2017

Three-Dimensional Container Loading: A Simulated Annealing Approach

Hanan Mostaghimi Ghomi
Universty of Windsor

Bryan Gary St. Amour
University of Windsor

Walid Abdul-Kader
University of Windsor

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 7 (2017) pp. 1290-1304

Three-Dimensional Container Loading: A Simulated Annealing Approach

[1]Hanan Mostaghimi, [2]Bryan St. Amour and [1#]Walid Abdul-Kader

[1]Faculty of Engineering, University of Windsor, Windsor (Ontario), Canada

[2]Department of Computer Science, University of Windsor, Windsor (Ontario), Canada

#ORCID: 0000-0002-2926-5059

Abstract

High utilization of cargo volume is an essential factor in the success of modern enterprises in the market. Although mathematical models have been presented for container loading problems in the literature, there is still a lack of studies that consider practical constraints. In this paper, a Mixed Integer Linear Programming is developed for the problem of packing a subset of rectangular boxes inside a container such that the total value of the packed boxes is maximized while some realistic constraints, such as vertical stability, are considered. The packing is orthogonal, and the boxes can be freely rotated into any of the six orientations. Moreover, a sequence triple-based solution methodology is proposed, simulated annealing is used as modeling technique, and the situation where some boxes are preplaced in the container is investigated. These preplaced boxes represent potential obstacles. Numerical experiments are conducted for containers with and without obstacles. The results show that the simulated annealing approach is successful and can handle large number of packing instances.

Keywords: knapsack, packing sequence, rotation, obstacles, simulated annealing

INTRODUCTION

Logistics has recently played an important role in the success of modern enterprises. Packing boxes inside a container is an essential material handling activity in manufacturing and transportation industries. It is also a key function for operating supply chain efficiently. The efficient use of transportation devices, like containers and palettes, leads to significant cost saving. Moreover, high utilization of transportation devices reduces the traffic of goods and protects natural resources. Therefore, optimal loading of a container decreases the shipping cost and increases the stability of the load. Container loading problem has practical values, and it can be applied to various fields. Loading cars, trucks, trains, or ships can be also considered as a container loading or a three-dimensional packing problem. Furthermore, cargo volume is an important factor used by motor vehicle companies to market their products as sub-compact, compact, midsize, or full size. If inaccurate, the cargo volume found may downgrade the vehicle from its real size. For example, a mid-size may be downgraded to a compact, which results in a potential loss of the market when compared to other competitor's vehicles (Hifi, 2002; Bortfeldt et al., 2003; and Wang et al., 2008).

The basic form of the container loading problem is packing the best subset of rectangular boxes (called cargo) into a large object (called container) to maximize the total value of the loaded boxes. The boxes should not be overlapped and should lie entirely in the container. According to the topology introduced by Wascher et al. (2007), the containers can be either homogeneous or heterogeneous. If the boxes placed in the given container are identical, it is called homogeneous; however, if various types of boxes are loaded, the container is considered as heterogeneous. Besides the non-overlapping constraints, some other practical constraints should be considered in the real-world container loading problem, such as cargo vertical stability, preplaced boxes, and box rotation (Junqueira et al., 2012; Bortfeldt et al., 2012). However, not many papers have considered these practical constraints in their proposed models.

The problem addressed in this research and as per the topology presented by Dyckhoff (1990), belongs to 3/B/O/F (3: three-dimensional, B/O: one object/bin and items selection, F: few items of different types), while Wascher et al. (2007) classify it as the three-dimensional single orthogonal knapsack problem. The given problem considers the packing of rectangular items into a container to maximize the total value of the packed items by minimizing the amount of lost space. The value of boxes is assumed to be equal to their volume. The rotation of the boxes is considered as well. The multidimensional knapsack Problem (MKP) is a NP-hard optimization problem that can be shown by reduction from the one-dimensional packing problem (Egeblad and Pisinger, 2009). Although technological knowledge has been enhanced, solving real knapsack problems is still a challenge. Due to NP-hardness of the packing problem, only heuristic methods, and a few exact algorithms have been presented.

In this paper, a mixed integer linear model and a simulated annealing algorithm are developed to address a more comprehensive knapsack problem where practical considerations, such as vertical stability and preplaced (obstacles) constraints, are tackled. These practical constraints and box rotation contribute significantly to a study of a more

realistic 3D knapsack problem as proposed in this research work. The aim of this proposed research is to provide managers and decision-makers with an adequate modeling tool that helps make shipping goods more efficient and eco-friendly as fewer trips can be made if higher container utilization is achieved. Also, auto-makers can market the class size of their vehicles more accurately.

LITERATURE REVIEW

The focus of most of the container loading or three-dimensional cutting and packing problems is on the rectangular bins. Fekete and Schepers (2004) propose a new method for obtaining lower bound classes for higher-dimensional packing problem. The major objective of this paper is to define good criteria for removing a candidate set of boxes. Dual feasible function is a way to build conservative scales. The computational results are mainly limited to the two-dimensional packing problem. Hifi (2004) proposes a dynamic algorithm and an exact depth-first search to solve the three-dimensional cutting problem. Orientation and guillotine constraints are considered. Optimal solutions are obtained for a significant number of instances, but not all of them. Althaus et al. (2007) consider the trunk packing problem where the box dimensions are as per the SAE J1100 standard. They propose two discretized methods. First, the space to be packed is discretized. Then, an approximation approach is considered using linear inequalities. The space discretization causes insufficient representation of the boxes. Additionally, the runtime of the enumerative algorithms is exponential.

Although considerable advancement has been made in the development of exact algorithms, heuristic algorithms still play an important role in solving three-dimensional knapsack problems. Only heuristic methods can provide reasonable solutions within an acceptable running time for instances of real-world sized problems. Pisinger (2002) develops a wall-building based heuristic. Both homogenous and heterogeneous instances are considered. Moreover, several ranking rules are studied to select the best layers' depths. Bortfeldt et al. (2003) propose a parallel tabu search approach for a single container loading problem and give little consideration to heterogeneous instances. Wang et al. (2008) also present a heuristic method for a heterogeneous container loading problem and developed a dynamic space decomposition approach based on the tertiary tree structure. Egeblad and Pisinger (2009) propose a simulated annealing based methodology for the two and three-dimensional knapsack problems, and a three-dimensional knapsack model is presented. The authors present an iterative heuristic approach for the knapsack problem that is based on the sequence triple representation. Also, Yamazaki et al. (2000) apply a variety of packing sequences including sequence-triple in their 3-D packing solution approach. To control the heuristic method, simulated annealing is used. However, rotation of boxes and preplaced boxes (obstacles) are not considered in the three-dimensional model and experiments. Wu et al. (2010) consider the three-dimensional bin packing problem with variable bin heights. A mixed integer programming model is proposed, and they also present the case when more than one type of bin is used. A genetic algorithm-based heuristic is proposed for packing a batch of objects. Goncalves et al. (2012) propose a multi-population biased random-key genetic algorithm for the single container loading problem. Maximal-space representation is used to manage the container free space. The authors consider stability and orientation constraints; however, they do not develop a mathematical model for the given problem. Peng et al. (2009) present a hybrid simulated annealing algorithm for three-dimensional container loading problem. Firstly, a heuristic algorithm is used for encoding feasible packing solutions, and then the simulated annealing algorithm is applied to search in the encoding neighborhood. Hongtao et al. (2012) address a three-dimensional single container loading problem by using a multi-stage search based simulated annealing algorithm. Wei et al. (2012) use a reference length approach to address the three-dimensional strip packing problem. In another paper, Wei et al. (2015) address the problem of multiple container loading cost minimization problem by using a new approach that combines column generation technique with a goal driven search.

Other research works related to design automation focus on three-dimensional placement of circuit elements by exploring the layout of the integrated circuits (Obenaus and Szymansky, 2003), and Cheng et al. (2005) address floor planning for 3-D VLSI Design. While this technique is less known to container loading practices, it carries some similarities and it is more efficient in terms of search time than other methods such as partitioning placement.

Models that provide information on optimal objective function value and bounds help to assess the solution quality of heuristic algorithms. Although modeling three-dimensional knapsack problems considers practical constraints, it is still at its beginnings. Junqueira et al. (2012) present mixed integer linear programming models for the container loading problem. Vertical and horizontal stability of the cargo, as well as cargo load bearing strength, are considered in the proposed model. However, the models are only able to handle moderate sized problems. Table 1 compares some relevant papers and models, and shows their similarities and differences.

Per the literature, not all papers consider box rotation since it increases the search space significantly. Bin stability constraints have likewise been just considered in a few papers. To the best of our knowledge, preplaced boxes (obstacles) have not been studied in three-dimensional knapsack problems, although it is so essential for such problems since it is often required to place certain boxes in certain positions. These constraints can also be used in the case of having a non-rectangular container. Therefore, it is important to study more practical constraints in the knapsack problem. This proposed research work aims to contribute to the literature so that a 3D knapsack problem can be tackled where box rotation is considered to help finding more practical packing configurations. Furthermore, preplaced boxes (bin with some obstacles) and vertical stability constraints are considered and addressed. This is useful, especially when considering trunk loading for auto size classification as indicated earlier in the Introduction section.

Table 1: Models Parameters Comparison

Paper	Box Rotation	Vertical Stability	Preplaced Boxes	Mathematical Model	Heuristic Approach	Able to solve Large Problems
Junqueira et al. (2012)		√		√		
Goncalves et al. (2012)	√	√			√	√
Wu et al. (2010)	√			√	√	√
Egeblad & Pisinger (2009)	Just 2D			√	√	√
Wang et al. (2008)	√				√	√
Bortfeldt et al. (2003)	√	√			√	√
Pisinger (2002)	√				√	√
Proposed Research	√	√	√	√	√	√

PROBLEM DEFINITION

In this study, the three-dimensional knapsack problem is considered where there is one bin with fixed size and a set of boxes, and each box has an associated size. The aim is to find an efficient solution methodology to pack rectangular boxes in a single bin so that the total value of the packed boxes is maximized, or equivalently the empty spaces left are minimized. The boxes are assumed to be strongly heterogeneous, which means that there is relatively many different types of boxes and a small number of boxes for each box type (Wascher et al., 2007). Moreover, the packing is considered feasible if each box lies entirely in the bin and the packed boxes do not overlap. The edges of all boxes must be parallel to the edges of the bin (orthogonal packing). The boxes are assumed to be of rectangular shape; however, the bin can be considered either of rectangular or nonrectangular shape. In the case of having preplaced boxes (obstacles), the bin is assumed to be non-rectangular.

Some practical considerations that play an important role in modeling more realistic knapsack problems, such as box rotation and bin stability, are presented. The algorithm assumes that the boxes can be freely rotated in six different orientations. However, it is possible to relax this constraint and fix a box in a specific orientation. The boxes need not to be packed in layers, and the bottom of each box must be supported by the top of other boxes or the bin floor. In addition, some boxes whose left-bottom-behind (LBB) corner should be placed in a specific position are considered as preplaced boxes or obstacles. The value of each box is equal to its volume. It is assumed that the dimensions of all boxes and the bin are integers; thus, the placement is to be done in integer steps. Let C be a rectangular bin with width W, height H and depth D. The origin of the Cartesian coordinate system is located at the LBB corner of the container, w_i, h_i, and d_i are respectively, the width, height and depth of box type i. and (x_i, y_i, z_i) represent the coordinates of the LBB corner of the box.

A mixed integer programming formulation is presented for the given problem. Some real-world knapsack problem constraints are considered in the model, and to the best of our knowledge, they have not been addressed previously. These constraints are vertical stability and preplaced boxes (obstacles). Since the three-dimensional knapsack problem is NP-hard, it is difficult to solve. Additionally, the flexibility of the orientation of boxes increases the search space significantly so that the difficulty of finding the optimal solution is increased as well. Some exact algorithms and heuristic methods are proposed in the published literature. As exact algorithms require more time to find a solution, heuristic approaches are more popular and can be effective alternatives to finding an optimal or near optimal solution. The proposed three-dimensional solution methodology is based on sequence triple representation, which is defined below under *Placement Algorithm*. Simulated annealing is used as the meta-heuristic method. As the number of box types (or box dimensions and variety) is finite, the use of simulated annealing is favoured by its efficiency in neighborhood search.

MATHEMATICAL FORMULATION

A mixed-integer programming model of the 3D-knapsack problem is formulated in this section. The mathematical model is based on work done by Egeblad and Pisinger (2009) and Wu et al. (2010). Some modifications are made to their models and include new constraints addressing vertical stability and obstacles, which were not considered in any previously published works. Constraints (1) – (4) are adapted from Egeblad and Pisinger (2009); however, the authors did not consider box orientation in their model. The binary position variables, which show the orientation of the boxes, are integrated in constraints (5) – (17). This makes the model more comprehensive. They are described below in this section.

The following are the main assumptions considered for the mix integer linear model:

1. The boxes are strongly heterogeneous,
2. The boxes must be located orthogonally,
3. The boxes can freely rotate,
4. The box and bin dimensions are assumed to be non-negative integer,
5. The value of a box is equal to its volume, and

6. The X, Y, and Z axes of the bin are shown in the following figure.

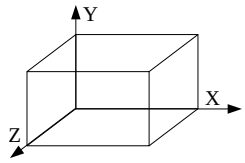


Figure 1: The X, Y, and Z axes of the bin

Notation

The model notation for the parameters and variables are as follows:

- *Variables:*

(x_i,y_i,z_i): Left-Bottom Behind (LBB) coordinates of box i

Xw_i: 1 if width of box i is parallel to the container's X; 0 otherwise

Zw_i: 1 if width of box i is parallel to the container's Z; 0 otherwise

Yh_i: 1 if height of box i is parallel to the container's Y; 0 otherwise

Zd_i: 1 if depth of box i is parallel to the container's Z; 0 Otherwise

r_{ij}, l_{ij}: 1 if box i is to the right (r_{ij}) or to left (l_{ij}) of box j; 0 otherwise

o_{ij}, u_{ij}: 1 if box i is over (o_{ij}) or under (u_{ij}) box j; 0 otherwise

b_{ij}, f_{ij}: 1 if box i is behind (b_{ij}) or in front (f_{ij}) of box j; 0 otherwise

s_i: 1 if box i is packed; 0 otherwise

y_{ij}^a: 1 if $x_j \geq x_i$ (x coordinate of the LBB corner of box j is greater than or equal to x coordinate of the LBB corner of box i); 0 otherwise

x_{ij}^a: 1 if $x_j < x_i'$ (x coordinate of the LBB corner of box j is less than x coordinate of the Right-Bottom-Behind (RBB) corner of box i); 0 otherwise

- *Parameters:*

(w_i, h_i, d_i): width, height, and depth of box i

(W, H, D): width, height, and depth of the bin

(r, s, k): Left-Bottom Behind (LBB) coordinates of the preplaced boxes (obstacles)

(a, b, c, d): binary orientation parameters of the preplaced boxes

P_i: value of box i

M: large number

Pb: set of preplaced boxes

y_{ij}^{b}: 1 if $z_j \geq z_i$ (z coordinate of the LBB corner of box j is greater than or equal to z coordinate of the LBB corner of box i)
0 otherwise

x_{ij}^{b}: 1 if $z_j < z_i'$ (z coordinate of the Left-Bottom-Front (LBB) corner of box j is less than z coordinate of the Left-Bottom-Front (LBF) corner of box i)
0 otherwise

y_{ij}^{c}: 1 if $x_j' > x_i$ (x coordinate of Right-Bottom-Behind (RBB) corner of box j is greater than x coordinate of the LBB corner of box i)
0 otherwise

x_{ij}^{c}: 1 if $x_j' \leq x_i'$ (x coordinate of RBB corner of box j is less than or equal to x coordinate of the RBB corner of box i)
0 otherwise

y_{ij}^{d}: 1 if $z_j' > z_i$ (z coordinate of the LBB corner of box j is greater than z coordinate of the LBF corner of box i)
0 otherwise

x_{ij}^{d}: 1 if $z_j' \leq z_i'$ (z coordinate of the LBF corner of box j is less than or equal to z coordinate of the LBF corner of box i)
0 otherwise

z_{ij}^{a}: 1 if $x_i \leq x_j < x_i'$ (both x_{ij}^{a} and y_{ij}^{a} are equal to one)
0 otherwise

z_{ij}^{b}: 1 if $z_i \leq z_j < z_i'$ (both x_{ij}^{b} and y_{ij}^{b} are equal to one)
0 otherwise

z_{ij}^{c}: 1 if $x_i < x_j' \leq x_i'$ (both x_{ij}^{c} and y_{ij}^{c} are equal to one)
0 otherwise

z_{ij}^{d}: 1 if $z_i < z_j' \leq z_i'$ (both x_{ij}^{d} and y_{ij}^{d} are equal to one)
0 otherwise

Cs_{1ij}: 1 if $x_i \leq x_j < x_i'$ and $z_i \leq z_j < z_i'$ (both z_{ij}^{a} and z_{ij}^{b} are equal to one)
0 otherwise

Cs_{2ij}: 1 if $x_i \leq x_j < x_i'$ and $z_i < z_j' \leq z_i'$ (both z_{ij}^{a} and z_{ij}^{d} are equal to one)
0 otherwise

Cs_{3ij}: 1 if $x_i < x_j' \leq x_i'$ and $z_i \leq z_j < z_i'$ (both z_{ij}^{c} and z_{ij}^{b} are equal to one)
0 otherwise

Cs_{4ij}: 1 if $x_i < x_j' \leq x_i'$ and $z_i < z_j' \leq z_i'$ (both z_{ij}^{c} and z_{ij}^{d} are equal to one)
0 otherwise

$$x_i' = x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i)$$

$$z_i' = z_i + d_i Zd_i + h_i(1 - Zw_i - Zd_i) + w_i Zw_i$$

Objective Function:

The objective function is to maximize the value of the packed boxes: $Max \sum_{i=1}^{n} P_i s_i$

Subject to:

$$r_{ij} + l_{ji} + b_{ij} + f_{ji} + o_{ij} + u_{ji} = s_i + s_j - 1 \quad \forall i, j, i \neq j \tag{1}$$

$$x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i) \leq x_j + M(1 - l_{ij}) \quad \forall i, j, i \neq j \tag{2a}$$

$$x_j + w_j Xw_j + h_j(Zw_j - Yh_j + Zd_j) + d_j(1 - Xw_j - Zw_j + Yh_j - Zd_j) \leq x_i + M(1 - r_{ij}) \quad \forall i, j, i \neq j \tag{2b}$$

$$z_i + d_i Zd_i + h_i(1 - Zw_i - Zd_i) + w_i Zw_i \leq z_j + M(1 - b_{ij}) \quad \forall i, j, i \neq j \tag{2c}$$

$$z_j + d_j Zd_j + h_j(1 - Zw_j - Zd_j) + w_j Zw_j \leq z_i + M(1 - f_{ij}) \quad \forall i, j, i \neq j \tag{2d}$$

$$y_i + h_i Yh_i + w_i(1 - Xw_i - Zw_i) + d_i(Xw_i + Zw_i - Yh_i) \leq y_j + M(1 - u_{ij}) \quad \forall i, j, i \neq j \tag{2e}$$

$$y_j + h_j Yh_j + w_j(1 - Xw_j - Zw_j) + d_j(Xw_j + Zw_j - Yh_j) \leq y_i + M(1 - o_{ij}) \quad \forall i, j, i \neq j \tag{2f}$$

$$x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i) \leq W \tag{3a}$$

$$y_j + h_i Yh_i + w_i(1 - Xw_i - Zw_i) + d_i(Xw_i + Zw_i - Yh_i) \leq H \tag{3b}$$

$$z_i + d_i Zd_i + h_i(1 - Zw_i - Zd_i) + w_i Zw_i \leq D \tag{3c}$$

$$Xw_i + Zw_i \leq 1 \tag{4a}$$

$$Zw_i + Zd_i \leq 1 \tag{4b}$$

$$0 \leq Zw_i - Yh_i + Zd_i \leq 1 \tag{4c}$$

$$0 \leq 1 - Xw_i - Zw_i + Yh_i - Zd_i \leq 1 \tag{4d}$$

$$0 \leq Xw_i + Zw_i - Yh_i \leq 1 \tag{4e}$$

$$(x_i, y_i, z_i) = (r, s, k) \tag{5}$$

$$(Xw_i, Zw_i, Zd_i, Yh_i) = (a, b, c, d) \quad \forall i \in Pb \tag{6}$$

$$x_j - x_i \leq M.y_{ij}^a x_j - x_i \geq M(y_{ij}^a - 1) \quad \forall i, j, \ i \neq j \tag{7a}$$

$$x_i' - x_j \leq M.x_{ij}^a x_i' - x_j \geq M(x_{ij}^a - 1) + 0.5 \quad \forall i, j, \quad i \neq j \tag{7b}$$

$$\frac{y_{ij}^a + x_{ij}^a - 1}{2} \leq z_{ij}^a \leq \frac{y_{ij}^a + x_{ij}^a}{2} \quad \forall i, j, i \neq j \tag{7c}$$

$$z_j - z_i \leq M.y_{ij}^b z_j - z_i \geq M(y_{ij}^b - 1) \quad \forall i, j, i \neq j \tag{8a}$$

$$z_i' - z_j \leq M.x_{ij}^b z_i' - z_j \geq M(x_{ij}^b - 1) + 0.5 \quad \forall i, j, i \neq j \tag{8b}$$

$$\frac{y_{ij}^b + x_{ij}^b - 1}{2} \leq z_{ij}^b \leq \frac{y_{ij}^b + x_{ij}^b}{2} \quad \forall i, j, i \neq j \tag{8c}$$

$$x_j' - x_i \leq M.y_{ij}^c x_j' - x_i \geq M(y_{ij}^c - 1) + 0.5 \quad \forall i, j, i \neq j \tag{9a}$$

$$x_i' - x_j' \leq M.x_{ij}^a x_i' - x_j' \geq M(x_{ij}^c - 1) \quad \forall i, j, i \neq j \tag{9b}$$

$$\frac{y_{ij}^c + x_{ij}^c - 1}{2} \leq z_{ij}^c \leq \frac{y_{ij}^c + x_{ij}^c}{2} \quad \forall i, j, i \neq j \tag{9c}$$

$$z_j' - z_i \leq M.y_{ij}^d z_j' - z_i \geq M(y_{ij}^d - 1) + 0.5 \quad \forall i, j, i \neq j \tag{10a}$$

$$z_i' - z_j' \leq M.x_{ij}^d z_i' - z_j' \geq M(x_{ij}^d - 1) \quad \forall i, j, i \neq j \tag{10b}$$

$$\frac{y_{ij}^d + x_{ij}^d - 1}{2} \leq z_{ij}^d \leq \frac{y_{ij}^d + x_{ij}^d}{2} \quad \forall i, j, i \neq j \tag{10c}$$

$$\frac{z_{ij}^a + z_{ij}^b - 1}{2} \le Cs_{1ij} \le \frac{z_{ij}^a + z_{ij}^b}{2} \qquad \forall i,j,i \ne j \tag{11}$$

$$\frac{z_{ij}^a + z_{ij}^d - 1}{2} \le Cs_{2ij} \le \frac{z_{ij}^a + z_{ij}^d}{2} \qquad \forall i,j,i \ne j \tag{12}$$

$$\frac{z_{ij}^c + z_{ij}^b - 1}{2} \le Cs_{3ij} \le \frac{z_{ij}^c + z_{ij}^b}{2} \qquad \forall i,j,i \ne j \tag{13}$$

$$\frac{z_{ij}^c + z_{ij}^c - 1}{2} \le Cs_{4ij} \le \frac{z_{ij}^c + z_{ij}^d}{2} \qquad \forall i,j,i \ne j \tag{14}$$

$$Cs_{1ij} + Cs_{2ij} + Cs_{3ij} + Cs_{4ij} = u_{ij} + o_{ij} \qquad \forall i,j,i \ne j \tag{15}$$

$$x_i' = x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i) \tag{16}$$

$$z_i' = z_i + d_i Zd_i + h_i(1 - Zw_i - Zd_i) + w_i Zw_i \tag{17}$$

$$r_{ij}, l_{ij}, o_{ij}, u_{ij}, b_{ij}, f_{ij} \in \{0,1\} \tag{18}$$

$$Xw_i, Zw_i, Zd_i, Yh_i \in \{0,1\} \tag{19}$$

$$x_{ij}^a, x_{ij}^b, x_{ij}^c, x_{ij}^d, y_{ij}^a, y_{ij}^b, y_{ij}^c, y_{ij}^d, z_{ij}^a, z_{ij}^b, z_{ij}^c, z_{ij}^d \in \{0,1\} \tag{20}$$

$$s_i, Cs_{1ij}, Cs_{2ij}, Cs_{3ij}, Cs_{4ij} \in \{0,1\} \tag{21}$$

$$(x_i, y_i, z_i) \ge 0 \tag{22}$$

Constraint (1) ensures that if box i and box j are packed, they must be placed left (*l*), right (*r*), under (*u*), over (*o*), behind (*b*) or in-front (*f*) of each other. Constraints (2) guarantee that any two i and j boxes do not overlap, while considering the box rotation. The binary position variables (Xw_i, Zw_i, Yh_i, Zd_i) are used to allow box rotations. Constraint set (3) ensures that all boxes are placed within the bin's dimensions. Constraint set (4) is used to ensure that the binary variables that show the position of the boxes are controlled to represent practical positions. Constraint (5) and (6) are used to fix the coordinates and orientations of the preplaced boxes, where Pb is a set of preplaced boxes. Constraints (7) – (10) ensure vertical stability. These constraints compare the four corners of each newly packed box with the points that cover the top of other packed boxes. If one of the corners has the same x and z coordinates as one of the mapped points, it means that the new box is located under or above that box. Constraint set (7) is used to define the binary variable z_{ij}^a, and it includes three parts. Constraint (7a) ensures that if $x_j \ge x_i$, then y_{ij}^a is equal to one; otherwise it is equal to zero. Constraint (7b) makes sure that if $x_j < x_i$, then x_{ij}^a is one; otherwise it is equal to zero. Constraint (7c) guarantees that when y_{ij}^a and x_{ij}^a are both equal to one, then z_{ij}^a is equal to one. Similarly, constraint sets (8), (9), and (10) are used to define the binary variables z_{ij}^b, z_{ij}^c, and z_{ij}^d. Constraints (11) – (14) show whether the x and z coordinates of the corner of the new box are equal to x and z coordinates of the mapped points on the top of the packed boxes. Constraint (15) ensures that if these coordinates are the same, then the new box should be located on top of or under the packed box. Constraints (16) and (17) define x_i' and z_i'. Finally, constraints (18) – (21) represent the binary variables while constraint (22) represents the integer variables.

The given mathematical model was coded in GAMS/Cplex, and the computational tests were run on an Intel® Core™ i5 CPU @ 2.67GHz processor with 4.0 GB RAM. The model at first was run for an instance with 5 boxes; it reached the optimal solution in 53 seconds. Then the instance with 6 boxes is considered, and the solution time is equal to 6 minutes and 14 seconds. However, the solution time for the instance with 7 boxes increased significantly to 4 hours and 4 minutes. The optimal results for instance with 8 boxes was obtained after 21 hours and 39 minutes. The model was not able to reach optimal solution for instance with 9 boxes even after 3 days; thus, the algorithm was terminated before reaching the solution. According to the results, optimal solutions were only possible in a reasonable time for small size instances (up to 8 boxes).

While this research problem is well-known as NP hard, the mathematical model as presented above has helped define and outline this problem in more details. However, as addressed in the following section, the heuristic algorithm is required to get solutions for larger instances in a reasonable time.

SOLUTION METHODOLOGY – HEURISTIC APPROACH

Three sequences are considered for the boxes to pack. These sequences show the relative box locations. They are known as sequence triple. Sequence triple is one of the most successful representations in the literature and defines the packing order. This section will first discuss placement approach in the subsection. Then, simulated annealing is discussed, which controls the local neighbourhood search. Finally, orthogonal rotation, preplaced boxes (obstacles), four-corner packing, and box insertion order are explained later in this section.

Placement Algorithm

Three sequences A, B, and C represent the fully robot packable packing, where A, B, and C are permutations of the numbers 1 ... n, and n is the total number of boxes to be placed in the bin.

These sequences denote the relative placement of each of the two i and j boxes with respect to each other. Each sequence is defined as follows:

· *A-chain*: If box i appears before box j in the A-chain, then box i is located to the left of, on top of, or in front of box j.

· *B-chain*: if box i appears before box j in the B-chain, then box i is located behind, to the left of, or below, box j.

· *C-chain*: If box i appears before box j in the C-chain, then box i is located to the right, under, or in front of box j.

Based on the three given sequences, it is possible to determine whether box i is located on the left side of, below, or behind box j. It is observed that box i always appears before box j in B-chain for all three given placements. Thus, the order of placement of the boxes in the bin can be based on the B-chain. The first box is placed at the origin, and the succeeding boxes are placed according to their relative position to boxes that are already packed. The coordinates of each new box are calculated based on the following formulae:

$$x_i = max\left(0, max_{j \in P_x}(x_j + w_j)\right)$$

$$y_i = max\left(0, max_{j \in P_y}(y_j + h_j)\right)$$

$$z_i = max\left(0, max_{j \in P_z}(z_j + d_j)\right)$$

where P_x, P_y, and P_z are the subsets of packed boxes located on the left, below, and behind the new box.

To consider vertical stability and reduce the gap between the boxes, some modifications have been applied to calculate the y-coordinate of each packed box. These modifications are explained below.

- ***Vertical Stability***

As it is assumed that the (x, y, z) coordinates of the boxes and their dimensions are integer, it is possible to map a set of points that a certain box covers. Let (x_i, y_i, z_i) be the Left-Bottom Behind (LBB) coordinates of each box to be packed. The algorithm considers four corners of the given box. If the x and z coordinates of one of these corners are equal to the coordinates of one of the points at the top of any packed box, then the height of that box is returned. The y-coordinate of the new box would then be equal to the maximum of those values. The proposed novel approach, which is illustrated below, ensures that the vertical stability is satisfied.

1. Consider (x_i, y_i, z_i)

$\forall j \in P_y$: compute x'_j and z'_j

Where $x_j \le x'_j \le x_j + w_j - 1$ and $z_j \le z'_j \le z_j + d_j - 1$

If ($x_i = x'_j$ and $z_i = z'_j$) then

return $y_j + h_j$

else go to 2

2. Consider ($x_i + w_i$, y_i, z_i)

$\forall j \in P_y$: compute x'_j and z'_j

Where $x_j + 1 \le x'_j \le x_j + w_j$ and $z_j \le z'_j \le z_j + d_j - 1$

If ($x_i + w_i = x'_j$ and $z_i = z'_j$) then

return $y_j + h_j$

else go to 3

3. Consider (x_i, y_i, $z_i + d_i$)

$\forall j \in P_y$: compute x'_j and z'_j

Where $x_j \le x'_j \le x_j + w_j - 1$ and $z_j + 1 \le z'_j \le z_j + d_j$

If ($x_i = x'_j$ and $z_i + d_i = z'_j$)

return $y_j + h_j$

else go to 4

4. Consider ($x_i + w_i$, y_i, $z_i + d_i$)

$\forall j \in P_y$: compute x'_j and z'_j

Where $x_j + 1 \le x'_j \le x_j + w_j$ and $z_j + 1 \le z'_j \le z_j + w_j$

If ($x_i + w_i = x'_j$ and $z_i + d_j = z'_j$) then

return $y_j + h_j$

else return 0

return $y_i = max(0, (y_j + h_j))$

The algorithm pushes each packed box downward where possible such that its bottom can be supported by the floor of the bin or by the top of other packed boxes.

Simulated annealing

Although it is relatively not difficult to develop a simulated annealing heuristic approach, choosing a good neighborhood and cooling procedure, which itself depends on several different parameters, is usually necessary for the algorithm to work efficiently. The cooling procedure is different for various types of problems, and even between instances of the same problem. Therefore, it is difficult to find out a good cooling procedure. In the proposed simulated annealing algorithm, the temperature is reduced when a new solution is accepted, according to the following function:

$$t \to t/(1 + \beta t)$$

where β is the cooling parameter. Besides the cooling down procedure, the process is allowed to heat up again whenever it appears to be getting trapped. The heating up function is:

$$t \to t/(1 - \alpha t)$$

where α is the heating parameter. The temperature is reduced when the solution is accepted and increased when the solution is rejected. Parameter α must be smaller than β as the number of acceptances is small when compared to the number of rejections (Dowsland, 1993).

The neighbourhood of each solution is defined as one of these five permutations: either exchange two boxes from one of the sequences; exchange two boxes in sequences A and B; exchange two boxes in sequences A and C; exchange two boxes in sequences C and B; or exchange two boxes in all sequences. The solutions are compared based on the bin utilization. The formula used for calculating the utilization percentage is as follows:

$$utilization\ percentage = \frac{total\ volume\ of packed\ boxes}{volume\ of\ bin} \times 100$$

It should be noted that the volume (or internal volume) of the bin is a reference to an ideal solution that may not be attained due to the discrete nature of the boxes being loaded. However, this volume is used in the relation above to check how far the solution obtained by the simulated annealing algorithm is from the ideal solution.

Orthogonal Rotation

An extension we add to the typical A, B, C-chain representation of the packing is to allow for the boxes to be rotated orthogonally with respect to the bin. Width, height, and depth of all boxes are respectively parallel to x, y, and z axis, and w_i, h_i, and d_i represents the width, height, and depth of box i, respectively. It is possible to obtain better packing if the boxes are rotated in different directions. Boxes are allowed to be rotated in one of the following orientations:

WHD: Standard orientation.

WDH: Swap the height and the depth.

HWD: Swap the width and the height.

HDW: Swap the width and the height, and then swap the height with the depth.

DHW: Swap the depth with the width.

DWH: Swap the depth with the width, and then swap the depth with the height.

The given rotation is applied to the simulated annealing by adding an additional transformation to the neighbourhood generating routine. The orientation of the boxes is generated randomly at first. Thus, an additional vector R, which shows the orientation of the boxes, is stored. The sequence triple is stored as well.

Obstacles

If Pb is the set of rectangular obstacles (or preplaced boxes) with known coordinates (x, y, z) and known dimensions (w, h, d), then the obstacles are fixed into the bin at the beginning of the algorithm. The packing is created from the sequence triple (A, B, C) and those boxes that overlap with any obstacles in the set are removed. The bin free volume is calculated as follows:

Bin free volume = volume of bin – total volume of obstacles

Four-corner packing

Four packing schemes, one for each corner, are created. First, the coordinates of the boxes are calculated in relation to the current origin. Then, their real (x, y, z) coordinates are calculated in relation to the real origin of the bin which is its LBB corner. The processing technique is as follows:

W := bin width

H := bin height

D := bin depth

w := box width

h := box height

d := box depth

if (loading from front) then

// No change needed: this is the default loading method.

Return <x,y,z>

else if (loading from rear) then

return <W – x – w, y, D – z – d>

else if (loading from left side) then

return <W – z – w, y, x>

else if (loading from right side) then

return <z, y, D – x – w>

end

Order of box insertion

The order the boxes are inserted into the container is based on sequence B and can either be created randomly, or based on the volume of the boxes, which means that the boxes with larger volumes are packed first (in a first fit decreasing order).

NUMERICAL EXPERIMENTS

The proposed methodology is implemented in C++, and the code is tested using two different sets of boxes. The dimensions of the first set of boxes are taken from the SAE J1100 Standard, which includes 7 different types of boxes (after excluding the golf bag). The dimensions of these boxes are illustrated in Table 2a. Twelve instances are created by using the first set of boxes. These instances contain 36 and 70 boxes. The maximum allowed number of boxes for both types of instances is also shown in Table 2a.

The second set of boxes is generated randomly based on uniform distribution and includes 10 types of boxes. Two instances are created by using this set, which includes 50 boxes. The width, height, and depth of these boxes are selected from the intervals [100, 250], [50, 250], [100, 300] respectively. The

dimensions of the boxes and their maximum allowed number are shown in Table 2b. In cases where preplaced boxes are not considered, the dimensions of the bin for instances containing 36 boxes are equal to 800×700×1000 (mm). However, for instances with 70 boxes, they are equal to 1100×900×1400 (mm), and in the case of having instances with 50 boxes, the dimensions are equal to 600×500×700 (mm). When obstacles are present, the bin dimensions are equal to 1350×540×890 (mm) in instances with 36 boxes, and they are equivalent to 1100×900×1400 (mm) in other instances. The profits of the boxes are set to be equal to their volume.

Table 2a: Information on the First Set of Boxes

Box Type	Width (mm)	Height (mm)	Depth (mm)	Max. no. of boxes in instances with a total of 36 boxes	Max. no. of boxes in instances with a total of 70 boxes
1	229	483	610	4	7
2	165	330	457	4	7
3	229	406	660	2	5
4	216	457	533	2	5
5	203	229	381	2	5
6	178	356	533	2	6
7	152	114	325	20	35

Table 2b: Information on the Second Set of Boxes (Dimensions generated randomly)

Box Type	Width (mm)	Height (mm)	Depth (mm)	Max. no. of boxes
1	138	182	285	6
2	126	240	135	5
3	108	222	165	4
4	140	80	246	5
5	105	234	272	3
6	153	237	159	6
7	216	229	272	6
8	188	124	236	5
9	137	100	167	4
10	103	104	222	6

The names of the instances are *Knp-n-o-c-v*, where $n \in \{36,70,50\}$ is the number of boxes to be packed; *o* is the order of boxes in B-chain that can be based on the boxes volume (v), i.e. in first fit decreasing order, or randomly created (R); *c* shows whether (or not) the obstacles are considered and can be set as (obs) or (wo) respectively; and *v* represents the volume of the bin.

The number and dimensions of the obstacles (preplaced boxes) differ in various instances. Eight obstacles are defined for cases with 36 and 70 boxes. The dimensions of the obstacles and their coordinates are described in Table 3. For the instances where there are 70 boxes, four obstacles are defined in case of ceiling obstacles, and two obstacles are defined for middle ones. The dimensions and coordinates of these obstacles are illustrated in Table 4.

Table 3: Obstacles Dimensions and Coordinates for Instances with 36 and 70 Boxes

Obstacle dimensions (w, h, d) (mm)	Obstacle coordinates Instance of 36 boxes	Obstacle coordinates Instance of 70 boxes
(180, 220, 250)	(1170, 0, 160)	(920, 0, 160)
(320, 220,160)	(0, 0, 0)	(0, 0, 0)
(320, 220,160)	(1030, 0, 0)	(780, 0, 0)
(125, 220,160}	(0, 0, 160)	(0, 0, 160)
(200, 320, 320)	(0, 220, 0)	(0, 580, 0)
(200, 320, 320)	(1150, 220, 0)	(900, 580, 0)
(160, 208, 240)	(0, 332, 320)	(0, 692, 320)
(160, 208, 240)	(1190, 332, 320)	(940, 692, 320)

Table 4: Information on Ceiling and Middle Obstacles

Ceiling Obstacles		Middle Obstacles	
Dimensions (mm)	**Coordinates**	**Dimensions (mm)**	**Coordinates**
(200, 320, 320)	(0, 580, 0)	(500, 220, 160)	(300, 300, 0)
(200, 320, 320)	(900, 580, 0)	(500, 220, 160)	(300, 300, 1240)
(160, 208, 240)	(0, 692, 320)		
(160, 208, 240)	(940, 692, 320)		

Parameter Setting

Choosing a suitable cooling procedure and parameters is essential for the algorithm to work efficiently. After testing different cooling procedures, the one proposed by Dowsland (1993) works the best. The given cooling process has been explained earlier (see ***Simulated Annealing***). The cooling parameter β is selected to be 0.2, and α is equal to 0.002. Values for initial temperature are selected from {0.5, 0.4, 0.3, 0.2}. Based on the results, t_0=0.2 is the most suitable temperature.

Results and Sensitivity Analysis

Ten runs were conducted for each case. The worst, average, and best solutions are shown in Table 5. The values in the table illustrate the utilization percentage of the bin. In addition, the column "time" represents the run time for each case in minutes.

Table 5: Worst, Best, and Average Utilization

Case	Time (min)	Best (%)	Average (%)	Worst (%)
Knp-36-v-wo-560	10	88.49	86.19	83.92
	20	87.72	85.29	80.45
	30	88.08	86.23	83.43
	120	88.07	85.83	84.81
Knp-36-R-wo-560	10	83.51	80.83	77.31
	20	88.43	85.00	78.26
	30	86.51	83.65	80.19
	120	87.93	87.05	84.81
Knp-36-v-obs-649	10	76.42	74.54	70.76
	20	80.60	78.5	75.63
	30	81.06	79.55	77.64
	120	79.10	77.33	75.13
Knp-36-R-obs-649	10	82.23	79.15	77.14
	20	82.80	80.03	77.50
	30	80.77	79.22	77.58
	120	80.79	78.88	77.21
Knp-70-v-wo-1386	20	86.34	84.33	82.02
	30	85.99	84.24	82.17
	60	86.29	84.56	82.68
	120	86.44	84.96	82.71

Table 5: (Continued) Worst, Best, and Average Utilization

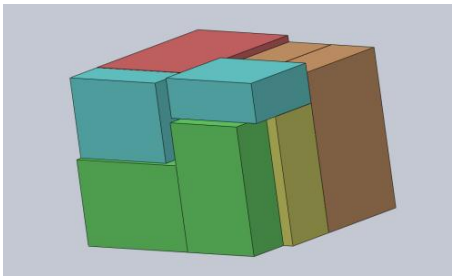
Case	Time (min)	Best (%)	Average (%)	Worst (%)
Knp-70-R-wo-1386	20	84.13	80.92	77.27
	30	84.80	83.39	82.49
	60	84.61	81.89	81.64
	120	85.59	83.59	79.57
Knp-70-v-obs-1386	30	79.74	77.24	75.73
	60	82.09	79.14	75.53
	120	80.12	78.93	76.84
Knp-70-R-obs-1386	30	78.12	75.59	75.06
	60	80.24	78.01	76.50
	120	83.66	79.67	78.34
Knp-70-v-obs[1]-1386	30	85.97	84.37	82.88
	60	85.05	83.30	82.06
	120	82.70	81.74	80.18
Knp-70-R-obs[1]-1386	30	82.31	80.68	78.39
	60	82.66	79.75	77.26
	120	83.09	80.09	78.65
Knp-70-v-obs[2]-1386	30	79.29	77.66	76.66
	60	78.97	78.46	77.74
	120	79.86	77.80	76.15
Knp-70-R-obs[2]-1386	30	79.74	77.89	76.00
	60	78.96	77.35	76.45
	120	82.50	78.75	76.15
Knp-50[3]-v-wo-210	20	85.49	84.02	82.95
	30	88.58	86.45	84.39
	60	86.56	85.36	83.97
	120	89.68	87.58	85.91
	180	88.31	87.02	85.93
Knp-50[3]-R-wo-210	20	86.79	84.70	82.87
	30	86.41	84.89	83.56
	60	88.07	85.53	84.20
	120	89.72	87.42	85.83
	180	88.06	86.55	85.56

For 36-box instances *(Knp-36-o-c-v*), the minimum running time was set to 10 minutes. Although the heuristic approach often reached the best solution in less than 10 minutes, the running time was increased to see whether the algorithm can jump out of the local optimal and find a better solution. Thus, the instances were run for 20, 30, and 120 minutes as well. Based on the results, increasing time does not significantly affect the solutions. It can be concluded that 10 minutes is sufficient for the heuristic approach to find the final solution.

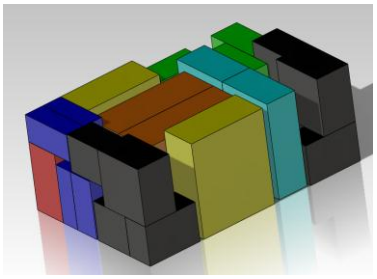
For scenarios that contain 70 boxes and where preplaced boxes are neglected, the algorithm was run for at least 20 minutes. The running time was increased to 30, 60, and 120 minutes. The results indicate that 20 minutes is sufficient to reach a good solution in these scenarios. However, when considering obstacles, the algorithm was tested for at least 30 minutes. This is because dealing with the obstacles increases the solution time. The running time was increased to 60 and 120 minutes. The results show that increasing the running time to 60 minutes allows the algorithm to reach better solutions; however, increasing the running time to 120 minutes does not improve the utilization significantly. Therefore, 60 minutes can be a sufficient running time to reach the final solution. In these cases, according to the results, when including ceiling

[1]*Ceiling obstacles*
[2]*Middle obstacles*
[3]*Second set of boxes*

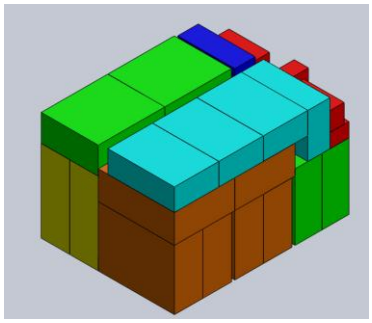
obstacles, the reasonable run time is equal to 30 minutes since handling the ceiling obstacles is easier than floor obstacles. When middle obstacles are present, the bin utilization is less than the one in other instances. These kinds of instances were run for 30, 60, and 120 minutes. Based on the obtained utilizations shown in Table 5, 30 minutes can be considered as a reasonable run time. In the case of *Knp-70-R-obs (middle)-1386*, the algorithm jumps out of the local minimum after 120 minutes and can obtain a better solution (higher bin utilization). Nevertheless, only the best utilization is improved, and the average and worst results do not change significantly. Moreover, the instances in which 50 boxes should be packed were run for 20, 30, 60, 120, and 180 minutes; 30 minutes is observed to be enough if it is required to obtain a satisfying solution in a short time. However, it seems that the algorithm can jump out of the local optimal and find a better solution after 120 minutes. In addition, one of the instances was run for 48 hours to find out if the algorithm can jump out the local optimal after a long time. Based on the result, the gained utilization does not change significantly. Thus, the results in Table 5 can be considered as a reference to make a conclusion. As illustrated in Table 5, the best utilization is obtained in most instances when the order of the boxes in B-chain is based on their volume, where the boxes are fit in decreasing order. Figure 2 illustrates the best results for some of the instances tested.



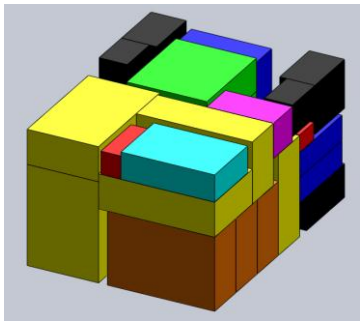
Knp-36-v-wo-560



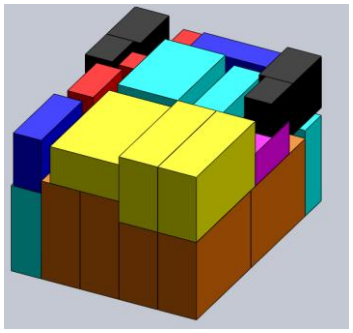
Knp-36-R-obs-649



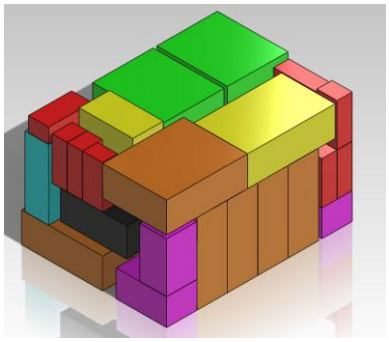
Knp-70-v-wo-1386



Knp-70-v-obs-1386

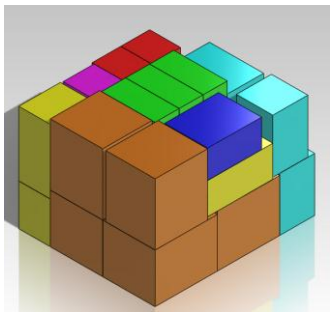


Knp-70-v-obs(ceiling)-1386

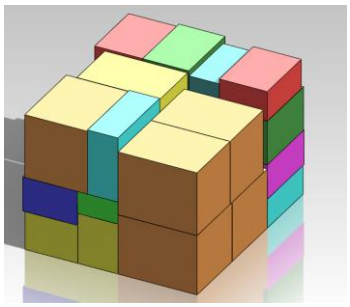


Knp-70-v-obs(middle)-1386

Figure 2: Best results for some instances



Knp-50-v-wo-210



Knp-50-R-wo-210

Figure 2: (Continued): Best results for some instances

For the instances that include obstacles, preplaced boxes are shown in black. As shown in Figure 2, the vertical stability is satisfied for all instances and there is no longer a box placed in the air. The bottoms of all packed boxes are placed either on the bin floor, or on the top of other packed boxes.

Algorithm Verification

To verify the proposed methodology, the *Knp-36-R-obs-649* scenario is run without considering vertical stability constraint; the best, worst and average results obtained in this case are respectively equal to 77.38%, 75.19%, and 76.2%, which are less than the utilizations obtained by considering the vertical stability constraint (82.23%, 77.14%, and 79.15%). As shown in the Figure 3, some of the boxes are placed in the air.

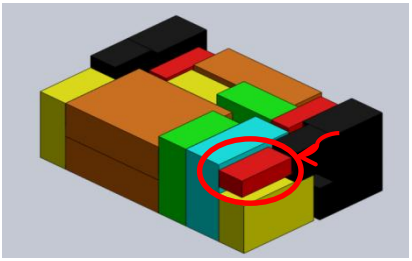


Figure 3: Result without Vertical Stability

CONCLUSIONS

A three-dimensional knapsack problem for container loading has been presented and discussed. The packing is considered orthogonal; boxes are rectangular and can be freely rotated. A mixed integer linear programming model has been proposed for the problem, and it has considered some practical constraints, such as box rotation, vertical stability, and preplaced boxes. The mathematical model, while limited due to the NP hardness of the problem, provided detailed information, and explained all the features considered in this bin packing scheme. To solve large instances in a reasonable time, a heuristic algorithm has been proposed based on the simulated annealing technique. The methodology is based on the sequence triple representation.

Various experiments have been conducted with different sets of boxes. The order of box insertion in the bin can be random or based on the box volume in a decreasing order. The solutions have been compared based on bin utilization. Sensitivity analysis has been conducted based on the running time to determine whether the algorithm can jump out of the local optimal by increasing time to reach a better solution.

The results illustrate that the proposed algorithm performs as intended. Good quality results can be obtained for large instances in a reasonable time. The algorithm can handle various instances and get satisfactory utilizations. According to the final results, better solutions can be obtained if the order of inserting boxes in the bin is based on the volume of the boxes that is in decreasing order. Moreover, the results show that the proposed approach is compatible with preplaced boxes or obstacles, and vertical stability issue is satisfied as well. In addition, the methodology can be used to deal with irregular bins where the bin is not rectangular by considering the irregular sections of the bin as preplaced boxes. The proposed approach can be considered to find a high utilization of the container, which decreases the transportation cost and goods traffic while increasing the stability of the load. This has managerial implications and helps decision-makers to be more cost-efficient.

For future research, boxes with non-integer dimensions can be considered. This adds more flexibility to the solution approach. In addition, non-rectangular and irregular shape boxes can be considered in the future.

ACKNOWLEDGEMENT:

The authors are grateful to the anonymous reviewers for their constructive comments that improved positively the manuscript. Special thanks go to Dr. Ron Barron, University of Windsor, for his help and guidance during the write-up of the research proposal from which this manuscript has emanated. The authors would like to acknowledge the financial support received from the Natural Sciences and Engineering Research Council of Canada (NSERC), Engage Grants Program, Application ID: EGP 424284 – 11, during the tenure of this R&D project.

REFERENCES

[1] Althaus, E., Baumann, T., Schomer E. and Werth, K., 2007, 'Trunk packing revisited,' *Experimental Algorithm, Lecture Notes in Computer Science,* vol. 4525, pp. 420-432.

[2] Bortfeldt, A. and Wascher, G., 2012, 'Container Loading Problems – A state-of-the-art-review,' *FEMM Working Papers from Otto-von-Guericke University Magdeburg, Faculty of Economics and Management*, no. 120007.

[3] Bortfeldt, A., Gehring, H. and Mack, D., 2003, 'A parallel tabu search algorithm for solving the container loading problem,' *Parallel computing in logistics*, vol. 29, no. 5, pp. 641-662.

[4] Cheng, L., Deng, L., and Wong, M.D.F 2005, 'Floor planning for 3D VLSI Design,' Proceedings of the 2005 Asia and South Pacific Design Automation Conference, ACM, pp. 405-411, New York, USA, DOI 10.1145/1120725.1120899.

[5] Dowsland, K.A., 1993, 'Some experiments with simulated annealing techniques for packing problems,' *European Journal of Operational Research*, vol. 68, no. 3, pp. 389-399.

[6] Dyckhoff, H., 1990, 'A typology of cutting and packing problems,' *European Journal of Operational Research*, vol. 44, pp. 145-159.

[7] Egeblad, J., Garavelli, C., Lisi, S. and Pisinger, D., 2010, 'Heuristic for container loading of furniture,' *European Journal of Operational Research*, vol. 12, pp. 881-892.

[8] Egeblad, J. and Pisinger, D., 2009, 'Heuristic approaches for two- and three-dimensional knapsack packing problem,' *Computer & Operations Research,* vol. 36, pp. 1026-1049.

[9] Fekete, S.P. and Schepers J., 2004, 'A general framework for bounds for higher-dimensional orthogonal packing problems,' *Mathematical Methods of Operation Research,* vol. 60, no. 2, pp. 311-329.

[10] Goncalves, J.F. and Resende, M.G.C, 2012, 'A parallel multi-population biased random-key genetic algorithm for a container loading problem,' *Computers & Operations Research,* vol. 39, pp.179-190.

[11] Hifi, M., 2004, 'Exact algorithms for unconstrained three-dimensional cutting problems,' *Computers & Operations Research,* vol. 31, no. 5, pp. 657-674.

[12] Hifi, M. 2002, 'Approximate algorithm for the container loading problem,' *Intl. Trans in Op. Res.*, vol. 9, pp. 747-774.

[13] Hongtao, W., Zhoujing, W., and Jian L., 2012, 'A simulated annealing algorithm for single container loading problem, 9th International Conference on Service Systems and Service Management (ICSSSM), pp. 551-556, July 2-4, 2012, DOI: 10.1109/ICSSSM.2012.6252298, published by IEEE.

[14] Junqueira, L., Morabito, R. and Yamashita D.S., 2012, 'Three-dimensional container loading models with cargo stability and load bearing constraints,' *Computers & Operations Research,* vol. 39, pp. 74-85.

[15] Obenaus, S. T., and Szymanski, T. H., 2003, 'Gravity: Fast Placement for 3-D VLSI,' ACM Transactions on Design Automation of Electronic Systems, vol. 8, no. 3, pp. 298–315.

[16] Peng, Y., and Zhang, D.F. 2009, 'A hybrid simulated annealing algorithm for container loading problem,' China Academic Journal, Electronic Publishing House, vol. 5, no. 2, pp. 124-134.

[17] Pisinger, D., 2007, 'Denser packing obtained in $O(n \log \log n)$,' *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 395-406.

[18] Pisinger, D., 2002, 'Heuristics for the container loading problem,' *European Journal of Operational Research,* vol. 141, no. 2, pp. 382-392.

[19] Wang, Zh., Li, K.W. and Levy J.K., 2008, 'A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach,' *European Journal of Operational Research,* vol. 191, no. 1, pp. 86-99.

[20] Wascher, G., Hauβner, H. and Schumann, H., 2007, 'An improved typology of cutting and packing problems,' *European Journal of Operational Research,* vol. 183, pp. 1109-1130.

[21] Wei, L., Oon, W.-C., Zhu, W., and Lim, A., 2012, 'A reference length approach for the 3D strip packing problem,' European Journal of Operational Research, vol. 220, no. 1, pp. 37-47.

[22] Wei, L., Zhu, W., and Lim, A., 2015, 'A goal-driven prototype column generation strategy for the multiple container loading cost minimization problem,' European Journal of Operational Research, vol. 241, no. 1, pp. 39-49.

[23] Wu, Y., Li, W., Goh, M. and de Souza, R., 2010, 'Three-dimensional bin packing problem with Variable Bin Height,' *European Journal of Operational Research*, vol. 202, no. 2, pp. 347-355.

[24] Yamazaki, H., Sakanushi, K., Nakatake, S., and Kajitani, Y., 2000, 'The 3D-Packing by meta data structure and packing heuristics,' IEICE Transactions, vol. E83-A, no. 4, April.