ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

THREE-DIMENSIONAL BIN PACKING PROBLEM WITH A STABILITY REJECTION CRITERION

Teemu Linkosaari[1,2*], Tero Urponen[2], Henrik Juvonen[2], Marko M. Mäkelä[1] and Yury Nikulin[1]

[1]Department of Mathematics and Statistics
20014, University of Turku
{teemu.linkosaari,marko.makela,yury.nikulin}@utu.fi

[2] Kine Robot Solutions
Vallihaudankatu 10, Turku, Finland
{tero.urponen,henrik.juvonen}@kine.fi

Keywords: packing, stability, three-dimensional, genetic algorithm, global search framework

Abstract. *Packing problems play an important role in transportation and supply chain management. This study aims to solve a practical three-dimensional bin packing problem, where highly heterogeneous sized boxes are to be packed on a pallet. Without supporting walls it is critical to ensure that the boxes are supported and stay stable. Thus, there are two goals: solution compactness and stability. This problem is a part of the task of minimizing the number of pallets needed.*

First, we use an existing bin packing algorithm based on packing indices, which converts an arbitrary list of boxes to a packing solution. The method is extended to be able to pack boxes from four corners instead of one corner. Moreover, a special method to check the stability of placed boxes is devised. To make the search space a bit smaller, grouping of boxes with same dimensions is used.

Secondly, a genetic algorithm is used to find a good permutation to the packing procedure. To further improve the solution quality, we utilize an existing global search framework with the concept of evolutionary gradient.

The efficiency of proposed method is tested and verified in an industrial setting. Two orientations are allowed: boxes are rotated around the vertical axis only. Volume utilization is maximized under the rejection stability criterion, i.e. we discard solutions if any packed box happens to be unstable. Solutions obtained are more stable and packable than in the previous studies.

1 INTRODUCTION

Packing boxes onto a pallet is an important task in the distribution industry. In order to minimize delivery costs, each pallet should be packed efficienlty and safely. This task belongs to the family of packing and cutting problems. In its basic form, the three-dimensional bin packing problem consists of finding the best three-dimensional packing pattern for a set of rectangular items onto a bin such that the volume utilization is maximized [8]. The bin dimensions are fixed and a subset of boxes are selected. In an other variant, a large number of boxes are packed into identical bins with the objective to minimize the number of bins used [6]. It is well-known this type of problem is NP-hard. Thus, exact methods work only for small problems and heuristics are needed for larger problems.

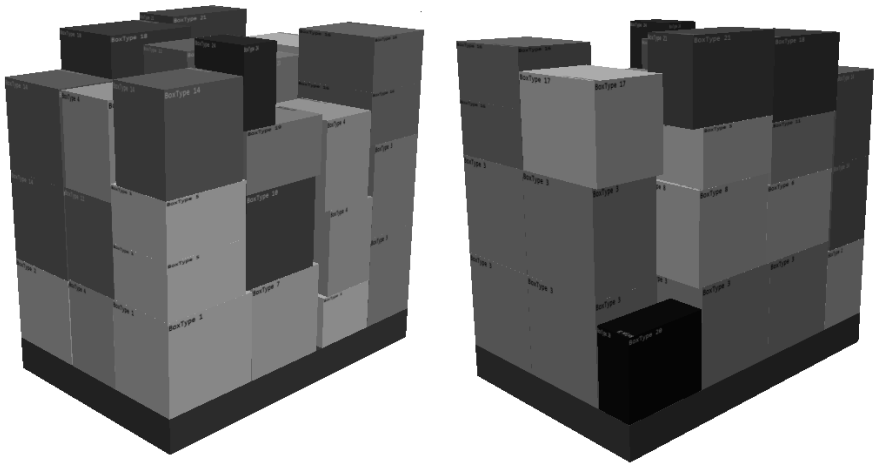

Figure 1: Stable four corner

This paper addresses the distributors's pallet packing problem: pack heterogenous set of boxes to the rectangular pallet without supporting walls, also studied by Schuster et al. [13]. A problem contains n different types of rectangular boxes. For each type i, the dimensions $l_i \times w_i \times h_i$ and the number of boxes n_i are known. The dimensions of boxes are integer (mm). It is assumed that boxes are placed orthogonally to the edges of a pallet and that they can be sideways rotated by $90°$. The goal is to pack all these boxes on a pallet of a size $L \times W$. Pallet has also a physical height limit H_{max}.

Load stability is often considered as one of the most important issues beyond volume utilization [3]. With respect to load stability, one may distinguish between vertical and horizontal stability and load bearing strength.

Vertical stability or static stability prevents boxes from falling down onto the pallet floor or on top of other boxes. Vertical stability issues are usually approached by demanding that the base of a box must be supported in total or partially by either the pallet floor or by an even space provided by the top surfaces of other boxes. It may be demanded that the center of gravity of each box must be supported by the top surface of another box or the pallet floor [9]. In our case, we also consider the center of gravity formed by the layers beneath.

Horizontal stability or dynamic stability assures that boxes cannot shift significantly while the pallet is being moved [2]. It refers to the capacity of the boxes to withstand the intertia of their bodies [8]. In pallet loading in particular, shrinking foil is being used in order to prevent

loads from falling apart [3]. Usually rigid body simulations are used to determine the dynamic stability of packing structure, see for example [12]. This iterative method is computationally expensive, since several iterations are needed to get the results. Hence, it is not directly suitable to be tested inside the packing procedure.

Load bearing strength refers to the maximum number of boxes that can be stacked one above each other, or more generally, to the maximum pressure that can be applied over the top face of a box, so as to avoid box damaging [8].

In this work, we concentrate on the vertical stability only. It is the most revelant to our work and a starting point. The other aspects can be studied in future.

Three-dimensional bin packing problems (3D-BPP) are well studied in literature. For a more comprehensive recent review, see Bortfeldt and Wäscher [3]. The solution techniques for packing problems can be classified into three types: exact methods, heuristics/approximate algorithms and meta-heuristics [6].

Exact methods are based on mathematical programming and graph theory. Chen et al. [4] provided a mixed integer linear programming (MILP) model to solve the 3D-BPP with orientation, which can solve small problems to optimality. Wu et al. [14] adopted the MILP model presented in Chen et al. [4] and coded in GAMS, also solving small instances. Martello et al. [10] proposed a method that can solve moderate instances for the general 3D-BPP and its robot-packable variant. Junqueira et al. [8] presented MILP models for the container loading problem that consider the vertacal and horizontal stability of the cargo and the load bearing strength. Only problems of a moderate size can be handled. At the moment it is almost impossible to find global optimum for practical problems with exact methods.

Most heuristics and approximation algorithms are based on the greedy wall- or layer-building techniques, which easily end up local optimum. For example, Huang and He [7] proposed a caving degree method, which tries to pack a container from eight different corners. It has performed well with known benchmark cases.

Meta-heuristics are the present best choice. Widely used methods are genetic algorithms, tabu search and simulated annealing. Wu et al. [14] provided a genetic algorithm for packing problem. He et al. [6] proposed a packing method, genetic algorithm and global search framework. These papers are the basis of our work.

First, we use an existing bin packing algorithm [6] based on packing indices, which converts an arbitrary list of boxes to a packing solution. The method is extended to be able to pack boxes from four corners instead of one corner. Moreover, a special method to check the stability of placed boxes is devised. To make the search space a bit smaller, grouping of boxes with same dimensions is used.

Secondly, a genetic algorithm [6] is used to find a good permutation to the packing procedure. To further improve the solution quality, we utilize an existing global search framework with the concept of evolutionary gradient.

The efficiency of proposed method is tested and verified in an industrial setting. Two orientations are allowed: boxes are rotated around the vertical axis only. Volume utilization is maximized under the rejection stability criterion, i.e. we discard solutions if any packed box happens to be unstable. Solutions obtained are more stable and packable than in the previous studies.

This work is organized as follows: Section 2 presents the stability rejection criterion. Two packing heuristics are described in section 3. Section 4 introduce the genetic algorithm and the global search framework. Numerical experiments are presented in section 5. Section 6 concludes the paper.

2 STABILITY REJECTION CRITERION METHOD

When boxes are palletized with robots, we must ensure that every single insertion of a box must stay stable. It is also important that the method must not be computationally too expensive, since this method is used every time when we try to place a box to a candidate placement. Thus, we propose a simple geometrical method to determine whether a placement is stable or not. This method is inspired by [1, 11, 15]. We use the Cartesian coordinate system to use for our fixed global coordinates. We define x as the length axis, z as the width axis and y as the height axis. In our method, stability includes three cases: maximum overhang, edge support and the center of gravity for every layer of boxes.

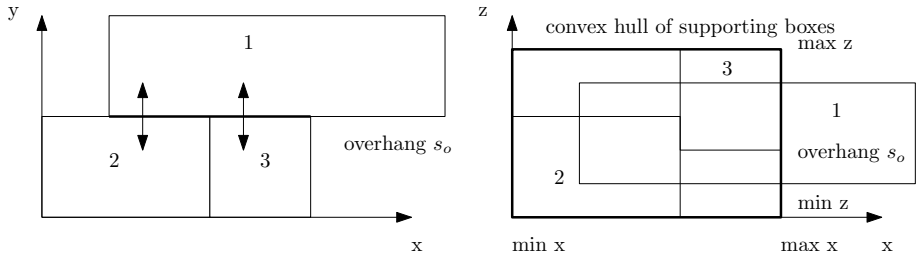

Figure 2: Overhang

First, the concept of *overhang* measures how far off the edge of a box can reach the top of other boxes. From experimental view point, we have found that minimizing overhang between box and its supporting boxes reduces instability. Determining overhang between two boxes is straingforward: we take the largest overhang from x- or z-axis. If the box is supported by more than one box, we take the convex hull of the supporting boxes and measure the maximum overhang between the box and the convex hull. For example in figure 2, box number one is supported by boxes numbered two and three. In x-axis, box one is over the convex hull by amount s_o. When we insert a box, we need only to consider the directly supporting boxes, since all the other boxes are checked in the previous iteration.

Secondly, the concept of *edge contact support* is devised. In some cases, the contact area between two boxes is too small and too close to the edges. When boxes are packed one above other, the pressure may deform the boxes below. Thus, we need to ensure that for every box, the contact area near the edge is sufficient. Let s_e be the amount of edge support required. For a box i, the contact surface area is defined by a rectangle $(x_i + s_e, x_i + wn_i - s_e, z_i + s_e, z_i + ln_i - s_e)$, where x_i and z_i are the locations and wn_i and ln_i are the oriented dimensions of box. When a box is placed to a candidate location, the supporting boxes are those that have a contact area between this smaller rectangle. So if there is not enough contact area, the placement is unstable. For example, in figure 3, box one is packed on top of boxes from 2 to 5. For box 3, the common edge contact support is less than s_e. Thus, it is not supporting box one.

Thirdly, we describe the method that checks the center of gravity of every layer of boxes. We assume that all the boxes are *rectangular rigid bodies*. In general, this is not true, since under pressure the boxes may deform producing inclined towers of boxes. A rigid body is said to be in *equilibrium* if the sum of the forces acting on it, and sum of the *moments* they apply on it,

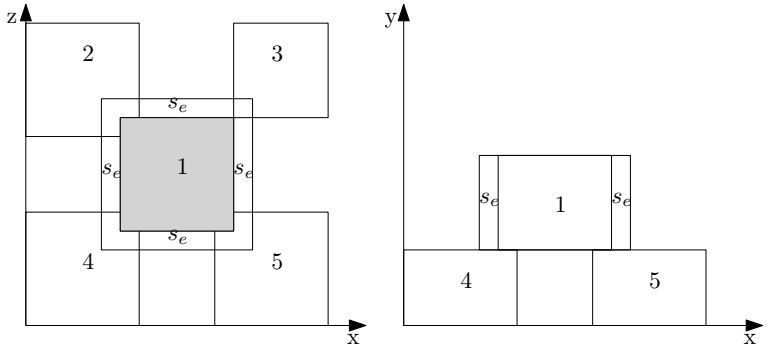

Figure 3: Edge contact support

are both zero. However, we do not need to calculate explicitly the forces or moments between bodies.

In our work, we adopt the next definition from [15]. A tower of boxes is *stable* if for every layer of boxes the center of gravity of layers $i + 1$ up to the top lies above the interior of the convex hull of the contact points between layers i and $i + 1$.

For *contact regions*, the motion produced by any distribution of contact forces over the entire contact region can be produced by equivalent forces acting on only the vertices of the contact region; the same is true for line segment contact regions [11]. Interior points of contact regions are not considered as *contact points*. Thus, a configuration of bodies can be considered to have finite many contact points only. [1]

In figure 4, we see a box a top of three boxes. We have marked all the contact points with black dots, the convex hull with bold line. The center of gravity is marked with a circle. The minimum distance from the center of gravity to the edge of the convex hull is marked with line.

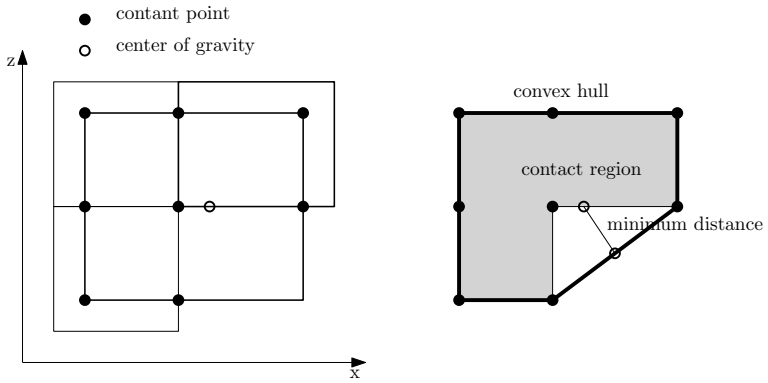

Figure 4: Three boxes below one box: the minimum distance from the center of gravity to the edge of convex hull

Since relative positions are stored into memory, we get the supporting boxes of already packed boxes easily. Only the first layer needs to be recalculated, i.e. we have to test every box whether it supports placed box. It is called finding *deep supporting boxes*. This tree structure cannot directly tell us whether all the the layers are stable.

Thus, the concept of *mass layer* is devised: By mass layer we mean a set of boxes that together form a layer for which we have a common top surface and a common bottom surface. For this box group we can determine the center of gravity. In figure 5, boxes 2 and 3 form a

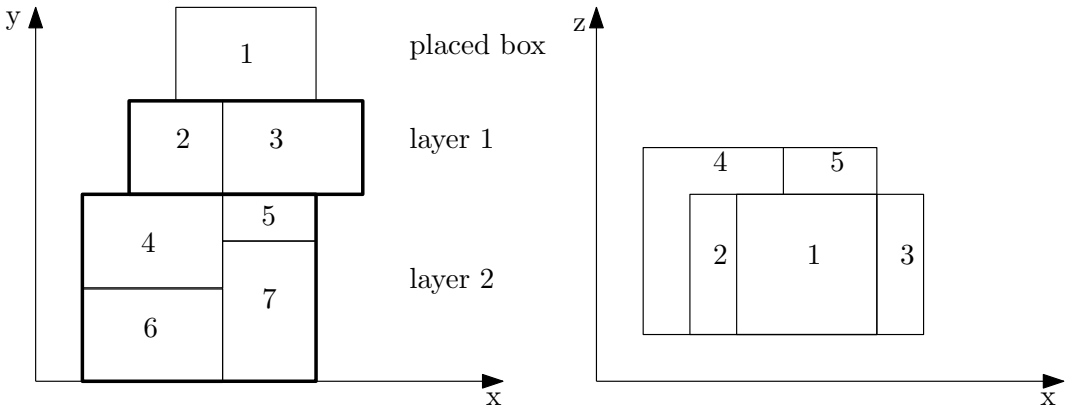

Figure 5: Mass layer

mass layer, also boxes 4, 5, 6 and 7. On the other hand, boxes 4 and 5 or 6 and 7 do not form a common surface, so they do not form a mass layer.

As an example, we now place the box number one to its candidate location, see figure 5. First, we calculate its center of gravity $\vec{r}_{cg}$ and project it to the xz-plane $\vec{r}_{cg}^{xz}$ above the first layer. If we don't know the weight distribution, we assume it to be uniform. So the center of gravity $\vec{r}_{cg}$ for the box i is simply $(x_i + w_i/2, y_i + h_i/2, z_i + l_i/2)$. Then, we check the overhang and edge support conditions. After that, the contact region and the contact points between the box one and the first layer is determined. We calculate the convex hull and measure the shortest distance d_{min} from $\vec{r}_{cg}^{xz}$ to edge of the convex hull, see figure 4. To calculate the convex hull, well-known methods can be used, like Graham Scan [5]. In principle, the distance d_{min} could be zero for a stable case, but for the robustness we reserve a tolerance d_{min}^{tol} for it. If $\vec{r}_{cg}^{xz}$ is inside the convex hull and d_{min} is larger than the tolerance value d_{min}^{tol}, then we proceed to the next layer.

We join the box one and the layer one together (boxes 1,2,3), and calculate its center of mass

$$\vec{r}_{cg}^{new} = \frac{m_1\vec{r}_1 + m_2\vec{r}_2}{m_1 + m_2} \tag{1}$$

where m_1 and m_2 are the masses of the box (one) and layer (of boxes two and three) and $\vec{r}_1$ and $\vec{r}_2$ are the center of gravities of the box and the next layer. Next, we continue to the second layer, calculate the contact region and contact points between layers 1 and 2. And again, we determine the convex hull and check whether $\vec{r}_{cg}^{new}$ is inside it, and the distance is within tolerance. This procedure is continued until we are at the bottom level. If all these tests pass, the current location of the box one is stable. In summary, the proposed method is described in algorithm 1.

Algorithm 1: Is the location stable?

Data: box b, candidate location l, list of packed boxes and parameters s_o, s_e, d_{min}^{tol}
Result: true or false
if *box is on floor* **then**
  stop and **return** true
end
Get all the supporting boxes D of the smaller rectangle defined by s_e.
if *there is no supporting boxes in* D **then**
  stop and **return** false
end
Let $A = \{b\}$ be the set of boxes above.
Calculate the center of gravity $\vec{r}_{cg}$ of A and project it to $\vec{r}_{cg}^{xz}$ plane of boxes that support A.
Convert all the supporting boxes D to mass layers M.
for *each layer* l *in* M *from top to down* **do**
  if l *is the first layer and the overhang condition for* s_o *is not met* **then**
    stop and **return** false
  end
  Let P the contact points of the contact region between the bottom of A and the top of layer l. Calculate the convex hull of contact points P.
  Determine the smallest distance d_{min} from $\vec{r}_{cg}^{xz}$ to the convex hull.
  if $d_{min} < d_{min}^{tol}$ **then**
    stop and **return** false
  end
  Join layer l to A. Calculate the center of gravity $\vec{r}_{cg}$ of A and project it to $\vec{r}_{cg}^{xz}$ plane that support A. If the plane is floor, stop and **return** true
end

3 THE PACKING METHODS

In this section, two constructive packing heuristics are described: a single corner method and a four corner method. For both methods we give as an input a list of n boxes $\pi_1, \pi_2, \ldots, \pi_n$ and a list of orientation for each box $\gamma_1, \gamma_2, \ldots, \gamma_n$. We will pack these boxes in this order and in this given orientation.

3.1 Single Corner Method

A *single corner method* is proposed by He et al. [6]. We call this method as a single corner method, since the method starts and tries to pack boxes towards a single corner of the pallet.

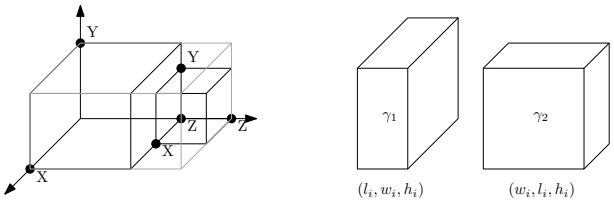

Figure 6: Reference points and orientations

The concept of *reference point* is used. It is a point where we can insert a box to its left-front-bottom corner (x_i, y_i, z_i). Reference point is *feasible* if certain conditions are met: box is not intersecting with the pallet's boundaries or any other box, and the stability rejection method should accept the location. The list of conditions can be easily extended.

An empty pallet contains one reference point at origo $(0, 0, 0)$. Then for every box π_i with dimensions (l_i, w_i, h_i) that is inserted to a selected reference point, three more reference points are created $(x_i + l_i, z_i, y_i)$, $(x_i, z_i + w_i, y_i)$, $(x_i, z_i, y_i + h_i)$, which we classify as a X-, Z- and Y- reference points, respectively. Origo is classified as O.

Next, we describe how to select a reference point. To measure the goodness of reference point we have two values: cage index I_{cage} and spare index I_{spare}. The concept of *cage bin* is used. It measures the volume for packed boxes, that is the smallest rectangular box that contains all the packed boxes. Its dimensions $l_c \times w_c \times h_c$ may grow as we pack more boxes. For reference point k, let the box π_i be packed at point k. After this, let V_i be the volume of all the packed boxes. Now we measure l_c^k, w_c^k, h_c^k, which are the length, width and height of the cage bin. The cage index is defined as

$$I_{cage} = \frac{l_c^k \times w_c^k \times (h_c^k)^2}{V_i}. \tag{2}$$

We penalize height dimension more, hence $(h_c^k)^2$. As stated in [6], only using cage index for reference point selection is not comprehensive enough. There are reference points which have the same cage index value, since the cage bin is not growing. A spare volume is defined as dynamic residual space with dimensions of $l_s \times w_s \times h_s$, where $l_s = (l_c - x_k), w_s = (w_c - y_k)$ and $h_s = (h_c - z_k)$. The spare index is defined as

$$I_{spare} = \frac{l_s \times w_s \times h_s^2}{ln_i \times wn_i \times hn_i} \tag{3}$$

Table 1: Different move chains depending on reference point type.

Reference point type	Moves in turn
Type X	$s \to (z) \to y \to (x) \to y$
Type Y	$s \to (z) \to y \to (x) \to y \to z$
Type Z	$s \to (x) \to y \to (z) \to y$
Type O	s

which is a measure of the spare-height over the filled rate of the spare volume. [6] The values ln_i, wn_i and hn_i are the dimensions of the rotated box.

After a box is placed on a reference point, the box can be moved along x-, y- and z-axis to make the packing more compact and to fill gaps between boxes. This *parallel moving* was also conducted in Wu et al. [14] and He et al. [6] with different strategy.

To perform parallel moves a concept of *parallel chains* is deviced. The reference point types are X,Y,Z,O and their associated parallel chains are presented in table 1. parallel chains are presented in table 1. We start from the location of the reference point s and we do movements to different directions in given order. The movements that are in brackets are not stored in the chain. So for example for the type X, the chain is s, y, y. The returned chain do not contain the same locations, only distinct elements are returned.

Let $R = \{r_1, r_2, \ldots, r_n\}$ be the list of n candidate reference points. We start from the best reference point r_1. Depending on its type we do all the parallel moves, which gives a chain of possible placements. Then we test from the chain end to the chain start, whether or not the placement is feasible. The first feasible placement is selected. If no placement is feasible, we take the second best reference point r_2. We repeat this procedure until we find the first feasible placement. Otherwise, if we can't find any, we try to pack on the top of the packing.

If no reference point is feasible to put the current box, we then put it at $v_0 = (0, 0, H_c)$, where H_c is the height of the current cage bin [6]. If this location is not feasible, we then try to do *reverse parallel moves*, that is, the opposite direction of the parallel moves. Let $-z$ and $-x$ mean reverse parallel moves to z-axis and x-axis. We then try to do the following sequence: $v_0 \to (-z) \to y \to (-x) \to y \to (-z) \to y$. If can't find a feasible placement, we try the next sequence: $v_0 \to (-x) \to y \to (-z) \to y \to (-x) \to y$. Finally if this doesn't work, we can try $v_0 \to (-x) \to y$ and $v_0 \to (-z) \to y$. If none of these moves produce a feasible placement, we must conclude that the given box and orientation sequence is infeasible.

If a feasible location for a box is found, then we store it and associated *relative positions* to the memory. For each box we store its supported boxes. At the later state of the iteration after we have identified the first layer, we can calculate all the later layers from memory. We call this finding *deep supporting boxes*.

Some further improvements are made. It is possible to combine boxes of the same size to a single box π_i. We combine two boxes only if their narrow ratio is within a set parameter. The *narrow ratio* is defined as $\min\{w_i/h_i, l_i/h_i\}$. For example, if $w_i < l_i$, the combined box dimension is $(2w_i, l_i, h_i)$. When we have found a final position for a composite box, it is then unwrapped and stored as its factors.

It is also useful to be able to continue on an interrupted, unfinished job. In this case, there

are two sets of boxes: those that are already packed and those that will be packed. For already packed boxes, we store them directly to its final location.

In summary. we give a pseudo code for the single corner packer in algorithm 2.

Algorithm 2: Single Corner Packer

Data: List of boxes $\pi_1, \pi_2, \ldots, \pi_n$ and their orientations $\gamma_1, \gamma_2, \ldots, \gamma_n$
Result: Packing solution
Let $R = \{(0, 0, 0)\}$ be a list of all reference points.
for *each box* i *in* $\pi_1, \ldots, \pi_n$ **do**
  Let $F = \emptyset$ be a list of feasible reference points.
  for *each reference point* r_k *in* R **do**
    if π_i *is inside pallet boundaries and do not intersect with packed boxes* **then**
      Mark a reference point r_k as feasible, insert it to F and compute the cage and spare index values of this point by (2) and (3), respectively.
    end
  end
  Sort F: take first spare index points and then cage index points are in increasing order
  while do
  end
  if *there are feasible reference points* $|F| > 0$ **then**
    Choose the best reference point among F and do parallel chain.
  else
    Pack box π_i upon the current cage bin and do reverse parallel moves
  end
  Store its final coordinates (x_i, y_i, z_i) and relative positions to other boxes
  Add three new x,y,z-reference points in R
  Update the dimensions of the cage bin
end

3.2 Four corner method

Our proposed four corner method is an extension to the single corner method. (The drawback of the single corner is ...) Instead of pushing boxes towards a single corner, this method tries to push boxes towards all four corners. Basically, every corner has its own cage bin, reference point list, parallel moves and a list of packed boxes that have been tried to pack that corner. It is like these corners are competing with each other on the space resource. Next we describe one way to implement this idea.

The basic idea is that for four corners we have four *inside packers*, which are mirrors of each other. In this way, we can use directly the structure of the single corner packer and we don't have to separate parallel moves for every corner. A concept of *inside packer* is similar to single corner packer, but the difference is that it can store boxes in two ways. First, it stores packed boxes which updates a cage bin and adds the three reference points to the list of reference points. Second, it can store boxes, called *virtual boxes*, that do not grow the cage bin and do not produce reference points. In both cases, relative positions are stored.

In figure 7, can be seen that we have four different packers called P_O, P_X, P_Z and P_{XZ}. The first packer P_O is our original, main packer. The second packer P_X is an x-axis mirror and the third P_Z is a z-axis mirror of the first packer. The fourth packer P_{XZ} is mirror of both axes.

Whenever we store a box to its location, we make four copies of it to every inside packer.

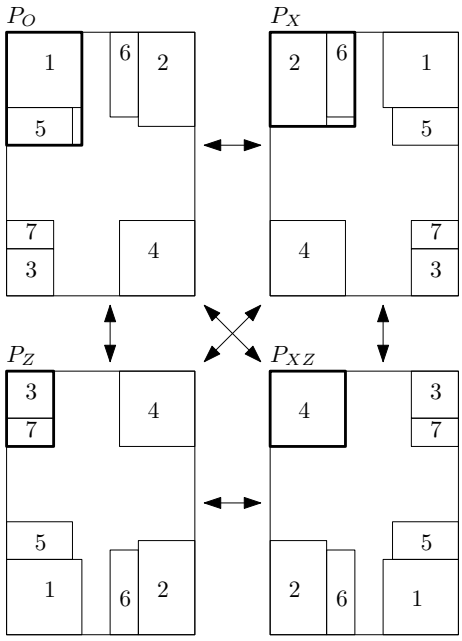

Figure 7: Four different corners.

Packer	store normally	store virtually
P_O	$P_O : O \mapsto O$	$P_X : O \mapsto X$, $P_Z : O \mapsto Z$ and $P_{XZ} : O \mapsto XZ$
P_X	$P_X : O \mapsto O$	$P_O : O \mapsto X$, $P_Z : O \mapsto XZ$ and $P_{XZ} : O \mapsto Z$
P_Z	$P_Z : O \mapsto O$	$P_O : O \mapsto Z$, $P_X : O \mapsto XZ$ and $P_{XZ} : O \mapsto X$
P_{XZ}	$P_{XZ} : O \mapsto O$	$P_O : O \mapsto XZ$, $P_X : O \mapsto Z$ and $P_Z : O \mapsto X$

Table 2: Different move chains depending on the reference point type.

By notation $P_X : A \mapsto B$ we mean a transformation from the packer A's coordinate system to the packer B's coordinate system and store the result to packer X. In table 2, it is listed how we store boxes to different inside packers, based on in which packer we orinally try to pack on. For example, if a box is packed to packer P_Z, we store it normally to that packer and make a z-axis copy to packer P_O etc. In figure 7, it can be seen these relations visually.

Reference points are selected in a similar way as in the single corner method. The only difference is that every inside packer has its own list of reference points. Thus, we join these lists and sort them. Although we make four copies of a box, we do not produce more reference points than previously, because they are scattered to different inside packers. First sort *spare indexes* and then *cage indexes* reference points in ascending order as previously. The reference points are now in order. Parallel moves are done within the inside packer where the selected reference point belongs to. The moves are done in chains as previously and we select the first feasible placement we find. If no feasible placement is found, then we try to pack on top of every inside packer.

As an example, in figure 7, we have packed seven boxes. The first four are packed to corners, because empty corner has always the best index value.

4 EVOLUTIONARY ALGORITHMS

In this section, we present the main ideas of our implementation of genetic algorithm and the global search framework. For more detailed explanation, refer to He et al. [6]. For small cases, enumerating all permutations can be evaluated, but for larger problems genetic algorithm or global search framework must be used.

4.1 The genetic algorithm

Genetic algorithms are widely applied in operations research due to their easy implementation and good quality solutions for hard problems. We have adopted our method based on the methods proposed by Wu et al. [14] and He et al. [6]. Our chromosome is *represented* as a permutation. We need to consider two factors: the order of boxes to be packed and the box rotation. All boxes are first sorted in volumetric descending order and numbered: $V_1 \geq V_2 \geq \cdots \geq V_n$. Next we decribe all the steps needed to perform the algorithm.

In the first step, an initial population of chromosomes are generated. Let P_{size} denote the size of the population. Sorting the boxes according to certain rules has demonstrated some advantages in providing better starting solutions. We create two chromosomes with the same box sequence in decreasing volume order. All boxes in chromosome one have orientation $\gamma_i = 1$ and all boxes in chromosome two have orientation $\gamma_i = 2$. The rest of the population are produced with random box sequences and box orientations. At every iteration, a new population is created through selection, crossover and mutation until a termination condition is met.

Step 2. Selection is the process to choose most of the better individuals in each generation for crossover and mutation. We adopt the tournament selection method where the objective value of a chromosome can be directly used as the selection criterion. It is also efficient to code and independent of the scaling the fitness function. First, the *elite mechanism* is applied: n_{elite} the best solution found so far have been placed into the reproduction pool. The number of bits in chromosome includes order bits and orientation bits. We may also set the minimum difference percentage in bits between elite members, so that we can also select worse inviduals. Then, the rest $P_{\text{size}} - n_{\text{elite}}$ individuals are selected by tournament process. Let p be the tournament size. For each tournament round, we repeatedly take p chromosomes at random and choose the best of them to be placed into the reproduction pool.

Step 3. Crossover is the process during which two parents generate two offsprings who inherit elements from each parent. For different chromosome representations different crossover operators are used. For permutation representation (He et al.), *partially matched crossover* operator is adopted for the box sequence. That is, a two-point crossover, a two random points in chromosome are selected for two parents and the two pairs between the crossover points of the two parents are switched over to form two children. Parental orientation is preserved in the nearly children chromosome. For example, let two box and orientation sequences be

$$\pi_1^1, \pi_2^1, \pi_3^1, \pi_4^1 | \gamma_1^1, \gamma_2^1, \gamma_3^1, \gamma_4^1 \quad \text{and} \quad \pi_1^2, \pi_2^2, \pi_3^2, \pi_4^2 | \gamma_1^2, \gamma_2^2, \gamma_3^2, \gamma_4^2.$$

We choose randomly two numbers, e.g. 2 and 3. We exchange all these boxes between these numbers and get

$$\pi_1^1, \pi_2^2, \pi_3^2, \pi_4^1 | \gamma_1^1, \gamma_2^2, \gamma_3^2, \gamma_4^1 \quad \text{and} \quad \pi_1^2, \pi_2^1, \pi_3^1, \pi_4^2 | \gamma_1^2, \gamma_2^1, \gamma_3^1, \gamma_4^2.$$

Repairing is needed if the children is not feasible due to some boxes appear in the children twice while some do not appear at all. *Random repair* creates one pool of duplicated bits and one pool of missing bits. The actual repairing replaces the duplicated bit with missing bit at random to restore feasibility. This was also used by Wu et al. [14].

In a step 4, mutation is the process to change some genes in a chromosome and produce a new chromosome. The *swap mutation* found favorable to the problems with asymmetrical feature, especially in solving the large-size instances (He et al. [6]). That is, the order of the boxes to be packed is changed and boxes are rotated. For example, first, by changing genes 2 and 4 the order is

$$\pi_1, (\pi_2), \pi_3, (\pi_4), \pi_5 | \gamma_1, (\gamma_2), \gamma_3, (\gamma_4), \gamma_5$$

is mutated to

$$\pi_1, (\pi_4), \pi_3, (\pi_2), \pi_5 | \gamma_1, (\gamma_4), \gamma_3, (\gamma_2), \gamma_5.$$

Then we give random orientation mutation, for example $1, 2, 1, 2, 1$ is mutated to $2, 1, 1, 2, 1$.

In the step 5, to evaluate the fitness of every chromosome of the population, we decode the chromosomes to packing solutions. The single corner or four corner packer with or without stability rejection method is used.

Several metrics can be used to measure the goodness of the packing solution. One packing objective is to maximize *volume utilization*. The concept of volume utilization is $V/(L \times W \times H^{sol})$, where V is the total box volume, L and W are the pallet length and width and H^{sol} is the height of the final solution.

In the step 6, we use the typical termination condition for genetic algorithm that is it stops when a fixed number of iterations I_n is reached. It can also stop when the objective function is not improved within a number of iterations.

Algorithm 3: Genetic algorithm

```
Data: A packing problem
Result: best chromosome and its packing solution
Produce an initial generation of chromosomes P_s and evaluate them
while the terminal condition is not met do
    Select n_e elite members
    while population size is not P_s do
        Select a chromosome pair with tournament selection
        Do crossover operator for this pair
        Repair
        Do mutation operator for this pair
        Evaluate the pair
        Add the chromosomes to new population
    end
end
return the best chromosome and its packing solution
```

4.2 The global search framework

In evolutionary algorithms, different runs may produce different solutions. Also with a single run, premature convergence is a main drawback. Therefore, several computational tests are carried out and the best solution is selected. Better solutions are possible to obtain if more computational tests are conducted. The problem is to determine how to perform these tests and when to stop. The aim is to carry out tests in reasonable computation time and get more robust results with less calibration. As a solution, a global search framework (He et al. [6]) is adopted. In this framework, both evolved solutions and the evolutionary gradient guide the search.

A concept of *evolutionary gradient* is adopted. In numerical optimization methods, especially in smooth convex optimization, when the gradient of an objective function with certain point is zero, the optimum is found. In evolutionary algorithms, when a number N of computational tests are carried out to evaluate the genetic algorithm, for each test l, the *relative deviation* $RD_l = (f^l_{\text{obj}} - f_{\text{best}})/f_{\text{obj}}$ is calculated, where f^l_{obj} is the objective value and f_{best} the current best solution. The mean relative deviation is defined as evolutionary gradient

$$E_{\text{dev}} = \frac{1}{N}\sum_{l=1}^{N} RD_l.$$

If all tests obtain the same objective value, then the gradient vanishes, in other words $E_{\text{dev}} = 0$ and it may not be necessary to carry out more tests. Otherwise, $(E_{\text{dev}} > 0)$, more tests are done for better solutions.

In the global search framework, these tests can be organized in three or two local phases. We have set the number of tests as a constant N (see He at al. [6] for more).

In the first phase, called *rough search*, N tests of the genetic algorithm is performed with initial generation including the heuristic seeds and random chromosomes. After the tests, the evolutionary gradient is calculated. If the gradient is zero, we are done. Otherwise, we continue on the next phase.

In the second phase, called *refining search*, N genetic tests are conducted. Each test is initialized with the advanced chromosomes from the last phase and random chromosomes. That is, all the N evolved solutions obtained in iteration g are put into the initial generation of each test in iteration $g + 1$. Heuristic seeds are also included. We repeat this phase until the gradient is eventually zero.

The third phase is optional and reserved for hard cases. If no new solution is found, only one refining search is executed, otherwise, with better solution is obtained, several refining searhes are to be conducted until the gradient is zero.

It is straightforward to implement algorithm 4 as a serial or a parallel code. For example, all for-loops can be done in parallel.

Algorithm 4: Global Search Framework

Data: A packing problem, number of tests N
Result: the best packing solution
The global iterations $g = 0$
for $l = 1 \dots N$ **do**
  Rough search: genetic algorithm with heuristic seeds
  $g \leftarrow g + 1$
end
Calculate $E_{dev}(g)$
while $E_{dev}(g) \neq 0$ **do**
  for $l = 1 \dots N$ **do**
    Refining Search: genetic algorithm with advanced seeds and heuristic seeds
    $g \leftarrow g + 1$
  end
  Calculate $E_{dev}(g)$
end
return the best packing solution

5 NUMERICAL EXPERIMENTS

In this section, we apply the proposed methods to our own test cases and the preliminary test results are analyzed. In table 3 are listed all the eight algorithm combinations: the genetic algorithm (GA) and the global search framework (GSF) are tested with the single (1) and four corner method (4) with and without the stability rejection method (S) or box combiner (B). When the algorithms are compared, we will refer to their given notation. For the global search framework, the number of rough and refining searches are listed also. All the algorithms are coded in C# in .NET Framework 4.

Notation	Algorithm	Corners	Stability	Rough Searches	Refined Searches
GA-1-(B)	GA	1	no	-	-
GA-4-(B)	GA	4	no	-	-
GA-1S-(B)	GA	1	yes	-	-
GA-4S-(B)	GA	4	yes	-	-
GSF-1-(B)	GSF	1	no	8	8
GSF-4-(B)	GSF	4	no	8	8
GSF-1S-(B)	GSF	1	yes	8	8
GSF-4S-(B)	GSF	4	yes	8	8

Table 3: The tested algorithm combinations.

5.1 Problem data

In table 4 are listed our real data test cases that were collected from a company. For every test, the pallet dimensions are $(L \times W \times H_{\max} = 1200 \times 800 \times 3000)$ in millimeters. In these cases, the number of boxes to be packed and the number of different types are listed. The number of boxes is no more than 40.

Test problems can be categorized into three groups: in group 1, $(N \leq 21)$ are a small number

of boxes, in group 2, $(26 \leq N < 40)$ a moderate number of boxes and in group 3, $(N = 40)$ are packed forty boxes with increasing number of box types.

Ten runs were conducted for each test case and the worst, best, average performance, the standard deviation of the run results, the improvement of the average performance against the actual operation and the average running times are shown. The best solution is selected from 10 computational tests.

Experiments were conducted with Google Cloud Platform [1]. We selected High-CPU machine type with 32 virtual CPUs and 28.8 GB of memory. A virtual CPU is implemented as a single hardware hyper-thread on a 2.3 GHz Intel Xeon E5 v3 (Haswell).

group 1		group 2		group 3	
problem	N/types	problem	N/types	problem	N/types
order264	11/7	order94	26/24	order1	40/24
order139	11/10	order265	25/19	order222	40/30
order120	15/12	order217	30/16	order213	40/31
order114	15/13	order21	30/21	order215	40/31
order106	20/18	order82	35/34	order226	40/36
order177	21/18	order61	35/29	order228	40/37
-		-		order227	40/38
-		-		order229	40/39

Table 4: Test cases

5.2 Algorithm parameter configuration

Algorithm parameter configuration impacts the performance and results. These are the preliminary settings and the parameters could be fine-tuned. Especially, different settings should be found depending on the problem size and the number of boxes. Currently, all the algorithms use the same setting.

In stability rejection method, the edge support limit is set to 20 mm. Mass distribution is assumed to be uniform based on the box volume. Maximum overhang is set to 50 mm and maximum overhang percentage is 20%. Minimum distance from the center of gravity to the edge of convex hull is set to 50 mm.

In packer parameters, two same sized boxes are combined if their narrow ratio is at least 0.8.

In genetic algorithm, the population size P_s is set to 200. The number of iterations I_n is fixed to 100. The crossover rate $C_r = 0.85$ and mutation rate $M_r = 0.2$. Orientation mutation rate $O_r = 0.1$. We take in every iteration 5 elite chromosomes for which the minimum difference percent is 0.3. For selection, tournament size is 5. In summary, $(P_s, I_n, C_r, M_r, O_r, E_n, E_d, T_n) = (200, 100, 0.85, 0.2, 0.1, 5, 0.3, 5)$. We use random repairer and swap mutation. Boxes are sorted in volumetric order.

In global search framework, the number of rough searches and refining searches or the number of GA tests within a global iteration are set to $N = 8$. Number of iterations of each GA test is fixed to 25.

[1] https://cloud.google.com

5.3 Results and analysis

The comparative study is carried out from three aspects: the packing height H/volume utilization F, computational time T and qualitative analysis of stability and packability. In the appendix are listed eight tables: In tables A1 to A4, are listed genetic algorithm results. We compare GA-1 to GA-4 and GA-1S to GA-4S. In tables A5 and A8, are listed global search framework results. Again we compare GSF-1 to GSF-4 and GSF-1S to GSF-4S.

First, we compare **packing height** and **volume utilization**. They are equivalent measures, so we can compare them together. Due to only two orientations, the search space of the problem is smaller than with six orienations, but it is harder to find tight solutions. To compare the maximum performance, running with six orientations should be tested.

When we compare the global search framework to the genetic algorithm we notice that volume utilization and packing height are better. This happens almost always. It takes more time to compute the GSF but it converges better and the deviation of volume utilization between different runs is lower. By only using the genetic algorithm the quality of solution varies more. In practice, GSF must be used, so we continue on compare their differences.

From the tables A1 - A8 we note that all the algorithms that use the four corner method on average volume utilization is higher and packing height is lower than in the results produced by the single corner method.

The search space of stable packing is smaller than the solution space without stability. In GSF, this means that volume utilization drops on average from 79% to 76%. The difference will be larger, when there are more than forty boxes to be packed.

There is a small advantage to use box combiner. Without stability GSF-1-B and GSF-4-B. have better volutime utilization. The four corner method seems to benefit more than single corner method. With stability the difference is not significant.

Secondly, **computational time** is an important factor to measure the efficiency. Tables A1 - A8 present the computational times T in seconds, which is an average of ten runs for each problem. The total run time of all tests is about 50 hours, but can be run in parallel in the cloud service within 15 minutes.

The performance of GA tells us what is the packing performance of each packer. The single corner method is always faster than four corner method. Using box combiner will make the search space smaller, but the execution time is quite the same. Without stability the four corner method takes 10% more time and with stability 30%. However, the stability method generates a lot of infeasible solutions. If the population size is 200, in the beginning there could be 150 infeasible solutions and in the end there is usually 50. This affects the convergence rate. For example, with stability the single corner method may produce more infeasible solutions, thus it converges faster.

The GSF performance depends on the convergence speed and thus the number of global iterations. The average number of global iterations g is about 3. High quality solutions typically needs many global iterations. Untable GSF needs 2.54 - 3.19 global iterations. Stable GSF needs 3.15 - 3.77 global iterations.

In the GSF, if we increase the number of rough and refining search e.g. from 5 to 10 tests, the volume utilizations are not much higher, but the computational time doubles. So we get almost the same result with less time.

Without stability GSF-4 takes 30% more time than GSF-1. With stability method, GSF-4S

can take up to 60% more timne than GSF-4.

In unstable GSF, there is no difference in performance between using box combiner or not in group 3. In group 4, box combiner takes a bit more time, but the results are also better. In stable GSF, However, in group 3 and 4, box combiner has performance advantage 10%. but volume utilizations drops.

Finally, from practical point of view, the most important factor is **stability** of the load. It is clear that methods that do not consider stability, do not have any guarantee that the boxes are packable. Boxes can almost be even in the air and this can be seen in the right solution of figure 8. In figure 9, the stability method fixes this drawback. In figure 10, although the packing solution is very compact, we see from the right figure that three boxes that would fall if packed. In stable version 11, there is not that problem. It is argued in literature that high volume utilization indicates high stability. However, our methods do not confirm this.

Qualitatively the single corner method procudes packing results in which the boxes have more side contanct to other boxes and the packing has two flat sides on the edges of the pallet. Also the final cage bin of the single corner method is always equal or smaller than the final cage bin of four corner. On the other hand, the four corner method produces four flat sides. This makes the four corner method visually more appealing. This also helps to wrap the shrinking foil evenly around the package.

For stable methods, the stability rejection method produces solutions that have high towers at the beginning of the iteration, but solution converges close to methods without stability. The tendency to produce towers, also produces more gaps between boxes. This can be improved with a finalization procedure that may push boxes towards each other with the constraint that the stability method must accept the solution.

6 CONCLUSION

The present work addresses three-dimensional bin packing problem with stability rejection criterion that includes some restrictions which had not been worked out altogether in previous works. In addition, it introduces improvements in the previous single corner packer and an extension that enables to pack boxes from four different corners. New research perspectives have to do with the inclusion of additional considerations such as dynamic stability and optimization criteria based on stability. A finalization procedure based on the stability checking can be used to rearrange and compact the end result. In conclusion, when the volume utilization is near 80 percent for larger and harder test cases, the genetic algorithm with the global search framework can produce reasonable and compact packing results.

REFERENCES

[1] Baraff D.: *Analytical methods for dynamic simulation of non-penetrating Rigid Bodies*, Computer Graphics, Volume 23, Number 3, July 1989.

[2] Bischoff, E.E., Janetz, F., Ratcliff, M.S.W., *Loading pallets with non-identical items*, European Journal of Operational Research 84 (1995) 681692.

[3] Bortfeldt A., Wscher G.: *Constraints in container loading – A state-of-the-art review*, European Journal of Operational Research 229 (2013) 1-20.

[4] Chen C.S., Lee S.M., Shen Q.S.: *An analytical model for the container loading problem*, European Journal of Operational Research 80 (1995) 68-76.

[5] Graham R. L.: *An efficient algorithm for determining the convex hull of a finite planar set*, Information processing letter 1 (1972) 132 - 133.

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

[7] Huang W., He K., *A caving degree approach for the single container loading problem*, European Journal of Operational Research 196 (2009) 93-101.

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

[9] Lin J., Chang C., Yang J.: *A study of optimal system for multiple-constraint multiple-container packing problems*, Advances in Applied Artificial Intelligence: 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006, Annecy, France, June 27-30, 2006. Proceedings,

[10] Martello S, Pisinger D, Vigo D, den Boef E, Korst J.: *Algorithm 864: general and robot-packable variants of the three-dimensional bin packing problem*, ACM Transactions on Mathematical Software 2007;33:112.

[11] Palmer R. S.: *Computational complexity of motion and stability of polygons*, Ph.D Thesis, 1989.

[12] Ramos A. G., Oliveira J. F., Gonçalves, Lopes M. P.: *Dynamic stability metrics for the container loading problem*, Transportation Research Part C 60 (2015) 480 - 497.

[13] Schuster M., Bormann R., Steidl D., Reynolds-Haertle S., Stilman M.: *Stable stacking for the distributor's pallet packing problem*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10) Oct. 2010.

[14] Wu Y., Li W., Goh M., de Souza R.: *Three dimensional bin packign problem with variable bin height*, European Journal of Operational Research 2010, 202, 347 - 55.

[15] Zwick U.: *Jenga*, Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '02, 2002.

A Appendix

		GA-1			GA-4		
problem	type	$maxF$	$minH$	$avgT$	$maxF$	$minH$	$avgT$
order265	25/19	72.54	710.00	15.56	80.47	640.00	19.60
order94	26/24	70.46	660.00	17.01	72.66	640.00	22.90
order217	30/16	74.57	705.00	23.96	79.06	665.00	29.05
order21	30/21	74.29	750.00	23.37	77.93	715.00	30.16
order61	35/29	72.91	930.00	35.44	75.76	895.00	38.60
order82	35/34	77.84	870.00	37.42	78.29	865.00	46.72
average		73.77	770.83	25.46	77.36	736.67	31.17
order1	40/24	77.17	1130.00	52.91	78.92	1105.00	56.31
order213	40/31	77.59	1060.00	51.56	78.33	1050.00	55.82
order215	40/31	78.30	935.00	48.85	78.30	935.00	54.54
order226	40/36	75.98	1110.00	50.93	78.09	1080.00	54.00
order227	40/38	77.62	925.00	48.78	79.33	905.00	54.76
order228	40/39	74.90	925.00	48.00	77.84	890.00	55.90
order229	40/39	79.25	920.00	50.35	79.69	915.00	62.38
order222	40/40	79.29	925.00	49.66	78.86	930.00	55.06
average		77.51	991.25	50.13	78.67	976.25	56.10

Table A1: Without box combiner

		GA-1-B			GA-4-B		
problem	type	$maxF$	$minH$	$avgT$	$maxF$	$minH$	$avgT$
order265	25/19	76.30	675.00	16.01	79.85	645.00	19.69
order94	26/24	70.46	660.00	17.14	73.81	630.00	21.35
order217	30/16	73.53	715.00	27.17	79.66	660.00	28.86
order21	30/21	74.29	750.00	23.09	76.85	725.00	28.77
order61	35/29	73.70	920.00	33.60	75.76	895.00	38.11
order82	35/34	78.29	865.00	35.35	78.75	860.00	44.20
average		74.43	764.17	25.39	77.45	735.83	30.16
order1	40/24	77.86	1120.00	46.82	78.57	1110.00	53.76
order213	40/31	76.87	1070.00	54.08	78.71	1045.00	53.00
order215	40/31	78.73	930.00	48.58	78.30	935.00	55.35
order226	40/36	75.30	1120.00	48.02	77.73	1085.00	56.66
order227	40/38	78.04	920.00	49.17	78.04	920.00	54.70
order228	40/39	72.92	950.00	51.27	78.73	880.00	55.26
order229	40/39	79.25	920.00	49.08	80.12	910.00	55.21
order222	40/40	78.86	930.00	49.28	78.44	935.00	58.04
average		77.23	995.00	49.54	78.58	977.50	55.25

Table A2: With box combiner

		GA-1S			GA-4S		
problem	type	$maxF$	$minH$	$avgT$	$maxF$	$minH$	$avgT$
order265	25/19	70.55	730.00	22.16	73.57	700.00	25.16
order94	26/24	68.89	675.00	20.10	70.99	655.00	27.27
order217	30/16	79.06	665.00	29.42	79.06	665.00	34.46
order21	30/21	74.79	745.00	31.81	76.85	725.00	33.89
order61	35/29	72.52	935.00	32.93	74.51	910.00	46.08
order82	35/34	73.61	920.00	36.28	76.96	880.00	46.17
average		73.24	778.33	28.78	75.33	755.83	35.51
order1	40/24	74.22	1175.00	56.66	76.16	1145.00	66.58
order213	40/31	71.52	1150.00	45.53	75.11	1095.00	63.54
order215	40/31	77.89	940.00	48.78	19 77.07	950.00	62.90
order226	40/36	66.93	1260.00	20.25	68.29	1235.00	43.10
order227	40/38	75.98	945.00	51.75	77.20	930.00	58.63
order228	40/39	74.90	925.00	58.75	73.31	945.00	67.12
order229	40/39	74.40	980.00	31.83	75.56	965.00	54.98
order222	40/40	78.44	935.00	62.08	78.44	935.00	64.03
average		74.28	1038.75	46.95	75.14	1025.00	60.11

Table A3: Without box combiner

		GA-1S-B			GA-4S-B		
problem	type	$maxF$	$minH$	$avgT$	$maxF$	$minH$	$avgT$
order265	25/19	69.60	740.00	19.51	76.87	670.00	24.16
order94	26/24	69.93	665.00	20.21	69.93	665.00	23.27
order217	30/16	72.02	730.00	29.71	84.12	625.00	33.48
order21	30/21	74.29	750.00	31.36	77.93	715.00	35.56
order61	35/29	72.91	930.00	34.29	73.31	925.00	41.88
order82	35/34	74.02	915.00	32.82	75.67	895.00	47.72
average		72.13	788.33	27.98	76.30	749.17	34.35
order1	40/24	74.54	1170.00	53.58	75.83	1150.00	67.08
order213	40/31	71.52	1150.00	41.13	74.43	1105.00	63.80
order215	40/31	77.48	945.00	47.33	77.48	945.00	60.66
order226	40/36	64.87	1300.00	21.72	69.41	1215.00	38.85
order227	40/38	77.62	925.00	49.81	77.20	930.00	62.95
order228	40/39	74.09	935.00	55.32	73.31	945.00	63.71
order229	40/39	75.17	970.00	28.41	71.48	1020.00	50.75
order222	40/40	78.44	935.00	59.05	77.61	945.00	71.92
average		74.22	1041.25	44.55	74.59	1031.88	59.97

Table A4: With box combiner

		GSF-1				GSF-4			
problem	type	g	$maxF$	$minH$	$avgT$	g	$maxF$	$minH$	$avgT$
order265	25/19	2.60	79.85	645.00	37.36	2.63	81.10	635.00	58.19
order94	26/24	3.10	72.66	640.00	50.22	2.40	80.17	580.00	57.18
order217	30/16	3.00	75.10	700.00	62.42	3.33	80.26	655.00	99.58
order21	30/21	2.70	76.85	725.00	56.48	2.90	78.48	710.00	86.83
order61	35/29	3.20	74.11	915.00	92.14	2.50	75.76	895.00	95.95
order82	35/34	3.00	79.68	850.00	87.07	3.40	79.68	850.00	137.30
average		2.93	76.37	745.83	64.28	2.86	79.24	720.83	89.17
order1	40/24	3.30	79.28	1100.00	130.11	2.80	79.64	1095.00	158.15
order213	40/31	2.80	76.16	1080.00	114.94	2.50	83.08	990.00	140.91
order215	40/31	3.30	78.30	935.00	135.16	2.00	78.30	935.00	115.40
order226	40/36	3.10	76.32	1105.00	120.49	3.22	78.45	1075.00	181.82
order227	40/38	2.80	78.47	915.00	112.34	2.33	78.04	920.00	155.83
order228	40/39	3.10	78.28	885.00	128.44	2.80	77.84	890.00	159.13
order229	40/39	2.90	81.01	900.00	111.43	2.60	80.12	910.00	170.27
order222	40/40	2.30	78.86	930.00	94.51	2.10	78.44	935.00	117.27
average		2.95	78.34	981.25	118.43	2.54	79.24	968.75	149.85

Table A5: Without box combiner

		GSF-1-B				GSF-4-B			
problem	type	g	$maxF$	$minH$	$avgT$	g	$maxF$	$minH$	$avgT$
order265	25/19	3.80	80.47	640.00	53.76	3.30	81.75	630.00	71.04
order94	26/24	2.50	71.54	650.00	38.09	2.50	84.55	550.00	59.28
order217	30/16	2.70	76.75	685.00	55.05	2.70	79.66	660.00	75.53
order21	30/21	3.10	76.33	730.00	65.22	2.70	77.93	715.00	75.47
order61	35/29	2.90	77.49	875.00	82.71	2.70	81.70	830.00	99.99
order82	35/34	3.10	79.21	855.00	90.27	3.70	79.68	850.00	140.44
average		3.02	76.97	739.17	64.18	2.93	80.88	705.83	86.96
order1	40/24	2.90	80.01	1090.00	115.75	3.20	80.01	1090.00	168.93
order213	40/31	3.90	81.84	1005.00	163.11	3.00	83.50	985.00	160.31
order215	40/31	3.10	78.30	935.00	121.62	2.50	79.58	920.00	135.05
order226	40/36	2.90	76.32	1105.00	117.58	2.60	79.56	1060.00	135.77
order227	40/38	3.10	78.47	915.00	128.19	2.70	78.47	915.00	167.87
order228	40/39	3.40	77.84	890.00	136.75	3.40	79.63	870.00	184.20
order229	40/39	3.70	81.92	890.00	148.13	2.80	80.57	905.00	169.21
order222	40/40	2.50	79.29	925.00	102.08	2.40	78.86	930.00	134.62
average		3.19	79.25	969.38	129.15	2.83	80.02	959.38	157.00

Table A6: With box combiner

		GSF-1S				GSF-4S			
problem	type	g	$maxF$	$minH$	$avgT$	g	$maxF$	$minH$	$avgT$
order265	25/19	3.50	76.87	670.00	67.74	2.90	78.63	655.00	76.59
order94	26/24	2.70	69.93	665.00	48.79	2.20	69.93	665.00	54.64
order217	30/16	3.40	81.51	645.00	83.50	3.00	81.51	645.00	97.84
order21	30/21	3.30	77.39	720.00	91.27	3.50	78.48	710.00	126.76
order61	35/29	3.20	75.76	895.00	86.09	4.00	77.05	880.00	161.97
order82	35/34	4.20	78.29	865.00	113.55	3.70	80.15	845.00	151.81
average		3.38	76.62	743.33	81.82	3.22	77.62	733.33	111.60
order1	40/24	4.20	76.50	1140.00	193.29	4.30	79.64	1095.00	263.60
order213	40/31	4.20	78.71	1045.00	149.99	4.10	75.46	1090.00	231.63
order215	40/31	3.30	77.89	940.00	121.28	3.50	77.89	940.00	195.94
order226	40/36	4.20	70.57	1195.00	69.51	4.20	73.33	1150.00	153.96
order227	40/38	3.40	76.79	935.00	130.43	3.60	78.04	920.00	228.79
order228	40/39	2.60	75.30	920.00	112.09	3.70	75.71	915.00	220.03
order229	40/39	3.50	76.35	955.00	83.30	3.67	75.95	960.00	196.21
order222	40/40	3.50	78.44	935.00	171.13	3.10	78.44	935.00	200.59
average		3.61	76.32	1008.13	128.88	3.77	76.81	1000.63	211.34

Table A7: Without box combiner

		GSF-1S-B				GSF-4S-B			
problem	type	g	$maxF$	$minH$	$avgT$	g	$maxF$	$minH$	$avgT$
order265	25/19	3.50	72.54	710.00	62.24	3.60	79.85	645.00	89.15
order94	26/24	2.60	72.09	645.00	44.36	3.10	70.46	660.00	71.68
order217	30/16	3.60	79.66	660.00	87.18	3.30	81.51	645.00	103.39
order21	30/21	3.10	74.29	750.00	82.08	2.90	80.17	695.00	101.76
order61	35/29	2.50	73.31	925.00	70.11	3.20	75.76	895.00	121.12
order82	35/34	3.60	75.67	895.00	102.49	3.10	77.40	875.00	122.50
average		3.15	74.59	764.17	74.74	3.20	77.52	735.83	101.60
order1	40/24	3.70	75.83	1150.00	166.89	4.70	79.28	1100.00	285.29
order213	40/31	4.00	73.11	1125.00	119.50	3.60	76.51	1075.00	193.20
order215	40/31	3.60	78.30	935.00	129.39	3.20	79.15	925.00	169.58
order226	40/36	4.20	71.47	1180.00	67.49	3.50	72.08	1170.00	120.22
order227	40/38	3.70	78.04	920.00	137.09	2.80	77.20	930.00	165.36
order228	40/39	3.20	74.90	925.00	134.46	3.00	74.90	925.00	162.82
order229	40/39	3.30	77.16	945.00	71.29	3.60	77.16	945.00	187.87
order222	40/40	3.10	78.44	935.00	147.54	3.00	78.44	935.00	178.27
average		3.60	75.91	1014.38	121.71	3.43	76.84	1000.63	182.83

Table A8: With box combiner

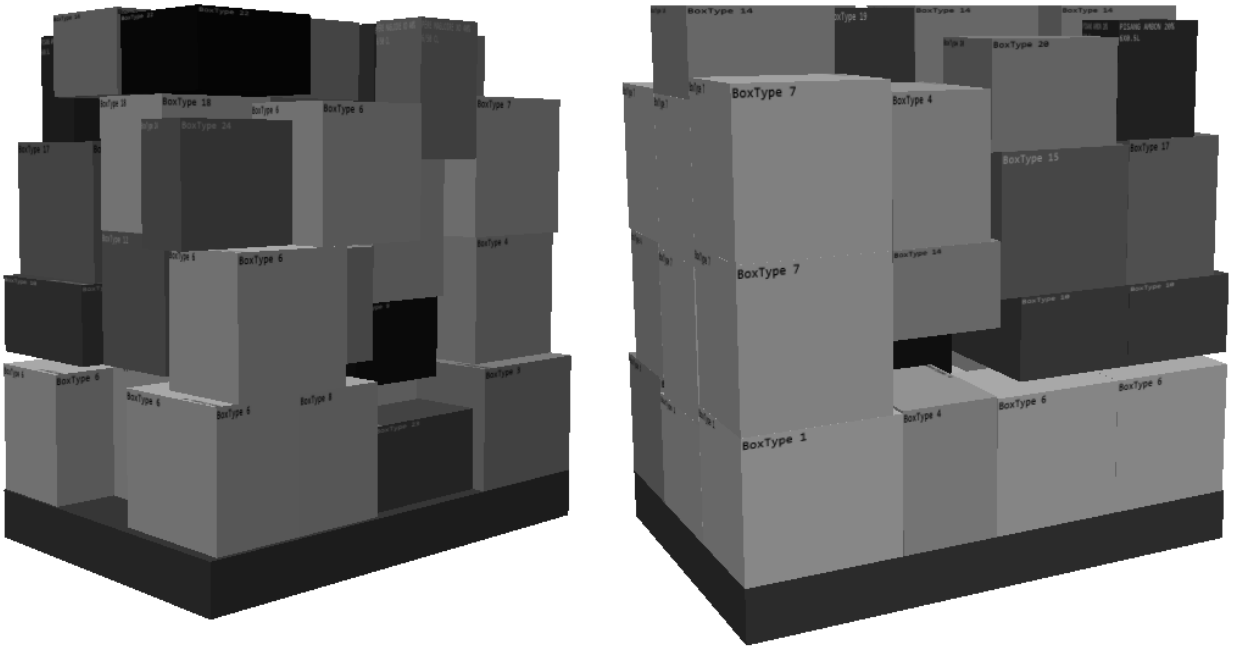

Figure 8: Unstable single corner

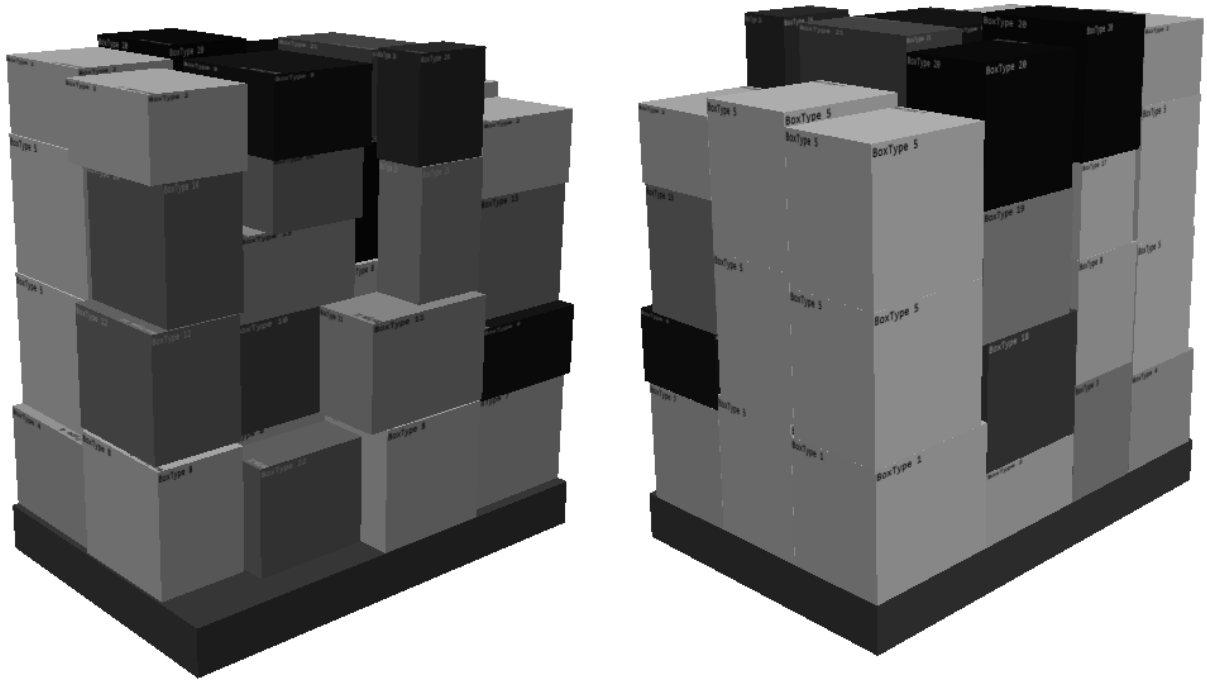

Figure 9: Stable single corner

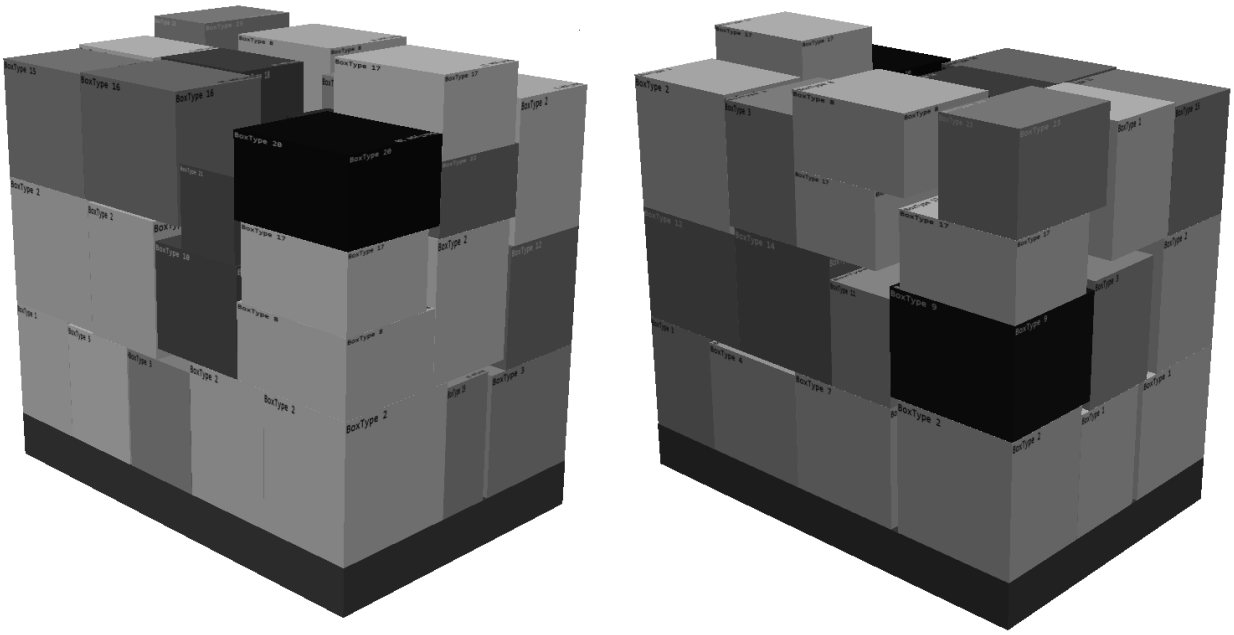

Figure 10: Unstable four corner

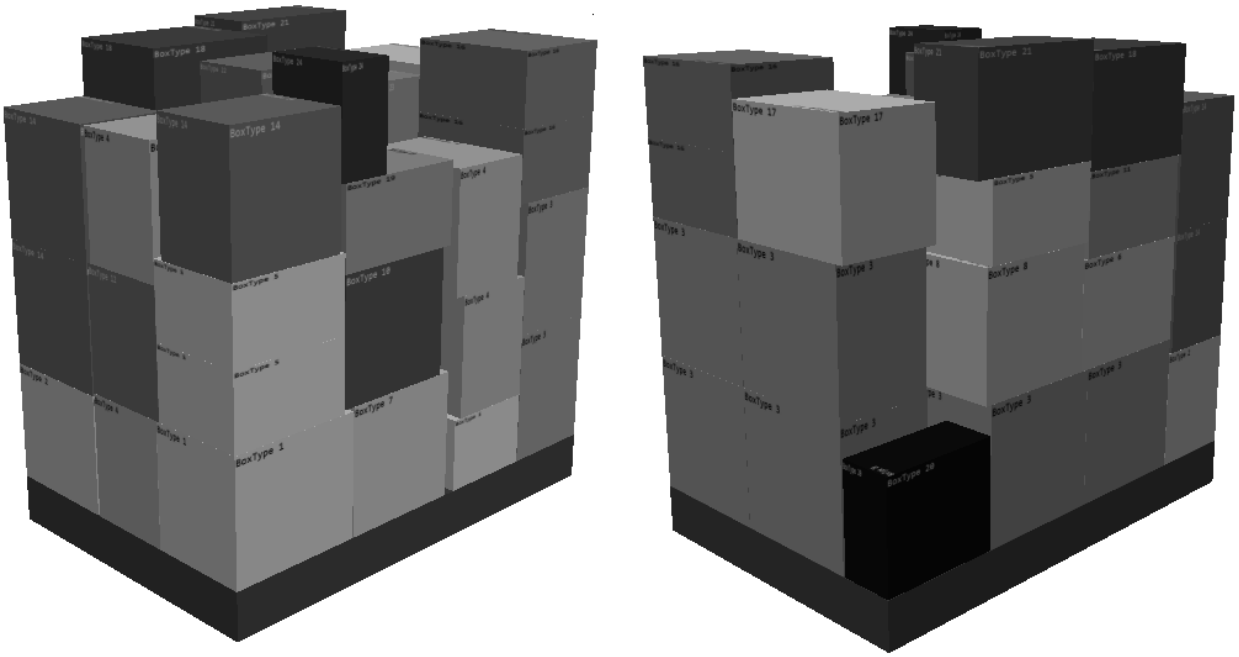

Figure 11: Stable four corner