate the solution quality in a reasonable time, the stopping criterion is set as “if the run time ($T_2$) reaches 180 s” for these three heuristic

**Table 6**
Types of boxes for an online desktop computer retailer.

| Type of boxes | Equipment packed into box | Box size $(p_i, q_i, r_i)$ |
|---|---|---|
| 1 | Host case | (57, 53, 24) |
| 2 | Mouse and Scatter of Wire and Driver | (39, 34, 30) |
| 3 | Keyboard | (51, 19, 4) |
| 4 | 22″ LCD Monitor | (58, 12, 39) |
| 5 | 24″ LCD Monitor | (63, 12, 41) |
| 6 | 27″ LCD Monitor | (69, 16, 45) |
| 7 | 29″ LCD Monitor | (80, 13, 39) |
| 8 | 32″ LCD Monitor | (86, 51, 17) |

**Table 3**
Dataset in Instances 7–10 (cf. Problem 7–10 in Tsai et al., 2014).

| Instance # | Box sizes | | | | | | $n$ | $(\bar{x}, \bar{y}, \bar{z})$ |
|---|---|---|---|---|---|---|---|---|
| | (56, 50, 29) | (57, 43, 25) | (39, 34, 30) | (51, 19, 4) | (61, 45, 25) | (67, 48, 27) | | |
| 7 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | (150, 57, 30) |
| 8 | 1 | 1 | 1 | 1 | 1 | 0 | 5 | (120, 95, 30) |
| 9 | 2 | 1 | 1 | 1 | 1 | 0 | 6 | (100, 81, 50) |
| 10 | 2 | 1 | 1 | 1 | 1 | 1 | 7 | (120, 89, 50) |

**Table 4**
Efficiency comparison of Instances 1–10 in Experiment 1.

| Instance # | # Of boxes ($n$) | Optimal Obj.($x^*$) | Reference MILP (CPU Times) | SL CPU times (Best/Median/Worst) | BL CPU times (Best/Median/Worst) | Proposed algorithm CPU times (Best/Median/Worst) |
|---|---|---|---|---|---|---|
| 1 | 4 | 28 | 10.8 | (2.3/7.3/15.5) | (3.1/9.0/12.1) | (1.1/3.1/5.1) |
| 2 | 5 | 30 | 35.3 | (3.0/7.9/14.5) | (3.0/8.5/14.6) | (2.1/5.6/7.5) |
| 3 | 6 | 35 | 180.5 | (3.9/9.6/18.5) | (3.9/10.6/14.3) | (2.3/7.1/9.3) |
| 4 | 7 | 28 | 355.2 | (5.3/17.5/22.5) | (6.7/15.5/22.5) | (3.1/5.3/11.3) |
| 5 | 8 | 9 | 99.3 | (2.2/13.1/15.5) | (2.1/6.5/9.1) | (2.0/4.6/7.8) |
| 6 | 9 | 10 | 190.5 | (3.1/11.3/20.5) | (3.4/8.7/13.6) | (2.1/6.7/8.1) |
| 7 | 4 | 127 | 11.4 | (4.1/8.8/15.4) | (5.4/7.9/13.6) | (2.9/3.1/4.5) |
| 8 | 5 | 102 | 25.1 | (4.3/10.3/19.5) | (4.9/11.7/17.8) | (4.5/6.9/9.5) |
| 9 | 6 | 92 | 75.4 | (15.1/19.4/25.5) | (16.5/17.7/25.3) | (5.9/8.3/12.3) |
| 10 | 7 | 101 | 301.3 | (32.5/49.1/55.8) | (31.6/46.3/55.1) | (18.2/21.7/25.9) |

**Table 5**
Effectiveness comparison of Instances 1–10 in Experiment 1 (within 10-s run time).

| Instance # | # Of boxes ($n$) | SL Objective value ($x$) (Best/Median/Worst) | BL Objective value ($x$) (Best/Median/Worst) | Proposed algorithm Objective value ($x$) (Best/Median/Worst) |
|---|---|---|---|---|
| 1 | 4 | ($28^*$, 34, 37) | ($28^*$, $28^*$, $28^*$) | ($28^*$, $28^*$, $28^*$) |
| 2 | 5 | ($30^*$, 37, 43) | ($30^*$, $30^*$, $30^*$) | ($30^*$, $30^*$, $30^*$) |
| 3 | 6 | ($35^*$, 40, 45) | ($35^*$, $35^*$, $35^*$) | ($35^*$, $35^*$, $35^*$) |
| 4 | 7 | ($28^*$, 30, 33) | ($28^*$, 29, 29) | ($28^*$, $28^*$, $28^*$) |
| 5 | 8 | ($9^*$, 12, 12) | ($9^*$, $9^*$, $9^*$) | ($9^*$, $9^*$, $9^*$) |
| 6 | 9 | ($10^*$, 14, 14) | ($10^*$, $10^*$, $10^*$) | ($10^*$, $10^*$, $10^*$) |
| 7 | 4 | ($127^*$, $127^*$, $127^*$) | ($127^*$, $127^*$, $127^*$) | ($127^*$, $127^*$, $127^*$) |
| 8 | 5 | (107, 135, 145) | ($102^*$, $102^*$, 145) | ($102^*$, $102^*$, $102^*$) |
| 9 | 6 | (112, 122, 142) | (112, 113, 113) | ($92^*$, $92^*$, $92^*$) |
| 10 | 7 | (113, 127, 127) | (113, 113, 113) | (105, 105, 105) |

**Table 7**
Dataset in Instance 11–16 (referring to an online desktop computer retailer).

| Instance # | Box type | | | | | | | | No. of boxes ($n$) | Volume of container $(\bar{x}, \bar{y}, \bar{z})$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | |
| 11 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 40 | (500, 150, 150) |
| 12 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 80 | (500, 150, 150) |
| 13 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 120 | (500, 200, 200) |
| 14 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 160 | (500, , 200, 200) |
| 15 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 200 | (500, 200, 200) |
| 16 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 240 | (500, 250, 200) |

**Table 8**
Computational results for Instances 11–16 of Experiment 2.

| Instance # | MILP | SL | | | BL | | | Proposed algorithm | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Optimal solution $(x, y, z)$ | Best solution $(x, y, z)$ | Worst solution $(x, y, z)$ | VU rate (%) | Best solution $(x, y, z)$ | Worst solution $(x, y, z)$ | VU rate (%) | Best solution $(x, y, z)$ | Worst solution $(x, y, z)$ | VU rate (%) |
| 11 | – | (140, 149, 150) | (182, 148, 150) | 54.19 | (137, 148, 150) | (182, 149, 150) | 55.75 | (120, 148, 149) | (136, 150, 150) | 64.07 |
| 12 | – | (295, 150, 150) | (310, 150, 150) | 51.09 | (265, 149, 150) | (295, 150, 150) | 57.25 | (258, 148, 148) | (275, 150, 150) | 60.00 |
| 13 | – | (295, 200, 200) | (318, 200, 200) | 43.11 | (245, 200, 200) | (295, 200, 200) | 51.90 | (195, 198, 195) | (206, 200, 200) | 67.56 |
| 14 | – | (330, 200, 195) | (389, 199, 195) | 52.70 | (248, 199, 195) | (300, 199, 195) | 70.47 | (234, 194, 195) | (234, 200, 195) | 76.61 |
| 15 | – | (405, 200, 200) | (454, 198, 195) | 52.33 | (307, 198, 195) | (405, 198, 195) | 71.52 | (293, 200, 195) | (293, 200, 195) | 74.19 |
| 16 | – | (410, 250, 200) | (458, 250, 200) | 49.62 | (282, 247, 200) | (410, 247, 200) | 73.03 | (278, 246, 195) | (280, 246, 195) | 76.28 |

Volume Utilization (VU) rate = $\left(\sum_i (p_i q_i r_i)/(xyz)\right) \times 100\%$, where $xyz$ means the best solution.

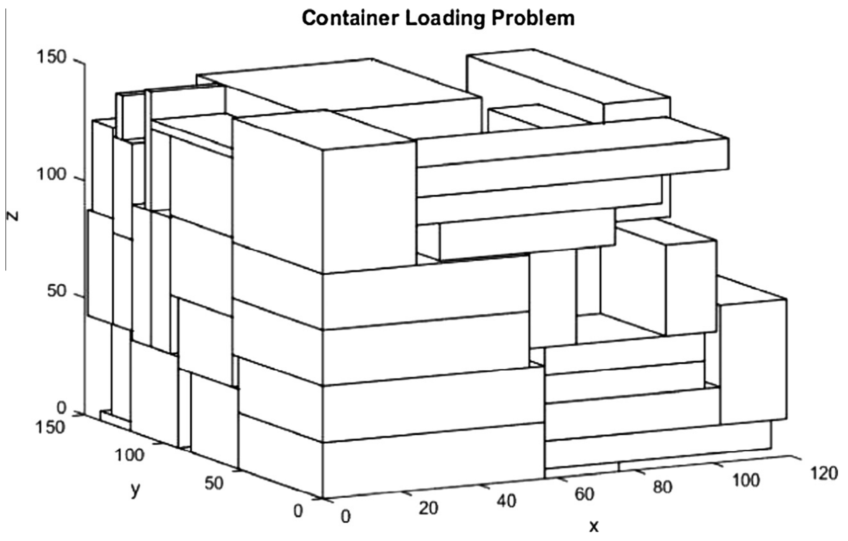

**Fig. 8.** The graphical representation of the proposed heuristic solution in Instance 11.

methods. In addition to the objective function, the volume utilization rate was used to identify the effectiveness.

All three heuristic algorithms were run 50 times for each instance. The best and worst results are tabulated in Table 8. Table 8 demonstrates that the reference MILP fails to produce optimal solutions within 6000 CPU seconds for those six instances, and the proposed heuristic is dominant in solution quality and computation time for all experimental instances. Taking Instance 11 for example, we can see that the best solution, worst solution, and volume utilization rate of SL are (140,149,150), (182,148,150), and 54.19. Those of BL are (137,148,150), (182,149,150), and 55.75. The proposed heuristic produces (120,148,149), (136,150,150), and 64.07. The graphical representation of the solution generated by the proposed heuristic in Instance 11 is depicted in Fig. 8. The computational results imply that the proposed heuristic can yield a solution not only with the best objective value but also a highest container space utilization.

The changes of the objective values over time for Instances 11–16 are shown in Fig. 9, where the $x$-axis stands for the computation time (in seconds) and the $y$-axis represents the objective value. The diamond, cross, and triangle represent the objective values yielded by SL, BL, and the proposed heuristic, respectively. From the tendency of the diagrams, we observe that proposed heuristic outperforms SL and BL from the starting point to the end point (CPU time in 180 s) for all the instances. The results imply that our proposed algorithm could avoid the aforementioned deficiencies in efficiency and effectiveness observed in the existing heuristics.

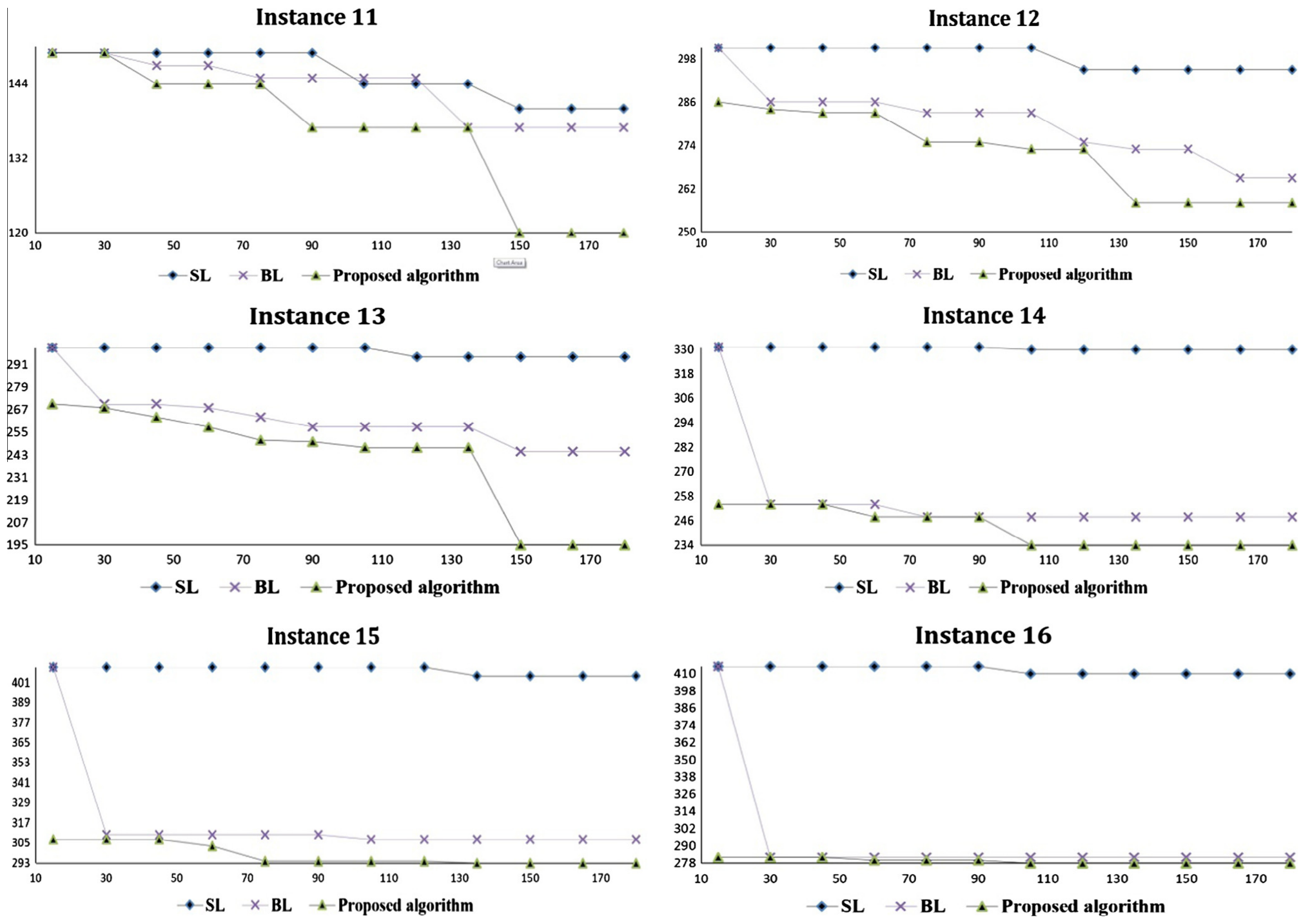

**Fig. 9.** The evolution diagrams of objective values in Instances 11–16.

## 5. Conclusions

This study has proposed a simple but effective polynomial-time loading placement heuristic algorithm to solve the 3DCLP. Numerical experiments on the benchmark instances derived from Tsai et al. (2014) have demonstrated that the reference MILP is incapable for large-size 3DCLPs while the proposed heuristic yields quality solutions and is more efficient and effective than two state-of-the-art heuristic methods. The effectiveness of the proposed heuristic could be attributed to the placement approach, which employs three-dimensional space searching and utilizes the space close to boundaries in the early stage. Also, the proposed heuristic is efficient due to its simplicity of design.

Future research could be conducted by designing an advanced heuristic and making the theoretical analyses such as error bound and reliability to enhance solving time and solution quality. Another direction for further study could be the development of advanced solution technique for the nonlinear-objective 3DCLP considered in literature (cf. Hu et al., 2012; Tsai & Li, 2006; Tsai et al., 2014). According to the preliminary test, the fundamental of the proposed heuristic procedure has the potential for dealing with the nonlinear objective function. Other directions are to take more logistic factors and transportation situations and into consideration, such as strong heterogeneity of box sizes, the weights of boxes, the transportation costs, the delivery times, and to integrate the concept of deterministic method into the proposed heuristic to enhance the solution quality.

## Acknowledgments

The authors are grateful to the editors and the anonymous referees for providing insightful comments and suggestions, which have helped improve the quality of the paper. The first author was supported by the Fundamental Research Funds for the Central Universities of China 268SWJTU15WCX01. The corresponding author was supported by the National Science Council of Taiwan under Grants NSC 102-2410-H-030-063-MY3.

## References

Almeida, A., & Figueiredo, M. B. (2010). A particular approach for the three-dimensional packing problem with additional constraints. *Computers & Operations Research, 37*, 1968–1976.
Bischoff, E. E., & Marriott, M. D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research, 44*, 267–276.
Bortfeldt, A., Gehring, H., & Mack, D. (2003). A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing, 29*, 641–662.
Bortfeldt, A., & Mack, D. (2007). A heuristic for the three-dimensional strip packing problem. *European Journal of Operational Research, 183*, 1267–1279.
Chen, C. S., Lee, S. M., & Shen, Q. S. (1995). An analytical model for the container loading problem. *European Journal of Operational Research, 80*, 68–76.
Chou, P.-F., & Lu, C.-S. (2009). Assessing service quality, switching costs and customer loyalty in home-delivery services in Taiwan. *Transport Reviews, 29*, 741–758.
Eley, M. (2002). Solving container loading problems by block arrangement. *European Journal of Operational Research, 141*, 393–409.
Faina, L. (2000). A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research, 126*(2), 340–354.
Fanslau, T., & Bortfeldt, A. (2010). A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing, 22*(2), 222–235.
George, J. A., & Robinson, D. F. (1980). A heuristic for packing boxes into a container. *Computers & Operations Research, 7*, 147–156.

GUROBI, 2014. Gurobi Optimizer 5.6.2. Reference Manual. <www.groubi.com>.

Hifi, M. (2004). Exact algorithms for unconstrained three-dimensional cutting problems: A comparative study. *Computers & Operations Research, 31*, 657–674.

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

Holland, J. H. (1975). *Adaptation in natural and artificial systems*.Cambridge, MA: MIT Press.

Hu, N. Z., Li, H. L., & Tsai, J. F. (2012). Solving packing problems by a distributed global optimization algorithm. *Mathematical Problems in Engineering*. http://dx.doi.org/10.1155/2012/931092.

Lins, L., Lins, S., & Morabito, R. (2002). An *n*-tet graph approach for non-guillotine packings of *n*-dimensional boxes into an *n*-container. *European Journal of Operational Research, 141*, 421–439.

Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research, 141*, 382–392.

Ratcliff, M. S. W., & Bischoff, E. E. (1998). Allowing for weight considerations in container loading. *OR Spectrum, 20*, 65–71.

Tsai, J. F., & Li, H. L. (2006). A global optimization method for packing problems. *Engineering Optimization, 38*, 687–700.

Tsai, J. F., Wang, P. C., & Lin, M. H. (2014). A global optimization approach for solving three-dimensional open dimension rectangular packing problems. *Optimization*. http://dx.doi.org/10.1080/02331934.2013.877906.

Zhang, D., Peng, Y., & Leung, S. C. H. (2012). A heuristic block-loading algorithm based on multi-layer search for the container loading problem. *Computers & Operations Research, 39*, 2267–2276.

Zhu, W., Huang, W., & Lim, A. (2012). A prototype column generation strategy for the multiple container loading problem. *European Journal of Operational Research, 223*, 27–39.