A caving degree approach for the single container loading problem

Article *in* European Journal of Operational Research · July 2009

DOI: 10.1016/j.ejor.2008.02.024

Kun He
Huazhong University of Science and Technology
46 PUBLICATIONS **244** CITATIONS

Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 196 (2009) 93–101

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

A caving degree approach for the single container loading problem

Wenqi Huang, Kun He [*]

College of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China

Received 26 April 2007; accepted 20 February 2008
Available online 29 February 2008

Abstract

Inspired by an old adage "Gold corner, silver side and strawy void", and improved by a new observation "Maximum value in diamond cave", a new heuristic approach is proposed for solving the three-dimensional single container loading problem. Differing from several previous approaches, its key issue is to pack the outside item into a corner or even a cave in the container such that the item is as compactly and closely as possible with other packed items. Experiments are on two groups of public and difficult benchmarks. For the 47 without-orientation-constraint instances from the OR-Library, experiments indicate an average packing utilization of 94.9%, which improves current best result reported in the literature by 3.9%. For the 800 strongly heterogeneous instances among 1500 representative benchmarks proposed by Bischoff et al., (100 instances in a set), experiments show an average packing utilization of 87.97%, which improves current best record reported in the literature by 0.28%. Besides, new best records are achieved on the latter five sets among the eight sets of strongly heterogeneous benchmarks.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Packing; Single container loading; Heuristics; Combinatorial optimization

1. Introduction

Packing problems are a kind of combinatorial optimization problems and find many applications in the real world such as: glass, paper or steel cutting, container loading, pallet loading, and VLSI design. Cutting problems are a kind of transformation, and algorithms solving a packing one are also suitable to the corresponding cutting one. A packing problem has two sets of data, containers and items, whose elements define geometric shapes, and the process is to pack a subset of items into the containers to optimize an objective function. There are several variations for packing problems and often an approach for one type could be used for others. A typology introduced by Dyckhoff (1990) initially provided an excellent classification for the problems. Wascher et al. (2007) improved it to a new categorization in dealing with recent developments and in making it more general.

This paper considers a Single Container Loading Problem (SCLP), which requires loading a subset of cuboid items into a single cuboid container such that the volume of the packed items is maximized. The items are weakly heterogeneous (Items can be grouped into relatively few types according to their size) or strongly heterogeneous (Only very few items are of identical size). In the typology of Wascher et al. (2007), this scenario can be classified as a three-dimensional cuboid SLOPP (Single Large Object Placement Problem) or SKP (Single Knapsack Problem), whereas Dyckhoff (1990) classify it 3/B/O/F, 3/B/O/M or 3/B/O/R (3: three-dimensional, B/O: one object/container and selection of items, F: few items of different types, M: many items of many different types, R: many items of relatively few different types). The three-dimensional packing is a generalization of two-dimensional or one-dimensional packing, and even the one-dimensional packing is known to be NP-hard (Lim et al., 2003).

Approaches for the SCLP can be classified as wall building and layering (George and Robinson, 1980; Loh and Nee, 1992; Bischoff and Ratcliff, 1995; Bortfeldt and Gehring, 2001; Gehring and Bortfeldt, 2002; Pisinger, 2002;

[*] Corresponding author. Tel.: +86 13 006163312; fax: +86 27 87545004.
E-mail addresses: wqhuang@mail.hust.edu.cn (W. Huang), brooklet60@gmail.com (K. He).

Lim et al., 2003; Moura and Oliveira, 2005; Bischoff, 2006), guillotine cutting (Morabito and Arenales, 1994), cuboid (block) arrangement (Bortfeldt and Gehring, 1998; Eley, 2002; Bortfeldt et al., 2003) and stack building (Gehring and Bortfeldt, 1997). Besides, many authors incorporated intelligent search methods to improve the solution quality, including tabu search (Bortfeldt and Gehring, 1998; Bortfeldt et al., 2003), genetic algorithm (Gehring and Bortfeldt, 1997, 2002; Bortfeldt and Gehring, 2001), tree search (Morabito and Arenales, 1994; Eley, 2002; Pisinger, 2002; Lim et al., 2003), and random search (Moura and Oliveira, 2005; Bischoff, 2006). The wall building and layering approach, first introduced by George and Robinson (1980), is most commonly used and modified by later researchers for its high efficiency and high quality. This approach usually opens a new layer (wall) with a width equals to some item dimensions, then each layer is filled up by a number of horizontal strips, and each strip is packed in a greedy way. The major disadvantage of this approach is that it creates layers as flat as possible, so unnecessary restrictions on the locations of items are imposed, which may miss optimal packing. Another efficient approach is cuboid (block) arrangement, which utilizes the characteristic of weakly heterogeneous problems and binds same size items into a larger cuboid to do the tentative packing. Thus it can achieve good results for weakly heterogeneous problems, and the volume utilization tends to become low when problems become more and more strongly heterogeneous.

Researchers also paid increasing attention to various practical constraints on the SCLP which may considerably complicate the task, including orientation constraint (Bischoff and Ratcliff, 1995; Gehring and Bortfeldt, 1997, 2002; Bortfeldt and Gehring, 1998, 2001; Davies and Bischoff, 1999; Eley, 2002; Bortfeldt et al., 2003; Lim et al., 2003; Moura and Oliveira, 2005; Bischoff, 2006), weight distribution (Gehring and Bortfeldt, 1997, 2002; Davies and Bischoff, 1999; Bortfeldt and Gehring, 2001; Eley, 2002), loading stability (Bischoff and Ratcliff, 1995; Gehring and Bortfeldt, 1997, 2002; Bortfeldt and Gehring, 1998, 2001; Eley, 2002; Bortfeldt et al., 2003; Moura and Oliveira, 2005), stacking constraint (Gehring and Bortfeldt, 1997, 2002; Bortfeldt and Gehring, 1998, 2001), load bearing strength (Bischoff, 2006), container weight capacity limit (Gehring and Bortfeldt, 1997, 2002; Bortfeldt and Gehring, 1998, 2001), on-line item supply (Hemminki et al., 1998), and multi-drop situations (Bischoff and Ratcliff, 1995). The orientation constraint requires that one or two side dimensions of the items may not be used as the height. This constraint has been widely taken into account in the literature and is considered in this paper. Loading stability, another important constraint, requires that: (1) all boxes are fully supported; or (2) several other boxes support each box; or (3) all boxes have at least three sides supported (Moura and Oliveira, 2005). Some researchers only permit condition (1) as a stable loading. This constraint will be considered in our future work.

We propose a new approach for solving the SCLP. The inspiration is from an old adage “Gold corner, silver side and strawy void” for Chinese Weiqi (Weiqi is a board game played with stones on a large square board. The objective of this game is to control a larger territory than one’s opponent by placing one’s stones so they cannot be surrounded and captured by the opponent, but can surround and capture any stones the opponent plays inside one’s own territory). The adage indicates the relative values of capturing regions in a corner, on a side or in the center of the Weiqi board. We improved the idea with “Maximum value in diamond cave” for the SCLP. The key issue is that the packing item always occupies a corner or even a cave if possible, such that the items are packed as compactly as possible. A “cave” is something like an empty hollow surrounded by many items. Experiments on 47 instances that have no orientation constraint in the OR-Library (Beasley, 1990) and on 800 strongly heterogeneous instances proposed by Davies and Bischoff (1999) improve current best results by 3.9% and by 0.28% respectively.

A former version of our approach appeared in (Huang and He, 2007), which is based on the work of Huang et al. (2007) for the two-dimensional rectangle packing. Main differences include the improved priority rules, caving degree definition and search method, which make the approach more efficient.

2. Problem specification

Let C be a cuboid container with fixed length L, width W and height H; consider C embedded into a three-dimensional Cartesian reference frame, in such a way that the lower-left-near corner coincides with the origin and the upper-right-far corner is placed at (L, W, H), as Fig. 1 shows. Let $S = \{1, 2, \ldots, n\}$ indexes a set of n cuboids, referred to as items too, with each item i having length l_i, width w_i and height h_i.

For each packed item i, let (x_{i1}, y_{i1}, z_{i1}) and (x_{i2}, y_{i2}, z_{i2}) denote the coordinates of the lower-left-near vertex and the upper-right-far vertex respectively. Consequently, $(x_{i2}-x_{i1}, y_{i2}-y_{i1}, z_{i2}-z_{i1})$ corresponds to the item dimensions on x-, y-, and z-axes. For example in Fig. 1, suppose item i is placed at the upper-left-near corner of the container and

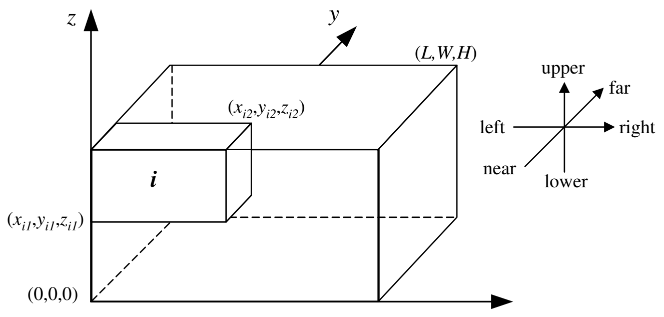

Fig. 1. Coordinate system.

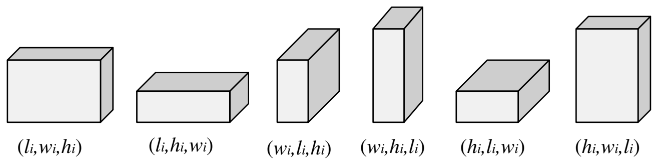

Fig. 2. Possible item orientations.

its dimensions on x-, y-, and z-axes is (l_i, w_i, h_i), then $(x_{i1}, y_{i1}, z_{i1}) = (0, 0, H - h_i)$, and $(x_{i2}, y_{i2}, z_{i2}) = (l_i, w_i, H)$.

Definition 1 (*Valid Orientations*, O_i). An item i with dimensions l_i, w_i, and h_i may have one to six orientations that are allowable when packing the item into the container, with item dimensions on x-, y-, and z-axes belonging to set O_i, O_i is not null and is a subset of $\{(l_i, w_i, h_i), (l_i, h_i, w_i), (w_i, l_i, h_i), (w_i, h_i, l_i), (h_i, l_i, w_i), (h_i, w_i, l_i)\}$. The six possible orientations are shown in Fig. 2, with their orientation number from 1 to 6, respectively.

Definition 2 (*Distance of Two Items*, d_{ij}). The distance between item i and j is $d_{ij} = \mathrm{dx}_{ij} + \mathrm{dy}_{ij} + \mathrm{dz}_{ij}$, as Fig. 3 shows, where

$$\mathrm{dx}_{ij} = \max(\max(x_{i1}, x_{j1}) - \min(x_{i2}, x_{j2}), 0);$$
$$\mathrm{dy}_{ij} = \max(\max(y_{i1}, y_{j1}) - \min(y_{i2}, y_{j2}), 0);$$
$$\mathrm{dz}_{ij} = \max(\max(z_{i1}, z_{j1}) - \min(z_{i2}, z_{j2}), 0).$$

Let δ_i be an indicator variable taking the value 1 or 0 according as item i is or is not packed. Then, the problem can be formulated as follows:

Input: A container with length $L \in Q^+$, width $W \in Q^+$ and height $H \in Q^+$; Set of n items with each item i having dimensions $l_i \in Q^+$, $w_i \in Q^+$, $h_i \in Q^+$ and valid orientations O_i (Q^+ represents the set of positive real numbers).
Goal: max $\sum_{i=1}^{n} l_i w_i h_i \delta_i$
Subject to:
(1) $(x_{i2}-x_{i1}, y_{i2}-y_{i1}, z_{i2}- z_{i1}) \in O_i$
(2) $\max(\mathrm{dx}_{ij}, \mathrm{dy}_{ij}, \mathrm{dz}_{ij})\delta_i\delta_j \geqslant 0$
(3) $0 \leqslant x_{ik} \leqslant L$, $0 \leqslant y_{ik} \leqslant W$, $0 \leqslant z_{ik} \leqslant H$, $k = 1, 2$
(4) $\delta_i \in \{0,1\}$

In constraint (1)–(4), i, j applies to $1,2,\ldots,n$ and $i \neq j$. Constraint (1) implies that each packed item is located orthogonally with a valid orientation; (2) implies that there is no overlapping between any two packed items; (3) implies that each packed item is completely in the container; (4) denotes whether an item is packed into the container. The goal is to pack as much item volume as possible.

3. The caving degree approach

Three algorithms are presented in this section: a basic largest caving degree algorithm, a strengthened caving degree algorithm, and a strengthened top-N caving degree algorithm. There are some definitions that need to be introduced first. The most important definition is caving degree, which is defined to judge how compact and close an item is with other items already packed in the container.

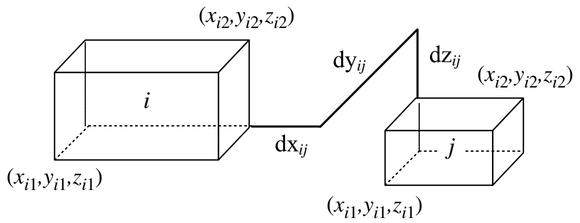

Fig. 3. Distance of two items.

3.1. Definitions

Definition 3 (*Corner*). A corner is an unoccupied space where three orthogonal surfaces of different items meet.

Fig. 4 shows two corners formed by three items. We can regard the six surfaces of the container as six flat items. So, any packing starts with the six flat items and eight corners of C.

Definition 4 (*Packing Action*). It is an action that places an outside item into the container so that it satisfies the problem constraints. The item being placed is called the action item.

Definition 5 (*Corner Occupying Action*, *COA*). A Corner Occupying Action (COA) is a packing action that places an item so that one of its vertices coincides with a corner.

A COA includes three aspects: which item to be packed, which corner to be selected, and which item orientation to be set.

Definition 6 (*Paste Number*, k_i). A surface of the COA item is called pasted if the surface contacts at least one surface of other items and the coinciding area is larger than 0. Paste number is the number of the COA item's pasted surfaces. ($k_i \in \{3,4,5,6\}$).

For example, in Fig. 5, suppose item a has been packed and we try to pack item b or c into the container. If we pack item b to the lower-left-near corner of the container, the lower, left and near surfaces of item b are pasted. So, the

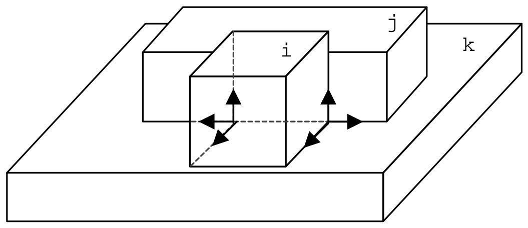

Fig. 4. Two corners.

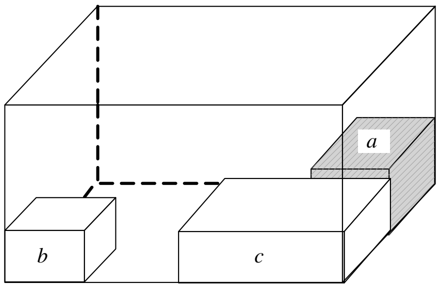

Fig. 5. Paste number and paste ratio.

paste number of this action is 3. If we pack item c to the lower-right-near corner of the container, the lower, right, near and far surfaces of item c are pasted. So, the paste number of this action is 4. A high paste number is better than a low one, because an action with higher paste number places the item more compactly with other packed items.

Definition 7 (*Paste Ratio*, r_i). It is the COA item's pasted area, divided by the COA item's total surface area. ($r_i \in (0,1]$)

For example, in Fig. 5, the lower surface, left surface and near surface of item b are fully pasted. So, the paste ratio of this action is 0.5; as to item c, suppose its x-, y-, and z-dimension is (4, 4, 1), the lower, right, and near surfaces are fully pasted and the far surface is half pasted. So, the paste ratio of this action is $(16+4+4+2)/48 = 0.54$. A high paste ratio is better than a low one, because an action with higher paste ratio places the item more compactly with other packed items.

Definition 8 (*Distance of a COA*, d_i). Each of the k pasted surfaces of a COA can be extended to an infinite plane. Aligned with the k planes and their normal directions (pointing to the space the action item is in), a space Ω is indicated in the container. Let B denote the set of items whose coinciding area or volume with space Ω is larger than 0, but they do not paste with i. Then the distance d_i of the COA is defined as $\min\{d_{ij} \mid j \in B\}$. Let $d_i = 0$ when $k_i = 6$.

For example, in Fig. 6, suppose the action item i occupies a corner formed by item a, b, and c. Its upper, lower, left and far surfaces are pasted ($k_i = 4$). The four planes defined by these four surfaces enclose a space Ω in the container, as the bold cuboid space shows. When computing d_i, we first find all items whose coinciding area or volume with space Ω is larger than 0: item a, b, c, d, e, and three flat items corresponding to the lower, right, and near surfaces of the container. Then we get rid of the items that paste action item i: item a, b, c, and a flat item corresponding to the lower surface of the container. Therefore, d_i is the distance between item i and item e, which is the minimum distance between item i and the set of {item d, item e, two flat items corresponding to the near and right surfaces of the container}.

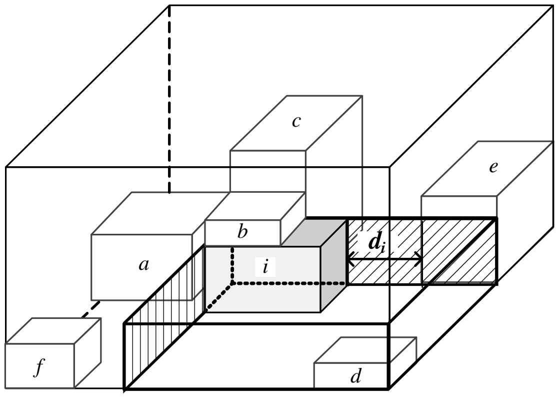

Fig. 6. Distance of a COA.

The smaller the d_i is the better a COA is, because an action with smaller distance packs the item more close to other packed items.

Definition 9 (*Adjacent Degree*, ad_i). It is a dimensionless value that normalizes distance d_i.

$$\mathrm{ad}_i = \exp\left(\frac{-d_i}{\sqrt[3]{l_i w_i h_i}}\right), \quad \mathrm{ad}_i \in (0,1].$$

Adjacent degree $\mathrm{ad}_i = 1$ when $d_i = 0$; ad_i decreases towards to 0 as d_i becomes very large. If two action items have the same distance d_i then the item with lower volume will have smaller adjacent degree. The higher the value of ad_i is, the closer the action item is to other packed items.

Definition 10 (*Caving Degree*, C_i). The caving degree of a COA is defined as follows:

$$C_i = 100 \cdot k_i + 10 \cdot \mathrm{ad}_i + r_i.$$

Because $3 \leqslant k_i \leqslant 6$, $0 < \mathrm{ad}_i \leqslant 1$, and $0 < r_i \leqslant 1$, the coefficients 100, 10 and 1 are defined such that caving degree is decided first by paste number, then by adjacent degree, and last by paste ratio. The higher the value of C_i is the more desirable a 'cave' is for the action item, because more surfaces of the item are pasted, more close it is to other packed items, and more area of its surface is pasted. A desirable 'cave' makes the item be packed as compactly and closely as possible with other packed items.

3.2. The packing process

The packing process is similar for the three algorithms. A packing step shows the evolution of the process, and at each step an outside item is packed into the container.

At step 0, the container is initialized with a flat dummy item at each surface of the container, the items in set S are put to a free item list, and the eight corners of the container are put to a free corner list.

After initialization, a packing procedure is invoked recursively. And at each step, it searches all COAs by iteration over all combinations of corners, types of outside item and valid orientations. Then a best COA is selected by some ranking rules, explained in detail in the next subsections, and the selected item is placed to a corner with one of its orientations. Items with same size belong to one item type. Same type items are only considered once at each step because they have the same COA set. Afterwards the list of free corners and the list of free items are updated, corners the item consumed are deleted and corners the item created are added.

This packing process will go on until at a final step all items have been packed into the container without overlapping, or none of the remainders can be packed in. In that case, it outputs the layout of all packed items and the volume utilization of the container.

3.3. The basic algorithm

At each step, the basic Algorithm A_0 always selects a COA with the largest caving degree.

Theorem 1. *At each step there must be at least eight corners unless the container utilization reaches* 100%.

Proof. There is at least one connected empty space in the container, enclosed by surfaces of other packed items, if the container utilization does not reach 100%. Apparently, item surfaces that meet at the space boundary are orthogonal with each other.

Start from any point in the empty space and keep doing the following until no move goes on work:

(1) Move down (↓) until the point touches an upper surface of any packed item, go to (2);
(2) Move left (←) until the point touches a right surface of any packed item, go to (3).
(3) Move near (↙) until the point touches a far surface of any packed item, go to (1).

Denote above movement as (↓, ←, ↙). Process (1), (2), (3) must stop at a time because we reach a new surface only once and the number of surfaces must be finite. In the end we could find a corner where three surfaces of different items meet.

Similarly, start from any point in the empty space and do movement (↓, →, ↙), (↓, ←, ↘), (↓, →, ↘), (↑, ←, ↙), (↑, →, ↙), (↑, ←, ↘) and (↑, →, ↘) respectively, we could find another seven corners. Here symbol "↑" means move up, "→" means move right, and "↘" means move far.

So, there must be at least eight corners when the container utilization does not reach 100%. □

E.g. in a two-dimensional space, if we try to pack a set of rectangle items into a rectangle container, then at each step there must be at least four corners where two edges of different items meet orthogonally. Fig. 7 shows a packing step at which the shady place is filled with rectangle items. Start from point o and do movement (↓, ←), (↓, →), (↑, ←), (↑, →), then we could find four corners, as marked in Fig. 7.

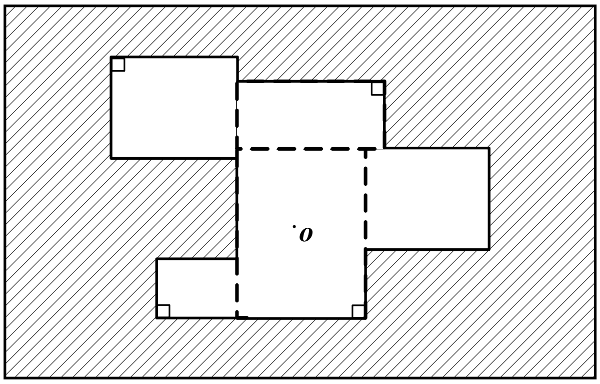

Fig. 7. Corners in a two-dimensional container.

Based on above definitions and analysis, Algorithm A_0 can be described as follows.

Algorithm A_0

```
1. At current step, for each free corner {
       For each type of the free items {
           For each valid item orientation {
               Arbitrarily select a free item according to the item type;
               Pack the item provisionally to the corner with current orientation;
               Check whether this packing action violates the problem constraints;
               If it is a COA, compute its caving degree;
   }}}
2. A best COA is selected to do basing on some parameters of the action item in a lexicographic order:
   1)  caving degree: larger
   2)  length of the long dimension: larger
   3)  length of the middle dimension: larger
   4)  length of the short dimension: larger
   5)  value of $x_{i2}$, $y_{i2}$,or $z_{i2}$ which corresponds to the long dimension of the container: smaller
   6)  value of $x_{i2}$, $y_{i2}$, or $z_{i2}$ which corresponds to the middle dimension of the container: smaller
   7)  value of $x_{i2}$, $y_{i2}$, or $z_{i2}$ which corresponds to the short dimension of the container: smaller
   8)  item orientation number: smaller
3. Update free corner list and free item list;
4. Repeat 1, 2 and 3, until all items have been packed into the container without overlapping, or none of the remainders can be packed in.
```

A_0 could compute independently by starting from step 0, or be a subprocess by starting from a step of the strengthened algorithms. Here "larger" means the larger is the better, and "smaller" means the smaller is the better.

Lexicographic order means that the algorithm uses Rule (1) to select a COA with the largest caving degree; if more than one COAs have the same largest caving degree, it uses Rule (2) to select a COA; in case of ties, it uses Rule (3); and so on. Rules (2)–(4) give priority to items with one long edge or large volume, say, which are harder to pack. Rules (5)–(7) try to put items into a relatively incommodious dimension of the container. E.g. for a container that $L > W > H$, Rule (5) compares the value of x_{i2}, Rule (6) compares the value of y_{i2}, and Rule (7) compares the value of z_{i2}; while for a container that $H > W > L$, Rule (5) compares the value of z_{i2}, Rule (6) compares the value of y_{i2}, and rule 7) compares the value of x_{i2}. Rule (2) to Rule (8) break tie to make the algorithm deterministic.

3.4. The strengthened algorithms

At each step, the strengthened Algorithm A_1 always selects a COA with the largest pseudo utilization.

Definition 11 (*Pseudo Utilization*, U_i). At current step, do a COA to get a new step and pseudo execute the basic Algorithm A_0, then the final container volume utilization obtained by A_0 is called the pseudo utilization of this action.

So, each COA has corresponding pseudo utilization. Starting from step 0, Algorithm A_1 executes as follows:

Algorithm A_1

```
1. At current step, for each free corner {
     For each type of the free items {
       For each valid item orientation {
         Arbitrarily select a free item according to the item type;
         Pack the item provisionally to the corner with current orientation;
         Check whether this packing action violates the problem constraints;
         If it is a COA, compute its pseudo utilization.
   }}}
2. A COA with the largest pseudo utilization is selected to do; in case of ties, ranking Rules (2)–(8) of A_0 are used in the same lexicographic order.
3. Repeat 1 and 2, until one of the followings is satisfied:
   (1) all items have been packed into the container without overlapping;
   (2) none of the remainders can be packed in;
   (3) a time limit is reached.
```

Time limit is set to 3 hours for the experiments in this paper. Fig. 8 shows how to select a COA at a step i: for each COA, A_1 computes the pseudo utilization by using A_0, which corresponds to a problem solution, and does a COA with the best pseudo utilization to get a new step $i+1$.

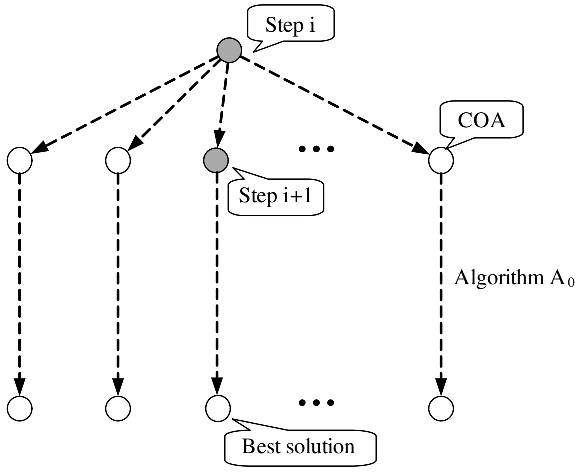

Fig. 8. One step search of Algorithm A_1.

The strengthened top-N Algorithm A_2 is similar to Algorithm A_1. The only difference is that at each step, instead of considering all COAs, A_2 orders these COAs by A_0 ranking rules (1)–(8) and just considers the best N actions. Then the one with the largest pseudo utilization is selected. If parameter N is set to 1, A_2 degenerates to A_0. The larger the value of N is, the larger the computing time is, and the higher the volume utilization tends to be. Parameter N is set to 20 for the experiments in this paper, such that A_2 could get a good tradeoff on utilization and time.

4. Computation and discussion

We have implemented and compiled the three caving degree algorithms on Java 2 Platform (Standard Edition, (J2SE) V 1.4.2_08) and run them on a 1.7 GHz personal computer. We will first introduce an example, and then evaluate Algorithms A_1 and A_2 on two groups of public, representative benchmarks: 47 without-orientation-constraint instances and 800 strongly heterogeneous instances. Due to the following reasons, the two groups of $47 + 800 = 847$ instances are rather more difficult.

Earlier methods for packing have always assumed that if there is a good way to pack cuboids, it will start from the bottom and build up the packing in layers, inserting each item so that it is contiguous with what is already packed. This is a natural approach because we expect a method to construct the packing in roughly the order in which we would pack. However, the caving degree method is not a natural approach because it does not necessarily fill up in layers, and it can even fill up all the corners and edges of the container before packing blocks in the center. E.g. there are ten items need to be packed into a (4, 4, 6) container. These items are in size (6, 3, 1), (5, 3, 1), (2, 3, 1), (1, 1, 1), (2, 2, 2) with each size having two items, and they can be packed in any orientation. Experimental results show that A_1 achieves a packing utilization of 93.75% in 47.09 seconds, and A_2 achieves an optimal packing utilization

Table 1
The packing order and position obtained by Algorithm A_2

Order	l_i, w_i, h_i	(x_{i1}, y_{i1}, z_{i1})	(x_{i2}, y_{i2}, z_{i2})
1	6, 3, 1	(0, 0, 0)	(1, 3, 6)
2	5, 3, 1	(0, 3, 1)	(3, 4, 6)
3	6, 3, 1	(3, 1, 0)	(4, 4, 6)
4	5, 3, 1	(1, 0, 0)	(4, 1, 5)
5	2, 3, 1	(1, 1, 0)	(3, 4, 1)
6	1, 1, 1	(0, 3, 0)	(1, 4, 1)
7	2, 3, 1	(1, 0, 5)	(3, 3, 6)
8	1, 1, 1	(3, 0, 5)	(4, 1, 6)
9	2, 2, 2	(1, 1, 1)	(3, 3, 3)
10	2, 2, 2	(1, 1, 3)	(3, 3, 5)

of 100% in 4.25 seconds. The detailed packing order and position for A_2 is shown in Table 1. Fig. 9a shows the packing order and layout at step 8. Fig. 9b shows the packing layout at step 10, which subsequently packs (2, 2, 2) and (2, 2, 2) to the center of the container at step 9 and at step 10.

4.1. The without-orientation-constraint benchmarks

For the without-orientation-constraint benchmark, each of the six orientations of the item should be considered, the searching space is much larger and the computing time increases substantially. So, it is more difficult than the same scale benchmark having orientation constraint.

There are nine sets of benchmark instances for the SCLP in the OR-Library (Beasley, 1990), which is a collection of test data for a variety of operational research (OR) problems. These instances belong to SLOPP according to the classification of Wascher et al. (2007). Instances in set 9 have no orientation constraint and are considered here. Current best result on set 9 for the SCLP is hold by algorithm MFB_L (Lim et al., 2003).

Lim et al. (2003) improved the wall building and layering approach by allowing any surface of the container to be a wall from which to obtain a Multi-Faced Buildup (MFB). MFB combined guillotine cutting for empty spaces and was not concerned whether packed items form a flat layer or not. Then, they used a Look-ahead strategy to improve the MFB (which they called MFB_L) by reserving more than one good solution at each packing step. Lim et al. (2003) computed all the nine sets of benchmark instances for the SCLP, and their results of MFB_L on sets 1–7 are about 3% better than the previous results in (Bischoff and Ratcliff, 1995). For set 9, their average packing utilization of 91% is the best result reported in the literature. There are 47 instances in set 9 and Lim et al., computed nos. 1–44.

The packing results of A_1 on the 47 instances are shown in Fig. 10, where x-axis represents the benchmark index, y-axis represents the volume utilization and the average volume utilization is given at the top, while the basic Algorithm A_0 achieves an average volume utilization of 87.41% with an average computation of 0.34 seconds. More comparisons with MFB_L on instances from no. 1 to no. 44 are shown in Table 2. Algorithm A_1 achieves an average volume utilization of 94.9% on the anterior 44 instances, which is 3.9% better than that of MFB_L. As for the time, since Lim et al. used a small-scale computer while we use a personal computer, it is not easy to compare the computing

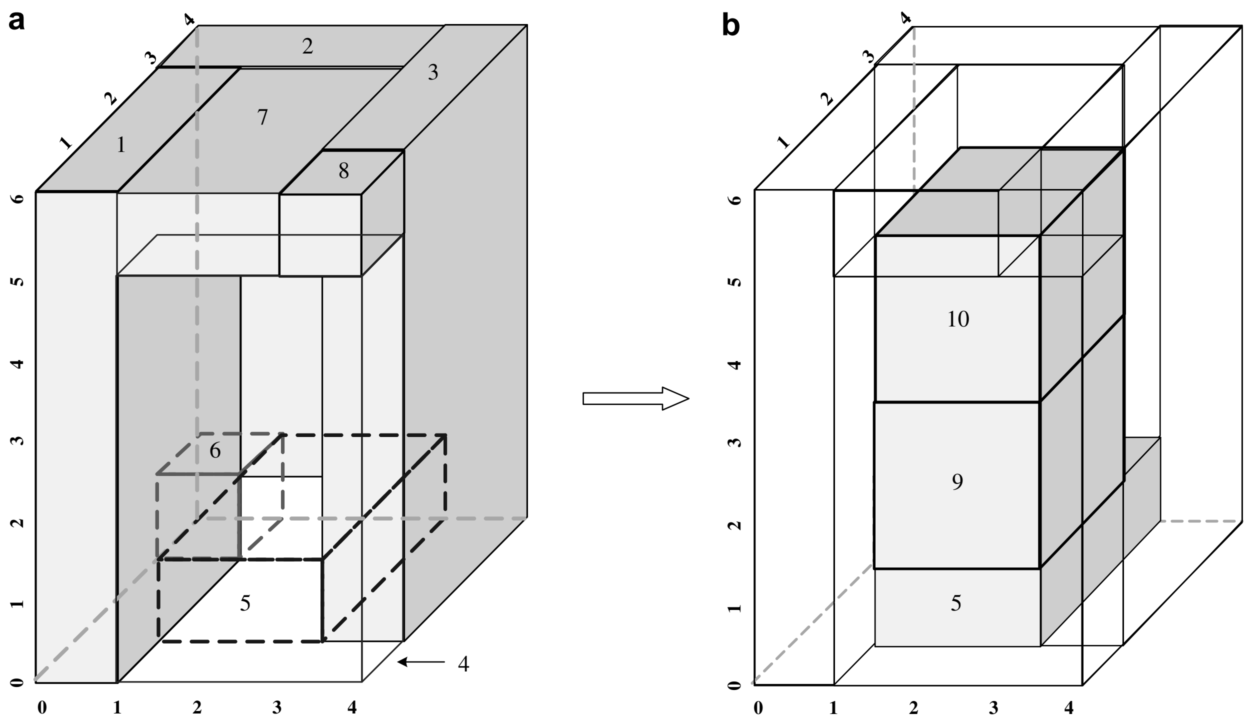

Fig. 9. The packing layout obtained by Algorithm A_2. (a) The layout at step 8; (b) The layout at step 10.

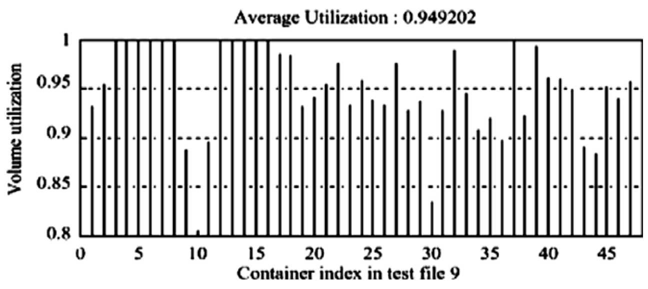

Fig. 10. Results of Algorithm A_1.

Table 2
Comparisons on performance and time

Algorithm	Utilization	Time	Computer	Language
MFB_L	0.910	205s	Unix Alpha 600 MHz	Java
A_1	0.949	804s	Windows 1.7 GHz	Java

time however the average times are in the same order of magnitude. Additionally, Algorithm A_2 also achieves an average volume utilization of 93.04% with an average computation of 74.86 seconds on the 44 instances. Both the volume utilization and the computing time of A_2 are better than that of current best result.

The caving degree approach gives priority to items that are harder to pack. So, it is more suitable for those problems whose optimal solutions pack all the items or for those that the total volumes of the items are approximately equal to the volume of the container. For the 47 instances, a total volume for all the items in an instance is much bigger than that of the container. So, a minority of the items is in the container and a majority of the items is outside for any solution. Even on the cases where only a minority of items can be packed, Algorithms A_1 and A_2 got markedly higher volume utilizations.

4.2. The strongly heterogeneous benchmarks

For the strongly heterogeneous benchmark, items are almost always of different sizes and valid packing actions are far more common than that of the same scale weakly heterogeneous one, the searching space is big and the computing time increases significantly. So, it is more difficult than the same scale weakly heterogeneous benchmark.

Bischoff and Ratcliff (1995) proposed seven sets of benchmark instances for the SCLP, which corresponds to benchmark instances in sets 1–7 of the OR-Library. Afterward, Davies and Bischoff (1999) further proposed eight sets of benchmark instances. So, there are 15 sets of benchmark instances, called BRD1 to BRD15, with each set having 100 instances and each instance having about 130 items. With respect to the item types vary from 3 for BRD1 to 100 for BRD15, the instances vary from weakly heterogeneous to strongly heterogeneous, i.e. items with same size are become less and less. The 1500 instances have been computed by several efficient approaches, including H_BR (Bischoff and Ratcliff, 1995), GA_GB (Gehring and Bortfeldt, 1997), TS_BG (Bortfeldt and Gehring, 1998), HGA_BG (Bortfeldt and Gehring, 2001), PGA_GB (Gehring and Bortfeldt, 2002) and GRASP (Moura and Oliveira, 2005). Their results indicate that the container utilizations are in a decreasing trend when the instances are more and more strongly heterogeneous. This is because many algorithms utilize the weakly heterogeneous characteristic, and bind same size items into a larger cuboid to do the tentative pack, so they can achieve good results for the weakly heterogeneous instances and the container utilizations become low when the instances are very strongly heterogeneous.

We select BRD8 to BRD15, the 800 strongly heterogeneous instances among the 15 sets, to do the computation. They belong to SKP according to the classification of Wascher et al. (2007). Current best record on each set of them is hold by PGA_GB. E.g. for BRD15: current best average utilization achieved by PGA_GB is 85.48%; experiments show that A_0 could achieve an average volume utilization of 84.47% with an average computation of 154 seconds, A_1 could not get result in the first 10 hours, while A_2 could achieve an average volume utilization of 87.73% with an average computation of 3.70 hours. The detailed utilization comparison on BRD8 to BRD15 is shown in Table 3, where the best value for each set appears in bold. It could be found that PGA_GB achieves the best results on three sets: BRD8–BRD10, and A_2 achieves the best results on five sets: BDR11–BRD15, which is 0.01%, 0.74%, 1.22%, 2.01%, and 2.25% higher than current best result respectively. The average utilization on all of the 800 benchmarks is 87.97%, which is 0.28% higher than current best results.

Table 3
Utilization comparison on BRD8 to BRD15 (%)

Algorithm (year)	H_BR(1995)	GA_GB(1997)	TS_BG(1998)	HGA_BG(2001)	PGA_GB(2002)	GRASP(2005)	A_2(2007)
BRD8	80.10	87.52	87.11	89.73	**90.26**	86.13	88.41
BRD9	78.03	86.46	85.76	89.06	**89.50**	85.08	88.14
BRD10	76.53	85.53	84.73	88.40	**88.73**	84.21	87.90
BRD11	75.08	84.82	83.55	87.53	87.87	83.98	**87.88**
BRD12	74.37	84.25	82.79	86.94	87.18	83.64	**87.92**
BRD13	73.56	83.67	82.29	86.25	86.70	83.54	**87.92**
BRD14	73.37	82.99	81.33	85.55	85.81	83.25	**87.82**
BRD15	73.38	82.47	80.85	85.23	85.48	83.21	**87.73**
Average	75.55	84.71	83.55	87.34	87.69	84.13	**87.97**

Algorithm A_2 runs sequentially on a 1.7 GHz personal computer with an average computation of 3.33 hours on the 800 benchmarks, while PGA_GB ran parallel on four 400 MHz Pentium 486PCs and one 33 MHz PC, with a mean computing time of 183.3 seconds (Gehring and Bortfeldt, 2002). Following SiSoftware Sandra we assume for the benefit of A_2 a MIPS (million instructions per second) proportion (400 MHz PC to 1.7 GHz PC) of about 1: 2.5. It could be inferred that the computing time of PGA_GB would be about 0.10 hours ($5 \times 183.3/2.5$ seconds) if it runs sequentially on a 1.7 GHz PC. A conclusion could be drawn that Algorithm A_2 improves current best result slightly on the 800 strongly heterogeneous benchmarks, and the run time proportion is about 1: 33. Though the computing time decouples that of PGA_GB, in our opinion however, relatively to the expense on computing time, how to improve the container volume utilization is more important.

5. Conclusion and future work

Inspired by an old adage “Gold corner, silver side and strawy void” for Chinese Weiqi, and improve the wisdom by a new observation “Maximum value in diamond cave”, we present a new heuristic approach for the single container loading problem. The new approach is intuitive and differs completely from previous heuristics. The novel approach always do the corner occupying action with the largest caving degree that makes the action item as compactly and closely as possible with other packed items. Experiments on two groups of public and difficult benchmark instances, i.e. 47 without-orientation-constraint and 800 BRD strongly heterogeneous instances, compare very favorably against past works in this area. The average volume utilizations of the caving degree approach improve current best records formally reported in the literature by 3.9% and by 0.28% respectively. Besides, it achieves new best records on five sets among the eight sets of BRD benchmarks: BDR11 to BRD15.

We will further improve the performance of the caving degree approach and test all the 15 sets of benchmark instances proposed by Bischoff et al. Other important constraints, such as loading stability, will be considered later. In addition, the caving degree approach may also be useful to other cutting and packing problems, especially for cuboid bin packing or for containers that are not cuboid but are still orthorhombic, e.g. containers with more than six sides and these may include things like the cargo hold of ships or aircrafts. More will be studied on the problem in the future.

Acknowledgements

Thanks are due to the anonymous referees for many instructive suggestions, which have greatly improved this paper. This work was supported by National Natural Science Foundation of China (Grant No. 60773194), NKBRPC (Grant No. G2004CB318000) and China Postdoctoral Science Foundation (Grant No. 20070420174). Special thanks are to the second author’s two brothers, Yu He and Chun-Lin He, who provide enough computers for the computation.

References

Beasley, J.E., OR-Library, http://people.brunel.ac.uk/~mastjjb/jeb/info.html, 1990.

Bischoff, E.E., 2006. Three-dimensional packing of items with limited load bearing strength. European Journal of Operational Research 168, 952–966.

Bischoff, E.E., Ratcliff, M.S.W., 1995. Issues in the development of approaches to container loading. OMEGA: The International Journal of Management Science 23 (4), 377–390.

Bortfeldt, A., Gehring, H., 1998. Applying tabu search to container loading problems. Operations Research Proceedings. Springer, pp. 533–538.

Bortfeldt, A., Gehring, H., 2001. A hybrid genetic algorithm for the container loading problem. European Journal of Operational research 131, 143–161.

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

Davies, A.P., Bischoff, E.E., 1999. Weight distribution considerations in container loading. European Journal of Operational Research 114, 509–527.

Dyckhoff, H., 1990. A typology of cutting and packing problems. European Journal of Operational Research 44, 145–159.

Eley, M., 2002. Solving container loading problems by block arrangement. European Journal of Operational Research 141, 393–409.

Gehring, H., Bortfeldt, A., 1997. A genetic algorithm for solving the container loading problem. International Transactions in Operational Research 4, 401–418.

Gehring, H., Bortfeldt, A., 2002. A parallel genetic algorithm for solving the container loading problem. International Transactions in Operational Research 9 (4), 497–511.

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

Hemminki, J., Leipala, T., Nevalainen, O., 1998. On-line packing with boxes of different sizes. International Journal of Production Research 36 (8), 2225–2245.

Huang, W.Q., He, K., 2007. An efficient algorithm for solving the container loading problem. In: Chen, B., Paterson, M., Zhang, G. (Eds.), Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE 2007), LNCS 4614. Springer-Verlag, Heidelberg, pp. 396–407.

Huang, W.Q., Chen, D.B., Xu, R.C., 2007. A new heuristic algorithm for rectangle packing. Computers and Operations Research 34 (11), 3270–3280.

Lim, A., Rodrigues, B., Wang, Y., 2003. A multi-faced buildup algorithm for three-dimensional packing problems. OMEGA: The International Journal of Management Science 31 (6), 471–481.

Loh, T.H., Nee, A.Y.C., 1992. A packing algorithm for hexahedral boxes. In: Proceedings of the Conference of Industrial Automation, Singapore, pp. 115–126.

Morabito, R., Arenales, M., 1994. An AND/OR graph approach to the container loading problem. International Transactions in Operational Research 1, 59–73.

Moura, A., Oliveira, J.F., 2005. A grasp approach to the container-loading problem. IEEE Intelligent Systems 20 (4), 50–57.

Pisinger, D., 2002. Heuristics for the container loading problem. European Journal of Operational Research 141 (2), 382–392.

Wascher, G., Haubner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 183 (3), 1109–1130.