Computers & Industrial Engineering 90 (2015) 305–313

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers

Xueping Li [*], Kaike Zhang

Department of Industrial and Systems Engineering, The University of Tennessee at Knoxville, Knoxville, TN 37996, USA

ARTICLE INFO

Article history:
Received 13 January 2015
Received in revised form 12 October 2015
Accepted 13 October 2015
Available online 17 October 2015

Keywords:
Three-dimensional bin packing (3D-BPP)
Differential evolution (DE)
Mathematical programming
Evolutionary algorithms (EAs)

ABSTRACT

We consider a multiple container loading problem, commonly known as the three-dimensional bin packing problem (3D-BPP), which deals with maximizing container space utilization while the containers available for packing are heterogeneous, i.e., varying in size. The problem has wide applications in cargo transportation, warehouse management, medical packaging, and so on. We develop a differential evolution (DE) algorithm hybridized with a novel packing heuristic strategy, best-match-first (BMF), which generates a compact packing solution based on a given box packing sequence and a container loading sequence. The effectiveness of the proposed algorithm is evaluated on a set of industrial instances and randomly generated instances. The results show that the proposed algorithm outperforms existing solution approaches in terms of solution quality.

1. Introduction

The multiple container loading problem can be stated as follows. There are m total boxes and each box i has dimensions (l_i, w_i, h_i). These boxes need to be packed into N containers with various dimensions (L_j, W_j, H_j). The boxes can be rotated in six ways. The problem is to orthogonally pack all boxes into a subset of containers such that the space utilization ratio is maximized; or equivalently, the wasted space of containers is minimized. The problem we consider is a generalization of the classic three-dimensional bin packing problem (3D-BPP) in which containers are identical. For clarity, we denote the problem under study to 3D-BPP-*h*, where *h* stands for heterogeneous, and we use the notation of 3D-BPP-*i* for the case of identical containers. The 3D-BPP-*i* is a well known operations research optimization problem that can be used to model many industrial operations, i.e., container and pallet loading, cargo and warehouse management, medical supplies packaging, and so on. It has been reasonably well studied (Martello, Pisinger, & Vigo, 2000; Lodi, Martello, & Vigo, 2002; Lodi, Martello, & Vigo, 2004; Martello, Pisinger, Vigo, Boef, & Korst, 2007; Fekete, Schepers, & van der Veen, 2007; Gonçalves & Resende, 2013). A closely related problem is to pack boxes into a single container. A number of studies have focused on this problem, and both exact and heuristic algorithms have been proposed, e.g., (Martello et al., 2000; Padberg, 2000; Karabulut & İnceoğlu, 2005; Parreño, Alvarez-Valdes, Tamarit, & Oliveira, 2008; Kang, Moon, & Wang, 2012). However, in many industrial situations, containers are different in sizes. For example, in the shipping and transportation industry, several types of standard containers with different dimensions are used. For marine transport, standard containers of 20 or 40 feet length are used. For land or air transport, many kinds of containers are used (Takahara & Miyamoto, 2005).

As a NP-hard problem, 3D-BPP-*i* problems have attracted much interest in the operations research community, and both exact and heuristic algorithms for this problem have been developed in past decades. While exact algorithms can find optimal solutions, it usually takes huge amount of time to solve even just moderate-sized instances. Heuristic algorithms, which cannot guarantee optimality, are often capable of providing fairly effective solutions with much less computational effort. Meta-heuristic algorithms, i.e., genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO), hold a reputation in solving difficult combinatorial optimization problems. The problem we consider, i.e., 3D-BPP-*h*, is NP-hard in the strong sense since it generalizes the 3D-BPP-*i*. In addition, it is more difficult in sense of combinatorial optimization because it not only needs to determine which boxes are packed together, but also what specific container they are packed in. Therefore, heuristics become the clear choice for solving real world industrial problems. In this paper, we present a heuristic method named differential evolution (DE) to solve 3D-BPP-*h*. Computational experiments are carried out on extensive random instances and some industrial instances. To the best of our knowledge, this is the first in the literature to tackle the 3D-BPP-*h* problem using the DE.

* Corresponding author at: 522 John D. Tickle Building, 851 Neyland Drive Knoxville, TN 37996-2315, USA. Tel.: +1 865 974 7648; fax: +1 865 974 0588.
E-mail addresses: Xueping.Li@utk.edu (X. Li), kzhang7@vols.utk.edu (K. Zhang).

http://dx.doi.org/10.1016/j.cie.2015.10.007
0360-8352/© 2015 Elsevier Ltd. All rights reserved.

The remainder of this paper is organized as follows. Section 2 reviews related literature. A mathematical formulation for the problem is presented in Section 3. Section 4 introduces the new approach with a detailed description of the DE and a novel packing strategy. Computational results are reported in Section 5. We conclude the study in Section 6.

2. Literature review

In the standard 3D-BPP-*i*, the rectangular boxes need to be packed orthogonally into a minimal number of rectangular containers of identical size. Martello et al. (2000) present exact and approximation algorithms for the 3D-BPP-*i* and discuss the lower bounds. Their algorithms are based on a concept called "corner points" which is used to track the feasible placement regions and a Branch & Bound procedure is employed to verify whether a set of boxes can be placed into a container. The algorithms are improved by Martello et al. (2007) with a new version of the procedure to compute corner points and an updated Branch & Bound algorithm. The algorithms are able to solve moderately large instances to optimality for the general 3D-BPP and its robot-packable variant. Fekete et al. (2007) develop a two-level tree search algorithm for solving higher-dimensional packing problems to optimality. Faroe, Pisinger, and Zachariasen (2003) provide a heuristic based on a guided local search for packing items with fixed orientations into a minimum number of identical bins. Lodi et al. (2002) develop a tabu search framework by exploiting a new constructive procedure for the problem where the carton orientations are fixed, i.e., they cannot be rotated, and a unified tabu search code for general multi-dimensional bin packing problems is developed later Lodi et al. (2004). Recently, Gonçalves and Resende (2013) present a biased random-key genetic algorithm (BRK-GA) for 2D and 3D bin packing problems in which a novel placement heuristic is proposed and hybridized in a genetic algorithm based on random keys. They use a concept of empty maximal-spaces to manage feasible placement positions, which is used in the heuristic to determine a bin and the free maximal space after each box is placed. Almeida and Figueiredo (2010) study the problem by considering some additional restrictions on the placement and propose a mathematical formulation as well as two special designed heuristic algorithms.

The 3D-BPP-*i* with a single bin aims to select a subset of boxes which can be packed into a container to pursue high utilization ratio. This problem is also called the single container loading problem (SCLP) in the literature. Padberg (2000) proposes an MILP formulation with a polyhedral analysis. Martello et al. (2000) present a two-level Branch & Bound algorithm. Their computation experiments demonstrate the algorithm can solve instances with 90 boxes to optimality in a few minutes. Kang et al. (2012) present a hybrid genetic algorithm for a 3D-BPP in which boxes are packed into a single bin to maximize the number of boxes packed. They propose a heuristic packing strategy referred to as the Improved-Deepest-Bottom-Left with Fill (I-DBLF) algorithm based on the deepest bottom left packing method developed by Karabulut and İnceoğlu (2005). Computational efficiency of the proposed packing strategy is achieved by introducing cuboid space objects which allow to pre-check the feasibility of box insertion. Liu, Tan, Xu, and Liu (2014) present a binary tree search algorithm for the SCLP. In this algorithm, all the boxes are grouped into strips and layers and generate solutions that satisfy the full support constraint, orientation constraint and guillotine cutting constraint. Zheng, Chien, and Gen (2015) develop a multi-objective genetic algorithm with a multi-population strategy and a fuzzy logic controller (FLC) to maximize the container space utilization and the value of total loaded boxes. Another variation of this problem is referred to as the open dimension problem (ODP). The ODP allows a single variable dimension to occur in the packing planning process. Bortfeldt and Mack (2007) present a layer-building heuristic method for the 3D-BPP which aims to packing boxes into a single container to minimize required container length. Wu, Li, Goh, and de Souza (2010) study a 3D-BPP problem with variable bin heights. They first propose an exact mathematical model based on Chen, Lee, and Shen (1995)'s model and then design a genetic algorithm to hybridize with a heuristic packing procedure. This problem is further studied by He, Wu, and de Souza (2012), and the authors develop an improved genetic algorithm for the 3D-BPP with variable carton orientations. They improve the decoding procedure for tighter packing and higher computational efficiency, then a novel global search framework (GSF) is proposed based on an evolutionary gradient method.

Unlike the 3D-BPP-*i*, little attention has been paid in the literature so far to the 3D-BPP-*h*. Chen et al. (1995) seem to be the first one providing a 0–1 mixed integer linear programming (MILP) model to solve the 3D-BPP-*h* with variable orientations and various bins sizes. The model uses left, right, top and bottom decision variables to determine the relative positions of boxes and orientation variables to mimic different packing orientations. However, this MILP model can only solve small instances to optimality. A very similar problem is studied by Takahara and Miyamoto (2005), in which an evolutionary approach using GA is proposed, where a pair of sequences (one for boxes and another for containers) is used as the genotype, and a heuristic is used to determine the loading plan given the sequence of boxes and containers. They use a package loading heuristic called "Branch Heuristics" to convert a chromosome to a packing solution. An obvious drawback of this heuristic procedure is that it fails to make full use of the available space. Eley (2003) proposes an approach based on a set partitioning formulation and use a tree search based heuristic to pregenerate single bin packing patterns. Their model is extended by Che, Huang, Lim, and Zhu (2011) and Zhu, Huang, and Lim (2012), who propose a heuristic and an approximation algorithm to generate columns. This line of methods may fail to solve problems when containers are heterogeneous since the patterns that need to be generated increase exponentially with the number of container types. Li, Zhao, and Zhang (2014) study a 3D-BPP with heterogeneous bins using a genetic algorithm. Recently, Alvarez-Valdes, Parreño, and Tamarit (2015) study the problem where several types of bins of different sizes and costs are available and the objective is to minimize the total cost of the bins used for packing the boxes. They develop a new lower bound for the problem based on an integer programming formulation.

In this paper, we study 3D-BPP-*h* with variable orientations and various bins sizes. A differential evolution algorithm with a heuristic procedure is proposed. In the algorithm, a chromosome contains a box packing sequence, and a container loading sequence is used to heuristically generate packing solution. These sequences evolve in the differential evolution algorithm which leads to improved solutions. The heuristic procedure is able to decode these sequences to a compact packing solution.

3. Mathematical programming model

We define the following notations. There are total m boxes indexed as 1, ..., m to pack, and total N containers indexed as 1, ..., N are available. These containers have different sizes. The length, width, and height of product i are denoted by l_i, w_i, and h_i, while the length, width, and height of box j are denoted by L_j, W_j, and H_j. The boxes can be rotated in six ways which are indexed by k (Fig. 1). The problem is to find an assignment of the boxes to the containers, i.e., placing the boxes in which container at what position. The objective is to minimize the wasted space of used containers after placing all the boxes into the containers, similar to the model in Chen et al. (1995).

3.1. Decision variables

The variables are defined as:

1. Binary variables:
   - $p_{ij},\ \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,N\},\ p_{ij} = 1$, if box i is packed in container j, otherwise $p_{ij} = 0$.
   - $u_j,\ \forall j \in \{1,\dots,N\},\ u_j = 1$ if container j is used, otherwise $u_j = 0$.
   - $x_{ik}^{+},\ \forall i,\ k \in \{1,\dots,m\},\ i < k,\ x_{ik}^{+} = 1$ if boxes $i,\ k$ are packed in the same container and box i is placed to the left of box k.
   - $x_{ik}^{-},\ \forall i,\ k \in \{1,\dots,m\},\ i < k,\ x_{ik}^{-} = 1$ if boxes $i,\ k$ are packed in the same container and box i is placed to the right of box k.
   - $y_{ik}^{+},\ \forall i,\ k \in \{1,\dots,m\},\ i < k,\ y_{ik}^{+} = 1$ if boxes $i,\ k$ are packed in the same container and box i is placed behind box k.
   - $y_{ik}^{-},\ \forall i,\ k \in \{1,\dots,m\},\ i < k,\ y_{ik}^{-} = 1$ if boxes $i,\ k$ are packed in the same container and box i is placed in front of box k.
   - $z_{ik}^{+},\ \forall i,\ k \in \{1,\dots,m\},\ i < k,\ z_{ik}^{+} = 1$ if boxes $i,\ k$ are packed in the same container and box i is placed below box k.
   - $z_{ik}^{-}, \forall i,\ k \in \{1,\dots,m\},\ i < k,\ z_{ik}^{-} = 1$ if boxes $i,\ k$ are packed in the same container and box i is placed above box k.
   - $l_i^x,\ l_i^y,\ l_i^z,\ l_i^x = 1,\ l_i^y = 1,\ l_i^z = 1$ indicates the length edge of box i is placed parallel to X-axis, Y-axis or Z-axis, respectively.
   - $w_i^x,\ w_i^y,\ w_i^z,\ w_i^x = 1,\ w_i^y = 1,\ w_i^z = 1$ indicates the width edge of box i is placed parallel to X-axis, Y-axis or Z-axis, respectively.
   - $h_i^x,\ h_i^y,\ h_i^z,\ h_i^x = 1,\ h_i^y = 1,\ h_i^z = 1$ indicates the height edge of box i is placed parallel to X-axis, Y-axis or Z-axis, respectively.

2. Continuous variables:
   - $(x_i, y_i, z_i),\ \forall i \in \{1,\dots,m\}$, the three variables together denote the left-bottom-back coordinate of the box i in a container.

3.2. Objective function

The objective function is to minimize the wasted volume in containers.

$$\sum_{j=1}^{N} u_j \cdot (L_j \cdot W_j \cdot H_j) - \sum_{i=1}^{m} l_i \cdot w_i \cdot h_i$$

3.3. Constraints

$$p_{ij} \leqslant u_j \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,N\} \tag{1}$$

$$\sum_{j}^{N} p_{ij} = 1 \quad \forall i \in \{1,\dots,m\} \tag{2}$$

$$x_i + l_i \cdot l_i^x + w_i \cdot w_i^x + h_i \cdot h_i^x \leqslant L_j + (1 - p_{ij}) \cdot M \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,N\} \tag{3}$$

$$y_i + l_i \cdot l_i^y + w_i \cdot w_i^y + h_i \cdot h_i^y \leqslant W_j + (1 - p_{ij}) \cdot M \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,N\} \tag{4}$$

$$z_i + l_i \cdot l_i^z + w_i \cdot w_i^z + h_i \cdot h_i^z \leqslant H_j + (1 - p_{ij}) \cdot M \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,N\} \tag{5}$$

$$x_i + l_i \cdot l_i^x + w_i \cdot w_i^x + h_i \cdot h_i^x \leqslant x_k + (1 - x_{ik}^{+}) \cdot M \quad \forall i,\ k \in \{1,\dots,m\},\ i < k \tag{6}$$

$$x_k + l_k \cdot l_k^x + w_k \cdot w_k^x + h_k \cdot h_k^x \leqslant x_i + (1 - x_{ik}^{-}) \cdot M \quad \forall i,\ k \in \{1,\dots,m\},\ i < k \tag{7}$$

$$y_i + l_i \cdot l_i^y + w_i \cdot w_i^y + h_i \cdot h_i^y \leqslant y_k + (1 - y_{ik}^{+}) \cdot M \quad \forall i,\ k \in \{1,\dots,m\},\ i < k \tag{8}$$

$$y_k + l_k \cdot l_k^y + y_k \cdot w_k^y + h_k \cdot h_k^y \leqslant y_i + (1 - y_{ik}^{-}) \cdot M \quad \forall i,\ k \in \{1,\dots,m\},\ i < k \tag{9}$$

$$z_i + l_i \cdot l_i^z + w_i \cdot w_i^z + h_i \cdot h_i^z \leqslant z_k + (1 - z_{ik}^{+}) \cdot M \quad \forall i,\ k \in \{1,\dots,m\},\ i < k \tag{10}$$

$$z_k + l_k \cdot l_k^z + w_k \cdot w_k^z + h_k \cdot h_k^z \leqslant z_i + (1 - z_{ik}^{-}) \cdot M \quad \forall i,\ k \in \{1,\dots,m\},\ i < k \tag{11}$$

$$x_{ik}^{+} + x_{ik}^{-} + y_{ik}^{+} + y_{ik}^{-} + z_{ik}^{+} + z_{ik}^{-} \geqslant p_{ij} + p_{kj} - 1 \quad \forall i,\ k \in \{1,\dots,m\},\ i < k,\ \forall j \in \{1,\dots,N\} \tag{12}$$

$$l_i^x + l_i^y + l_i^z = 1 \quad \forall i \in \{1,\dots,m\} \tag{13}$$

$$w_i^x + w_i^y + w_i^z = 1 \quad \forall i \in \{1,\dots,m\} \tag{14}$$

$$h_i^x + h_i^y + h_i^z = 1 \quad \forall i \in \{1,\dots,m\} \tag{15}$$

$$l_i^x + w_i^x + h_i^x = 1 \quad \forall i \in \{1,\dots,m\} \tag{16}$$

$$l_i^y + w_i^y + h_i^y = 1 \quad \forall i \in \{1,\dots,m\} \tag{17}$$

$$p_{ij},\ u_j,\ x_{ik}^{+},\ x_{ik}^{-},\ y_{ik}^{+},\ y_{ik}^{-},\ z_{ik}^{+},\ z_{ik}^{-},\ l_i^x,\ l_i^y,\ l_i^z,\ w_i^x,\ w_i^y,\ w_i^z,\ h_i^x,\ h_i^y,\ h_i^z \in \{0,1\} \quad \forall i,\ k \in \{1,\dots,m\},\ \forall j \in \{1,\dots,N\}$$

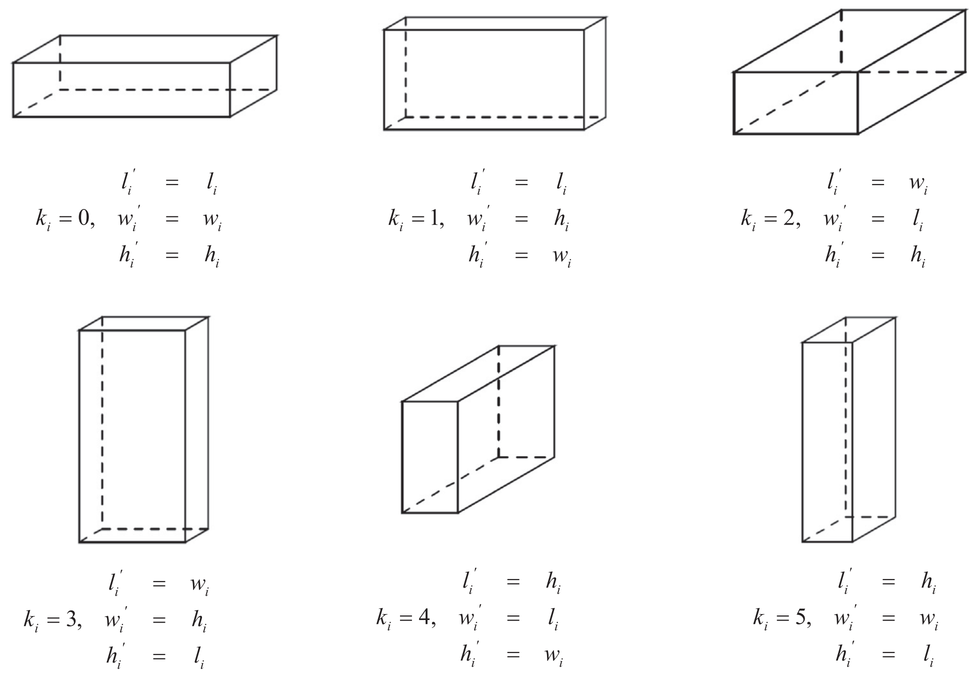
$$x_i,\ y_i,\ z_i \geqslant 0 \quad \forall i \in \{1,\dots,m\}$$

Constraints (1) and (2) ensure that all boxes are packed and a box can be assigned to a container only if this container is used. Constraints (3)–(5) ensure all boxes are placed within the container's dimension. Constraints (6)–(11) indicate that any two boxes i and k do not overlap with each other. Constraint (12) states that if two boxes are packed in the same container, then at least one of following relative positions, i.e., left, above, behind, will be true. This guarantees that they are not overlapped. Constraints (13)–(17) together define the proper orientation of a box. More detailed explanation of a similar model can be found in Chen et al. (1995).

4. Hybrid differential evolution algorithm

4.1. Overview of the algorithm

The proposed algorithm is based on a basic DE with a new heuristic packing strategy. The heuristic packing procedure uses the concept of empty maximal spaces to efficiently manage free space in containers. The heuristic packing strategy generates a compact packing solution based on a given box packing sequence (BPS) and a container loading sequence (CLS). The differential evolution algorithm is used to evolve such sequences.



$k_i = 0,\quad l_i' = l_i,\ w_i' = w_i,\ h_i' = h_i$

$k_i = 1,\quad l_i' = l_i,\ w_i' = h_i,\ h_i' = w_i$

$k_i = 2,\quad l_i' = w_i,\ w_i' = l_i,\ h_i' = h_i$

$k_i = 3,\quad l_i' = w_i,\ w_i' = h_i,\ h_i' = l_i$

$k_i = 4,\quad l_i' = h_i,\ w_i' = l_i,\ h_i' = w_i$

$k_i = 5,\quad l_i' = h_i,\ w_i' = w_i,\ h_i' = l_i$

Fig. 1. Six cases of item rotation.

4.2. Differential evolution

The DE (Storn & Price, 1997) is a powerful stochastic real parameter optimizer. Compared with other evolutionary algorithms (EAs), the DE is much simpler and more straightforward to implement, requiring fewer control parameters while performing better in a variety of fields (Das & Suganthan, 2011). Like traditional EAs, the DE has four main stages: initialization, mutation, crossover, and selection.

The DE starts with a generation of Np individuals: $X_{1,G},\ i = 1,\dots,Np$ where the index i denotes the ith individual and G denotes the generation to which the individual belongs. The DE generates the next generation by perturbing scaled differences of randomly selected and distinct population members, a process which differentiates DE from other EAs.

Mutation is a process that generates Np perturbed individuals by applying scaled differences of randomly selected individuals. The ith perturbed individual, $V_{i,G}$, is generated as follows:

$$V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G})$$

where $r1$, $r2$ and $r3$ are randomly generated in $(1,\dots,Np)$ and $r1 \neq r2 \neq r3 \neq i$. F is a control parameter which usually takes value in $[0.4, 1]$.

Crossover is operated over a target individual $X_{i,G}$ and its counterpart perturbed individual $V_{i,G}$ to generate a trial solution $U_{i,G}$.

Crossover operation is outlined as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } (random_{i,j}(0,1) \leqslant Cr \text{ or } j = j_{rand}) \\ x_{j,i,G} & \text{otherwise} \end{cases}$$

where $random_{i,j}(0,1)$ is a uniformly distributed random number in [0, 1] for each i, j. j_{rand} is randomly chosen from $[1, Np]$ to ensure at least one component of a new vector comes from $V_{i,G}$. Cr is a control parameter called crossover rate.

Selection process also distinguishes the DE from other EAs. In the DE, the trial individual is only compared to one solution – its counterpart target individual in the current generation, instead of being compared with all the individuals in the current generation. The individual for the next generation is selected by the following rules:

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \leqslant f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases}$$

where $f(\cdot)$ is the evaluation function for individuals. In minimization problems, an individual with a lower objective function value will survive from the tournament selection. We use a heuristic packing strategy described in Section 4.4 to evaluate an individual.

4.3. Adaption of DE for 3D-BPP-h

Since the DE is designed to solve real number parameters optimization problems, it cannot be used directly to solve container loading problems. In this subsection, we focus on the procedures for this specific problem, which includes the individual encoding scheme and initialization.

4.3.1. Individual encoding scheme

Each individual has two parts: a vector in dimension m for boxes and a vector in dimension N for containers, i.e., $X_{i,G} = \{(b_1, b_2, \dots, b_m), (c_1, c_2, \dots, c_N)\}$. The components of these vectors take the values in $[0, 1]$. The role of the DE is to evolve the encoded solutions which represent the box packing sequence or BPS, and the vector of the container loading sequence, CLS. The decoding of these two vectors into BPS and CLS is accomplished by sorting (see Fig. 2) and the decoding process operates on the two vectors respectively. The heuristic packing strategy described in Section 4.4 takes the BPS and the CLS as inputs to generate a packing solution for each individual. The trial vectors generated in mutation process may result in some components not falling in the interval $[0, 1]$; in that case, we set the components to the nearest feasible values.

4.3.2. Population initialization

Wang and Chen (2010) and Kang et al. (2012) propose to generate special chromosomes for the initial generation in their GAs to improve the solution quality. We follow the same mechanism to generate four special individuals through sorting in descending order, according to the values of volume, length, width and height of the boxes, respectively. This mechanism is based on the observation that the bigger boxes should be packed into the box earlier. In our encoding settings, we set $b_i = r/m$, where r is the position in the sequence of the box indexed i. The components for vector for the containers are generated randomly in their domain. The rest individuals in the first generation are generated randomly. Each component of the vectors is generated independently at random in the real interval $[0, 1]$.

4.4. Heuristic packing strategy

An individual is evaluated by the following heuristic packing strategy which maps an individual into a packing solution. Given a BPS and a CLS, the heuristic packing strategy converts these sequences to a packing solution. The traditional Deepest-Bottom-Left with fill (DBLF) heuristic and its variations are popular heuristics used for solving bin packing problems (Karabulut & İnceoğlu, 2005; Wang & Chen, 2010; Kang et al., 2012). The heuristic always searches for the space with minimal x (Deepest) coordinate to place the current item, and it uses z (Bottom) and y (Left) coordinates as tie breakers. We notice, however, that this heuristic has two drawbacks. First, only one coordinate (x) plays the dominant role in choosing the candidate space. Second, either the item or the space is determined and its counterpart is then selected based on certain priority rules. This results in potential loss of combinations that may lead to better solutions. Based on this observation, we propose the best-match-first (BMF) packing heuristic. In the next sections we describe the main components of this placement strategy.

4.4.1. Empty maximal spaces

We use the concept of empty maximal spaces (EMSs) to represent the free spaces in bins, i.e., a list of the largest empty cubic spaces available for packing which are not contained in any other space. Fig. 3 illustrates the concept. The outside frame illustrates the space currently available for packing. When one blue box (the smaller one in Fig. 3) is placed in the corner, it generates three empty maximal spaces which are represented by yellow cubics (the larger ones in Fig. 3). This concept has been recently used in Gonçalves and Resende (2013) and Parreño et al. (2008) to solve bin packing problems. The maximal spaces are represented by their vertexes with minimum and maximum coordinates, (x_i, y_i, z_i) and (X_i, Y_i, Z_i), respectively. We use the *difference process* introduced by Lai and Chan (1997) to update these empty maximal spaces and adopt the elimination rules proposed by Gonçalves and Resende (2013) to accelerate the process.

4.4.2. Priority of empty maximal spaces

We define the priority of EMSs by the following rules. For two EMSs with their minimum vertex coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2). We first compare the smallest coordinate values of the two vertexes, while the vertex with smaller value is given higher priority. If they tie, we compare the second smallest values and assign higher priority to the vertex with smaller one; if the two values are still the same, we compare the third smallest values. For example, to compare two vertexes $(3, 5, 4)$ and $(6, 3, 3)$, first we compare 3 and 3 which are the smallest values in each vertex coordinate. Since they have the same value, we continue to compare the second smallest ones, i.e., 4 for the first vertex and 3 for the second one. Since $4 > 3$, we assign a higher priority to the second vertex. The reason behind this prioritization is that we intend to place boxes in one corner and its adjacent sides of the container first, then its inner space. We sort the empty spaces in each bin according to the priority defined above after we update of the EMSs each time.

4.4.3. Placement selection

The placement assignment is determined by applying the selection criteria to several possible placement candidates. In each iteration, the first k_b unpacked boxes are chosen based on the order given by BPS. Then, the first k_e EMSs in the bin currently being considered are selected in the order defined above. For these k_b boxes and k_e EMSs, we find all the feasible placement assignments with 6-way rotations of a box. Then the (*box, empty maximal space*) pair with the largest fill ratio is first chosen for the placement. When one box has several feasible placements in one free empty space, the one that has the smallest margin is selected. This can be done by calculating as follows: The dimensions of the space are $(X_1 - x_1, Y_1 - y_1, Z_1 - z_1)$ as defined; let the dimensions of the boxes after rotation as (l', w', h'); and by subtracting two vectors, we get $(X_1 - x_1 - l', Y_1 - y_1 - w', Z_1 - z_1 - h')$ which represents the margins to three faces of the space. Then, the priority of the placement is defined the same as the one used in selecting EMSs. See Fig. 4 illustrates the idea in a two-dimensional case.

Algorithm 1. Best Match First Heuristic Packing Procedure

```
Input: Boxes packing sequences BPS, container loading sequence CLS
Output: A packing solution or null if not found
1  Let OC be the list of opened containers
2  OC ← ∅
3  while BPS ≠ ∅ do
4      Let P be a priority queue of candidate placements with the priority defined in Section 4.4.3
5      boxplaced ← false
6      for each opened container c in OC do
7          Let EMSs be the empty maximal spaces in c
8          j ← 1
9          while j ⩽ EMSs.size() and boxplaced = false do
10             k ← j + ke
11             while j < k and j ⩽ EMSs.size() do
12                 for i = 1 to kb and i ⩽ BPS.size() do
13                     for all 6 orientations bo do
14                         if Box BPS_i can be placed in EMSs_j with orientation bo then
15                             Add this placement combination to P
16                 j ← j + 1
17             if P ≠ ∅ then
18                 Make the placement indicted by P_1
19                 Update EMSs
20                 boxplaced ← true
21         if boxplaced = true then
22             break
23     while CLS ≠ ∅ and boxplaced = false do
24         Let EMS be the initial empty space in CLS_1
25         OC ← OC ∪ CLS_1
26         CLS ← CLS \ CLS_1
27         for i = 1 to kb and i ⩽ BPS.size() do
28             for all 6 orientations bo do
29                 if Box BPS_i can be placed in EMS with orientation bo then
30                     Add this placement combination to P
31         if P ≠ ∅ then
32             Make the placement indicted by P_1
33             Update EMSs
34             boxplaced ← true
35     if boxplaced = false then
36         return null
37 return Packing solution
```

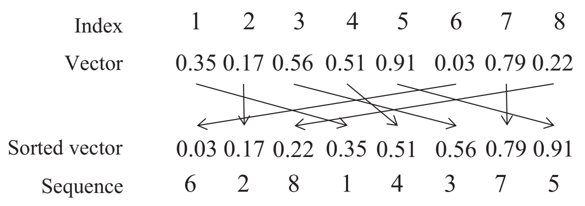
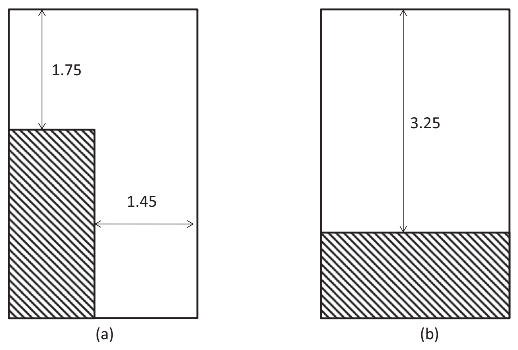

Index 1 2 3 4 5 6 7 8
Vector 0.35 0.17 0.56 0.51 0.91 0.03 0.79 0.22
Sorted vector 0.03 0.17 0.22 0.35 0.51 0.56 0.79 0.91
Sequence 6 2 8 1 4 3 7 5

Fig. 2. Illustration of decoding a vector to a sequence.



The placement in (a) has a margin of (1.45, 1.75) and (b) has a margin of (0, 3.25). Our method will choose the placement in (b), since it generates less margin in one direction.

Fig. 4. Orientation selection.

If there is no feasible placement assignment for the current boxes and spaces selection, we will try next k_e EMSs in the container, and the process continues until at least one feasible placement is found. If all EMSs in the current container are attempted, we move forward to try EMSs in the next opened container. If all opened containers are tried without finding one feasible placement, we open the next unopened container in the order given by CLS. If there is no container in the list, the algorithm stops without finding a feasible packing solution. In this case, we set the fitness of the individual to be zero. Otherwise, we always find one or more feasible placement assignments and select one with the largest fill ratio. Then we place the item in the chosen empty space, remove the item from the unpacked items list, and update the EMSs. If there is no element in the unpacked boxes list, the algorithm terminates with a solution and the fitness of the individual is set to be the fill ratio. The pseudo-code of the packing heuristic procedure BMF is given in Algorithm 1.

5. Computational study

In this section, we summarize the computational study results. The hybrid differential evolution algorithm is implemented in Java. All the experiments are conducted on a computer with 2.5 GHz CPU and 32G memory running the UNIX operating system. The parameters are set as follows: $Np = 100$, $F = 0.5$, $Cr = 0.75$, $k_b = 3$, $k_e = 3$ and the maximum generations G_{Max} is set 100. Because the problem we study, i.e., 3D-BPP-h is a generalization of 3D-BPP-i, our algorithm is capable to solve 3D-BPP-i instances. To demonstrate the efficiency of the proposed algorithm, we first compare the proposed algorithm with an exact algorithm that is designed to solve 3d-BPP-i problem. Then, we test the efficiency of the proposed algorithm by comparing it with known methods on 12 industrial instances and on random generated instances of 3d-BPP-h problem.

5.1. 3d-BPP-i problem

The propose algorithm is compared with an exact algorithm that solves 3d-BPP-i described in Martello et al. (2007). We adapt the code provided by the authors which can be obtained at http://www.diku.dk/pisinger/codes.html. Notice that their algorithm assumes the boxes may not be rotated, i.e., they are packed with each edge parallel to the corresponding bin edge. In order to make a fair comparison, we first adapt our algorithm such that it only considers a fixed box orientation. Then, we also provide the results of our algorithm in which 6-ways box orientations are allowed.

We follow the method described in Martello et al. (2000) to generate instances. The following five types of random boxes are considered for $W = L = H = 100$:

- Type 1: l_j randomly distributed in $\left[1, \frac{1}{2}L\right]$, w_j randomly distributed in $\left[\frac{2}{3}W, W\right]$, h_j randomly distributed in $\left[\frac{2}{3}H, H\right]$.
- Type 2: l_j randomly distributed in $\left[\frac{2}{3}L, L\right]$, w_j randomly distributed in $\left[1, \frac{1}{2}W\right]$, h_j randomly distributed in $\left[\frac{2}{3}H, H\right]$.
- Type 3: l_j randomly distributed in $\left[\frac{2}{3}L, L\right]$, w_j randomly distributed in $\left[\frac{2}{3}W, W\right]$, h_j randomly distributed in $\left[1, \frac{1}{2}H\right]$.
- Type 4: l_j randomly distributed in $\left[\frac{1}{2}L, L\right]$, w_j randomly distributed in $\left[\frac{1}{2}W, W\right]$, h_j randomly distributed in $\left[\frac{1}{2}H, H\right]$.
- Type 5: l_j randomly distributed in $\left[1, \frac{1}{2}L\right]$, w_j randomly distributed in $\left[1, \frac{1}{2}W\right]$, h_j randomly distributed in $\left[1, \frac{1}{2}H\right]$.

For each class k ($k = 1, \ldots, 5$), boxes of type k are chosen with the probability of 60%, and the remaining four types are chosen with the probability of 10% each. For each class, we consider instances with a number of boxes equal to 60, 80, 100, 120, 150. The containers are of the same size with $W = L = H = 100$.

Table 1 highlights our results. The column "#B" denotes size of the instances by the number of boxes. For each instance size, we solve 10 random instances. The column "#Solved" reports the number of instances among these 10 random instances that are solved to optimality within the time limit of 3600 s by the exact algorithm. The column "LB" and "UB" reports the average lower bound and the average upper bound of number of containers used.

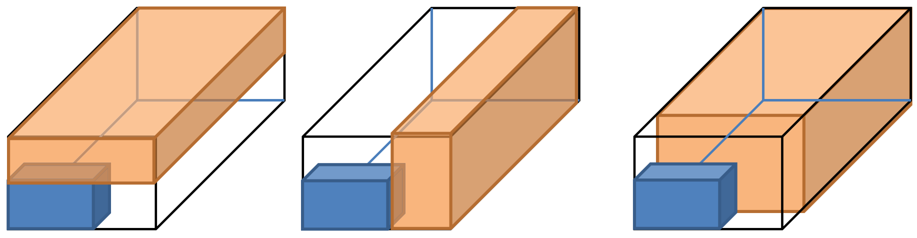


Fig. 3. Empty maximal spaces.

Table 1
A comparison of algorithms' performance on solving 3d-BPP-i problem.

Class	#B	Martello et al.' exact algorithm					DE + BMF (fixed orientation)		DE + BMF (6-ways orientation)	
		#Solved	LB	UB	Gap	Time	UB	Time	UB	Time
Class 1	50	10	13.5	13.5	0	144	13.8	132	12.2	120
	60	7	15.5	16.2	0.7	1226	16.6	33	15	34
	80	1	18.8	20.8	2	2745	21.6	49	19.7	53
	100	0	23.2	26.7	3.5	3600	26.7	66	24.7	66
	120	0	28.3	32	3.7	3600	32.1	91	29.8	90
	150	0	32.5	37.6	5.1	3600	38.6	117	35.9	115
Class 2	50	9	13.3	13.5	0.2	408	13.7	25	12.5	28
	60	9	15.9	16.2	0.3	428	16.7	34	15.2	37
	80	1	18.7	21	2.3	2949	21.3	48	19.3	53
	100	0	22.4	26	3.6	3600	26.7	66	24.4	70
	120	0	27.1	31.3	4.2	3600	32.3	85	29.9	87
	150	0	33	38.5	5.5	3600	39.3	108	36.5	110
Class 3	50	10	13.8	13.8	0	104	14	25	12.5	29
	60	7	16.1	16.6	0.5	1191	16.9	32	15.4	35
	80	1	18.4	20.3	1.9	2831	20.8	47	19.2	51
	100	1	22.4	25.4	3	3323	25.9	65	24.1	66
	120	0	27.6	32	4.4	3600	32.1	87	29.6	88
	150	0	33.8	38.3	4.5	3600	39.2	112	36	111
Class 4	50	10	22.2	22.2	0	0	22.2	21	21.4	23
	60	9	24.7	24.8	0.1	361	**24.8**	27	23.8	30
	80	7	31.4	32	0.6	1202	**31.9**	41	31.5	46
	100	1	38.8	40.9	2.1	3405	**40.8**	51	39.3	57
	120	2	46.3	48.5	2.2	3177	**47.6**	71	46.7	78
	150	0	57.4	60.9	3.5	3600	**59.8**	96	57.9	98
Class 5	50	10	11.1	11.1	0	184	11.1	49	10.3	48
	60	7	13	13.9	0.9	1375	**13.7**	57	12.7	56
	80	5	15.1	17.2	2.1	1923	**16.4**	111	15.3	100
	100	0	18.1	21.5	3.4	3600	21.9	130	20	116
	120	0	21.8	26.1	4.3	3600	**25.8**	195	23.9	161
	150	0	23.4	29.7	6.3	3600	**29.5**	302	27.8	221

Table 2
A comparison of algorithms performance on 12 industrial instances.

Inst	#B	#C	Cplex		GA + DBLF		GA + BMF		DE + BMF	
			Time	WS%	Time	WS%	Time	WS%	Time	WS%
1	12	8	2	26.67	4	26.67	3	26.67	6	26.67
2	12	8	170	15.45	3	33.25	2	33.25	3	33.25
3	12	8	6	50.64	1	50.64	3	50.64	5	50.64
4	20	8	28,800	18.30*	1	30.87	4	18.30	12	18.30
5	20	8	28,800	14.85*	1	37.59	2	21.34	61	19.63
6	20	8	28,800	22.62*	2	37.02	4	20.62	7	20.62
7	50	10	28,800	35.97*	9	35.10	14	22.48	29	22.48
8	50	10	28,800	46.87*	11	37.20	15	23.35	30	23.35
9	50	10	28,800	42.69*	9	32.39	12	20.82	34	20.82
10	78	15	28,800	51.86*	20	33.18	26	22.05	44	19.54
11	78	15	28,800	–	15	27.14	21	22.76	60	22.13
12	78	24	28,800	–	22	30.70	13	23.40	60	22.25

– No feasible solution is found within 28,800 s.
* Time limit of 28,800 s is reached. The best feasible solution found is reported.

The column "Gap" calculate the difference of the average upper bound and average lower bound. Column "Time" reports the average CPU time in seconds used in solving the instance. The proposed algorithm is a meta-heuristic algorithm; thus, it can only provide upper bounds for the problem. From the results, we observe that the exact algorithm can only solve small instances to optimality within a reasonable amount of time and the proposed algorithm can obtain very competitive solutions with much less computational efforts comparing with the exact algorithm. For instance class 4 and class 5, the proposed algorithm even finds better upper bound which are shown in bold. Also, we observe that by allowing 6-ways orientation, the average number of containers used is significantly decreased without adding more computational cost.

5.2. 3d-BPP-h problem

5.2.1. Results on industrial instances

We first test our algorithm with 12 instances that are obtained from an undisclosed pharmaceutical industry. The instances are ranged from 12 boxes, 8 containers to 78 boxes, 24 containers. We made a comparison of the performance of the proposed algorithm with several other solution methods. Table 2 summarizes

Table 3
A comparison of algorithms performance on random instances.

Class	#B	#C	GA + DBLF		GA + BMF			DE + BMF			
			Time	WS%	Time	WS%	#Sup_1	Time	WS%	#Sup_1	#Sup_2
Class 1	20	10	9	34.66	8	22.34	10	22	22.05	10	4
	50	20	43	31.24	52	19.92	10	80	19.55	10	7
	80	25	86	27.16	103	18.76	10	174	19.47	10	4
	100	30	91	23.23	96	18.38	10	191	18.88	9	3
	120	35	114	23.01	125	17.75	10	250	18.41	10	3
	150	40	130	21.35	136	17.20	9	313	19.00	9	2
Class 2	20	10	10	33.09	7	23.03	9	21	21.54	9	4
	50	20	32	29.26	47	20.00	10	79	20.02	10	5
	80	25	67	27.24	70	18.58	10	139	19.10	10	3
	100	30	86	23.76	95	17.22	10	174	18.76	10	3
	120	35	129	24.20	191	18.56	10	289	18.82	10	5
	150	40	113	22.23	143	17.62	10	288	18.83	9	5
Class 3	20	10	7	35.22	10	21.47	10	19	20.80	10	3
	50	20	43	30.59	49	19.17	10	82	18.70	10	6
	80	25	76	26.79	84	18.94	10	160	19.08	10	4
	100	30	94	24.57	115	18.51	10	201	18.81	10	6
	120	35	140	24.48	136	17.29	10	300	18.48	10	3
	150	40	159	22.42	214	16.51	10	360	18.23	10	2
Class 4	20	10	4	39.03	5	30.44	9	12	29.40	9	5
	50	20	21	28.96	22	20.25	10	46	21.20	10	5
	80	25	28	23.63	36	19.31	9	78	21.86	7	2
	100	30	36	25.12	45	19.70	9	106	22.15	9	2
	120	35	35	23.00	51	18.97	10	114	22.50	7	0
	150	40	64	21.26	76	18.07	8	198	21.38	4	2
Class 5	20	10	14	35.38	9	20.66	10	28	22.28	10	0
	50	20	116	32.71	75	18.81	10	183	17.59	10	6
	80	25	217	27.61	147	17.46	10	371	18.78	10	2
	100	30	423	27.44	226	15.98	10	669	17.13	10	3
	120	35	380	20.99	251	14.40	10	741	17.70	9	2
	150	40	480	21.38	386	14.92	9	1094	16.87	9	1

the computational results. The first column shows the index of the instances. The column "#B" and "#C' ' indicate the size of the instance with the number of boxes and the number of containers. The column "WS%" reports the percentage of wasted spaces in the solution. Because the goal is to minimize the percentage of wasted spaces, the lower this value is, the better the solution is.

We first make a comparison of the MILP model we outlined in Section 3 which is solved by the CPLEX 12.6 MIP solver. The results are shown in column "CPLEX". In the column "Time", we either report the time used to solve the model optimally or the time limit of 28,800 s (8 h) that is reached without finding a feasible solution or proven to be optimal.

A GA proposed by Wang and Chen (2010) is modified to solve 3D-BPP-*h* by adding a gene sequence to represent the container loading sequence. The parameters used are consistent with those in Wang and Chen (2010) except G_{Max} is set to 100 for a fair comparison. Since their algorithm uses a variation of DBLF as the heuristic packing strategy, we report the results in the column "GA + DBLF".

In order to test the efficiency of our heuristic packing strategy, a comparison is made with the DBLF heuristic. First, we modify the GA by using our BMF heuristic to convert sequences of boxes and containers to the packing solution. The results are reported under column "GA + BMF". In the final column "DE + BMF", we report the algorithm we have proposed in this paper.

From the results, we observe that CPLEX is only able to find an optimal solution for very small-size instances, such as 12 boxes, 8 containers. It fails to find optimal solutions for moderate-sized instances in a reasonable amount of time. Evolution-based algorithms always find feasible solutions. In terms of solution quality, BMF heuristic has a significantly better performance comparing with DBLF heuristic and the DE obtains a better solution for some instances compared with GA. It is worthy to note that GA + BMF produces effective solutions, compared with GA + DBLF. With respect to solution time, BMF heuristic takes slightly more time compared with DBLF. One reason is that the BMF needs to test more placement combinations.

5.2.2. Results on randomly generated instances

Due to the lack of benchmarking data for the 3D-BPP-*h* studies, we extend Martello et al. (2000)'s instance generator to fit our study and generate five classes of test data in the same way described in Section 5.1. For each class, we consider instances with a number of boxes equal to 20, 50, 80, 100, 120, 150 and a corresponding number of available containers 10, 20, 25, 30, 35, 40. All containers are generated with each of their dimensions randomly distributed in [50, 200]. For each class and instance size, 10 random instances are solved. Like Table 2, we compare two other heuristic methods with the proposed algorithm. The results are shown in Table 3. The column "Sup_1" reports the number of instances that the corresponding algorithm finds better solutions comparing with the GA + DBLF method. Similarly, "Sup_2" reports the number of instances when the corresponding algorithm finds better solutions comparing with GA + BMF. For most instances, all the algorithms terminate within 1000 s and the instances that are most difficult to solve are from class 5 while they have the highest utilization ratios. It can be observed that the BMF based heuristics significantly outperform the GA + DBLF method. The DE + BMF algorithm produces fairly effective solutions in a reasonable amount of time and in some cases it outperforms the GA + BMF method.

6. Conclusion

In this paper, we investigate a generalized three-dimensional bin packing problem (3D-BPP-*h*) in which the containers are heterogeneous, i.e., varying in size. The objective is to maximize the utilization of the container space. The 3D-BPP-*h* finds its application in many industrial operations, such as cargo transportation, warehouse management, medical packaging, and so on. We formulate the 3D-BPP-*h* as a mixed integer linear programming model. We propose a novel hybrid differential evolution algorithm that is augmented by a best-match-first (BMF) packaging heuristic strategy. A compact packing solution is then generated based on a given box packing sequence (BPS) and a container loading sequence (CLS) to reduce empty maximal spaces. We compare the proposed DE + BMF algorithm with benchmarking methods using real-world industrial instances and randomly generated instances. The results show that the DE + BMF outperforms the existing solution approaches in terms of solution quality.

References

Almeida, A. d., & Figueiredo, M. B. (2010). A particular approach for the three-dimensional packing problem with additional constraints. *Computers & Operations Research, 37*, 1968–1976.

Alvarez-Valdes, R., Parreño, F., & Tamarit, J. (2015). Lower bounds for three-dimensional multiple-bin-size bin packing problems. *OR Spectrum, 37*, 49–74.

Bortfeldt, A., & Mack, D. (2007). A heuristic for the three-dimensional strip packing problem. *European Journal of Operational Research, 183*, 1267–1279.

Che, C. H., Huang, W., Lim, A., & Zhu, W. (2011). The multiple container loading cost minimization problem. *European Journal of Operational Research, 214*, 501–511.

Chen, C., Lee, S., & Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research, 80*, 68–76.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*, 4–31.

Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum, 25*, 45–60.

Faroe, O., Pisinger, D., & Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing, 15*, 267–283.

Fekete, S. P., Schepers, J., & van der Veen, J. C. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Operations Research, 55*, 569–587.

Gonçalves, J. F., & Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics, 145*, 500–510.

He, Y., Wu, Y., & de Souza, R. (2012). A global search framework for practical three-dimensional packing with variable carton orientations. *Computers & Operations Research, 39*, 2395–2414.

Kang, K., Moon, I., & Wang, H. (2012). A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Applied Mathematics and Computation, 219*, 1287–1299.

Karabulut, K., & İnceoğlu, M. M. (2005). A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In *Advances in information systems* (pp. 441–450). Springer.

Lai, K., & Chan, J. W. (1997). Developing a simulated annealing algorithm for the cutting stock problem. *Computers & Industrial Engineering, 32*, 115–127.

Li, X., Zhao, Z., & Zhang, K. (2014). A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. In *Proceedings of the 2014 industrial and systems engineering research conference*.

Liu, S., Tan, W., Xu, Z., & Liu, X. (2014). A tree search algorithm for the container loading problem. *Computers & Industrial Engineering, 75*, 20–30.

Lodi, A., Martello, S., & Vigo, D. (2002). Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research, 141*, 410–420.

Lodi, A., Martello, S., & Vigo, D. (2004). Tspack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research, 131*, 203–213.

Martello, S., Pisinger, D., & Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research, 48*, 256–267.

Martello, S., Pisinger, D., Vigo, D., Boef, E. D., & Korst, J. (2007). Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software (TOMS), 33*, 7.

Padberg, M. (2000). Packing small boxes into a big box. *Mathematical Methods of Operations Research, 52*, 1–21.

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

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.

Takahara, S., & Miyamoto, S. (2005). An evolutionary approach for the multiple container loading problem. In *HIS '05: Proceedings of the fifth international conference on hybrid intelligent systems* (pp. 227–232). IEEE.

Wang, H., & Chen, Y. (2010). A hybrid genetic algorithm for 3d bin packing problems. In *IEEE fifth international conference on bio-inspired computing: Theories and applications (BIC-TA)* (pp. 703–707). IEEE.

Wu, Y., Li, W., Goh, M., & de Souza, R. (2010). Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research, 202*, 347–355.

Zheng, J.-N., Chien, C.-F., & Gen, M. (2015). Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem. *Computers & Industrial Engineering, 89*, 80–87.

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.