Research on Column Generation Algorithm for Solving Three-Dimensional Packing Problem

Meng Wang, Chengxia Liu, Minling Zhu,Yaokun Shi

***Abstract*—Three-dimensional packing problem refers to how to put a series of items of known size into a given container in three-dimensional space, so that the space utilization of the container is the highest. This problem has a wide range of applications in logistics, warehousing, manufacturing and other fields. We first solve the relevant mathematical model through a column-generation heuristic algorithm to obtain a layer containing a set of items. These layers are then stacked together into a container using a simple heuristic algorithm. After experiments, our method has a good loading rate and can complete the loading of items in a very short time.**

***Index Terms*—Three-dimensional packing, column generation algorithm, layer, heuristic algorithm.**

I. INTRODUCTION

The three-dimensional packing problem is defined as placing a series of items of known size into a given container. According to Martello et al. [1], it can be subdivided into the following three questions:

The first is to place a given item into one or more boxes so that the total value of the items in the box is maximum. The second is to pack the item into a box with a fixed width and depth, but an unrestricted height, and minimize the height position of the top item in the box. The third is to pack a set of items into an infinite number of boxes with a fixed size, with the goal of minimizing the number of boxes used.

This paper focuses on the second and third of the above problems. In the second problem, the goal is to pack all the items in a box of infinite height so that the height of the box is minimized. In the third type of problem, the goal is to pack all items in an unlimited number of boxes, but use the least number of boxes. The packing problem requires that items do not overlap and are perpendicular to the edge of the box. The second problem is important in this article and can be used as the basis for a column generation framework to generate valuable layers and solve the third problem.

For the third problem, there are two kinds of related research methods: exact formula and heuristic algorithm. For the exact algorithm, Chen et al. [2] proposed the first mixed integer programming formula based on the relative positions of items. Wu et al. [3] later extended this algorithm to allow objects to have direction. Hifi et al. [4] added several lower limits based on linear programming relaxation and effective inequalities to reduce the solution time. Junqueira et al. [5] considered the stability and load-bearing of items, and proposed a framework to group items. Paquay et al. [6] also considered the bearing capacity, direction, distribution of center of gravity and stability of objects. Martello et al. [1] proposed an enumeration search algorithm based on placement points. Later, Martello et al. [7] improved it by deciding which items to pack into which box at each node of the search tree, while ignoring the placement, so that the solution speed was faster.

For heuristic methods, compared with exact algorithms, they can solve a larger amount of data that is closer to the real situation. Heuristic algorithms generally have two types, one is based on the placement point method. Martello et al. [1] first introduced a placement point method for two-dimensional packing problems based on placement points. A placement point is the intersection of the three planes on the right, back, and top of the items already in the box. Crainic et al. [8] extended this method by introducing the concept of extreme value points. The extreme point is generated by the projection of the right back and top of the item already placed in the box. Increased the number of placement points by using projection. Zhu et al. [9] proposed a method of space debris consolidation, which combines space debris between placed items by pushing them to the edge.

Based on the extreme value point method, Crainic et al. [8] proposed the first-fit decline heuristic and best-fit decline heuristic, both of which use the items ordering rules based on height-volume and volume-height. After the items are sorted, the first-fit decline-heuristic places the next item in the first opened bin and reopens a new bin if it does not fit. The best-fit decline heuristic evaluates extreme points based on a value function, and then places the item on the best available extreme point. Zhu and Lim[10] propose a greedy prospective tree search algorithm that first generates item blocks and places them in a corner of the box, which then generates new spare space and new corners to place items in. Zhu et al. [9] proposed a heuristic method based on extremal points and spatial defragmentation, where items are placed on extremal points and then merged in space by pushing the items to the edge. They further improved this scheme by using spatial defragmentation technology to transfer items from one bin to another. Faroe et al. [11] proposed a heuristic algorithm to guide local search by moving items in any orthogonal direction or moving them to the same location in another bin if overlap is allowed. In order to find a viable place to place an item, they minimize the overlap between items and use the penalty function to select. Lodi et al. [12] proposed a tabu search heuristic in which each bin is first loaded with one item and the neighborhood is defined by selecting a bin and placing all its items into other bins. Items are layered in two steps, using a high-priority and area-first strategy in steps one and two, respectively, and then at the end of each step, solving the

Manuscript received June 12, 2024

one-dimensional problem of packing each layer into the box. Crainic et al. [8] proposed another tabu search algorithm TS2PACK, which first determines the items to be packed into each bin, and second determines the position of each item in the bin. They use the first adaptive decline heuristic algorithm based on extremal points to find an initial feasible solution. In the first step, the neighborhood is constructed by swapping items in different chests or moving items to another chests. In the second step, they find a viable packing solution in a box by swapping the relative positions of items.

II. MATHEMATICAL MODELING

A. Basic concept

Firstly, some basic concepts and data structures are given: the items in the three-dimensional box are cuboids orthogonal to coordinate axes, which mainly have the following properties: item.x,item.y,item.z respectively represent the coordinates of the left and lower vertices of the item. Item. dimensions represents the length, width, and height of the item itself.

Simple blocks are stacked by the same type of boxes facing the same direction, there is no gap between the boxes, and the stacking result must be exactly a cuboid.

Composite blocks are obtained by continuously compounding simple blocks, which are defined as follows:

(1) Simple blocks are the most basic composite blocks.

(2) There are two complex blocks a and b. Composite block c can be compounded in three ways: in the width direction, in the depth direction, and in the height direction. The size of c is the smallest cuboid containing a and b.

A layer is defined as a two-dimensional arrangement of items within the horizontal plane of a box, with no items stacked on top of each other. The definition of a layer does not allow multiple items to be stacked, which results in an uncompact arrangement of items. Thus, the concept of composite blocks is defined. A composite block is a group of individual items that are tightly stacked together. A composite block is a stable arrangement that provides support for items placed on top, as if they were individual items.

B. Mathematical model

For problem 3, the following mathematical model is established:

$$\min \sum_{b\in B} t_b$$

$$s.t. \sum_{l\in L}\sum_{s\in CI} f_{si} z_{sl} \alpha_l \geqslant 1 \qquad i \in I \qquad (1)$$

$$\sum_{l\in L} o^l u_{lb} \leqslant H \qquad b \in B \qquad (2)$$

$$u_{lb} \leqslant t_b \qquad b \in B, l \in L \qquad (3)$$

$$u_{lb} \leqslant \alpha_l \qquad b \in B, l \in L \qquad (4)$$

$$\alpha_l, u_{lb}, t_b \in \{0,1\} \qquad b \in B, l \in L$$

Assume that the set of all unpacked items is $i \in I$ and the set of usable boxes is $b \in B$. A new composite block composed of simple blocks and composite blocks is treated as a new single item and added to set I to form a new set CI. That is, CI contains both the original item $i \in I$ and the composite block $s \in SI$. The 0-1 variables z_{sl} and f_{si} indicate whether complex block s is in layer l and whether complex block s contains item i, respectively. The continuous variable o^l represents the height of layer l. The 0-1 variable t_b indicates whether box b is used. The 0-1 variable u_{lb} indicates whether to put the current layer into box b. The 0-1 variable α_l indicates whether layer l is selected.

Constraints (1) ensure that each item is included, and constraints (2) ensure that the sum of the heights of all layers placed in the box is not higher than the height of the box. Constraints (3) and (4) ensure that a layer can only be put into a box if that layer is selected and box b is used.

We use problem 2 to generate the layers required by problem 3. The mathematical model established for problem 2 is as follows:

$$\min \sum_{l\in L} o^l$$

$$s.t. \sum_{l\in L}\sum_{s\in CI} f_{si} z_{sl} = 1 \qquad i \in I \qquad (5)$$

$$o^l \geq h_s z_{sl} \qquad s \in CI, l \in L \qquad (6)$$

$$0.9WD \leq \sum_{s\in CI} w_s d_s z_{sl} \leq WD \qquad l \in L \qquad (7)$$

$$x_{sj} + x_{js} + y_{sj} + y_{js} \geq z_{sl} + z_{jl} - 1 \qquad j > s, s, j \in CI, l \in L \qquad (8)$$

$$x_{sj} + x_{js} \leq 1 \qquad j > s, s, j \in CI \qquad (9)$$

$$y_{sj} + y_{js} \leq 1 \qquad j > s, s, j \in CI \qquad (10)$$

$$c_s^1 + w_s \leq c_j^1 + W\, x_{sj} \qquad s \neq j, s, j \in CI \qquad (11)$$

$$c_s^2 + d_s \leq c_j^2 + D\, y_{sj} \qquad s \neq j, s, j \in CI \qquad (12)$$

$$0 \leq c_s^1 \leq W - w_s \qquad s \in CI \qquad (13)$$

$$0 \leq c_s^2 \leq D - d_s \qquad s \in CI \qquad (14)$$

$$x_{sj}, y_{sj} \in \{0,1\} \qquad s \neq j, s, j \in CI \qquad (15)$$

$$z_{sl} \in \{0,1\} \qquad s \in CI, l \in L \qquad (16)$$

$$o^l \geq 0 \qquad l \in L \qquad (17)$$

Suppose item i has a width of w_i, a depth of d_i, and a height of h_i, and complex block s has a width of w_s, a depth of d_s, a height of h_s, and an infinite number of boxes $b \in B$, which have a width of W, a depth of D, and a height of H. Suppose that the continuous variables c_i^1 and c_i^2 represent the coordinates of the width and depth of the left, back and bottom vertices of item i, respectively. If item i is in front of item j in both width and depth directions, then the 0-1 variables x_{ij} and y_{ij} are both 1, otherwise they are both 0. If the complex block s is contained in layer l, then the 0-1 variable z_{sl} takes on a value of 1. The continuous variable o^l represents the height of layer l.

Constraint (5) ensures that each item i can only exist in one layer (an item i can only be contained in a composite block, which can also only be in one layer). Constraint (6) ensures that the height of all composite blocks in the layer cannot exceed the height of the layer. Constraint (7) requires that the sum of the bottom area of all composite blocks in each layer cannot be less than 90% of the bottom area of the layer. We

hope that the composite blocks in the layer can occupy the layer as much as possible, and of course, the sum of the bottom area of all composite blocks cannot exceed the bottom area of the layer. The constraint (8) ensures that any two composite blocks have a relative position and cannot completely overlap. Constraints (9) and (10) to ensure that any two objects i and j in width and depth direction or items before items j, i or j in the items before i, or both in the vertical direction projection but belong to different layers in the same position, but not both in the behind or in front of the other party.

We use Figure 1 to illustrate constraints (11) and (12). If item j is ahead of item s in the width direction, then the width coordinates of the left, back, and bottom vertices of item j must be greater than the width coordinates of the left, front, and bottom vertices of item s. In other words, constraints (11) and (12) ensure that items j and s do not overlap. Constraints (13) and (14) ensure that all items cannot exceed the boundary of the box.

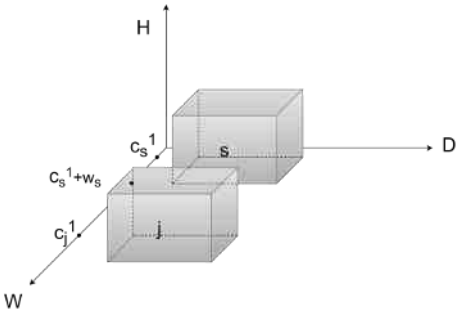

Figure 1: Item j is located before item s

As a large mixed integer programming problem with many 0-1 variables, the solution time of problem 2 will increase exponentially as the number of variables and constraints in the problem increase. The solution time of a mixed integer programming problem with n variables may be an integer multiple of the solution time of a linear programming problem with n constraints. Therefore, for large-scale mixed integer programming problems, sometimes we do not seek to find a global optimal solution, but seek an acceptable approximate solution in a certain time. To this end, we use the column generation algorithm to improve problem 2.

Column generation algorithm is an efficient method to solve large-scale linear programming problems, especially when the number of variables increases explosively with the increase of problem size. The basic idea of this algorithm is to start with a small model and gradually increase the variables and constraints to achieve the purpose of finally solving the original problem. Compared with traditional direct solving methods, column generation algorithm has obvious advantages in dealing with large-scale problems. By gradually increasing variables, it avoids directly dealing with a huge model containing all variables, thus reducing the complexity of the problem and improving the solving efficiency.

The master problem model for problem 2, constructed using the column generation algorithm, is as follows:

$$\min \sum_{l\in L} o^l \alpha_l$$
$$s.t. \sum_{l\in L}\sum_{s\in CI} f_{si} z_{sl} \alpha_l \geq 1 \qquad i \in I$$
$$\alpha_l \in \{0,1\} \qquad l \in L$$

In the case that the variables are integers, the optimal solution cannot be obtained using the column generation algorithm, so the master problem is relaxed as follows:

$$\min \sum_{l\in L} o^l \alpha_l$$
$$s.t. \sum_{l\in L}\sum_{s\in CI} f_{si} z_{sl} \alpha_l \geq 1 \qquad i \in I$$
$$\alpha_l \geq 0 \qquad l \in L$$

Due to the large scale of variables in problem 2, the feasible solution cannot be obtained in a short time by direct solution. The column generation algorithm can first ignore some variables, solve a small linear programming problem on the subset L' of the layer, and take its solution as the initial feasible solution. Therefore, the constrained master problem model of problem 2 is as follows:

$$\min \sum_{l\in L'} o^l \alpha_l$$
$$s.t. \sum_{l\in L'}\sum_{s\in CI} f_{si} z_{sl} \alpha_l \geq 1 \qquad i \in I \qquad (18)$$
$$\alpha_l \geq 0 \qquad l \in L'$$

Assuming that λ_i, $i \in I$ is the dual variable corresponding to constraint (18), then the dual problem of the main problem of this constraint is:

$$\max \sum_{i\in I} \lambda_i$$
$$s.t. \sum_{i\in I}\sum_{s\in CI} \lambda_i f_{si} z_{sl} \leq o^l \qquad l \in L'$$
$$\lambda_i \geq 0 \qquad i \in I$$

The number of tests for the new layer is $o^l - \sum_{i\in I}\sum_{s\in CI} \lambda_i f_{si} z_{sl}$, and the sub-problems are:

$$\min \sum_{l\in L} \left(o^l - \sum_{i\in I}\sum_{s\in CI} \lambda_i f_{si} z_{sl} \right) \qquad (19)$$
$$s.t.\ (6) - (17)$$

For minimization problems, the optimal solution of the dual problem is less than or equal to the optimal solution of the original problem, that is, the dual problem obtains a lower bound of the original minimal problem, so even if the subproblem cannot generate new columns, it provides a lower bound of the original principal problem.

If the subproblem is solved directly, it will still take a lot of time. Since the objective function of the subproblem is only a function of the items in the layer (variables z_{sl}) and not the position (variables c_s^1 and c_s^2), we can first ignore the position constraints (8) - (17) and look for the set of items that makes the number of tests smallest. If the number of tests is non-negative, the column generation algorithm is terminated; conversely, if the number of tests is negative, the column

generation algorithm can continue to iterate and solve the subproblem for the set S containing complex blocks at this time, instead of solving for the set I containing only a single block. Until a viable solution is found to check if there is a viable place for S in the layer, if there is, the algorithm terminates and the column is added to the master problem. Otherwise, the constraint $\sum_{s \in S} z_{sl} \leqslant |S| - 1$ is added to the relaxation subproblem and the iterative algorithm continues.

III. Algorithm Design

A. Algorithmic pseudo-code

In algorithm 1, all layers are obtained by solving the mathematical model through the solver. Since the composite blocks used to build the layers are repeated, the layer that does not contain repeated blocks for the first time is selected through algorithm 1. Sort the layers in descending order of density, choosing the higher density layer, which will make the box more stable. The item in the layer is detected, and if the item is not contained by the previous layer, the layer is added to the collection .

Algorithm 1: Layer selection

Input: Layer set L
Output: SL

```
Set SL = ∅
for layer l ∈ L do
    for item i in layer l do
        if item i is already included then
            Check the next layer
        end
    end
    Set SL ← SL ∪ l
end
return SL
```

In algorithm 2, B represents the set of built bins and SL represents the set of selected layers. The parameters h_b and h_l represent the height of box b and layer l, respectively. H is the maximum height of the box.

Not all items are included in layers, on the one hand because layers less than 90% density are discarded, on the other hand, the height of some items may be too different from the height of others to fit into a layer in a stable way. So algorithm 2 is just used to build most of the boxes, and for the rest of the items, we just need to put them on top of the box.

Algorithm 2: Bin construction algorithm

Input: Set SL
Output: B

```
Set B = ∅
Sort the layers in the collection SL
in descending order of density
Open an empty box b make B ← B ∪ b
for layer l ∈ SL do
    for bin b ∈ B do
        if h_b + h_l ⩽ H then
            Place layer l into b
        end
    end
    if Layer l cannot fit into any b ∈ B then
        Open a new empty box n and place l into n
        B ← B ∪ n
    end
end
return B
```

B. Experimental Results

The data comes from the actual packing data of Bozheng Logistics company, the container specification is 40HQ, and it is divided into pure bulk cargo and pure carton goods according to different goods. Consignment of mixed cargo; Pure pallet, pure wooden case and other large goods.

Some important features of the packed goods are the ratio of the depth and height of the goods to its width, as well as the occurrence frequency of the volume and weight of the goods. The statistics of the Tosan-mixed packing data in the data set are shown in Figure 2:

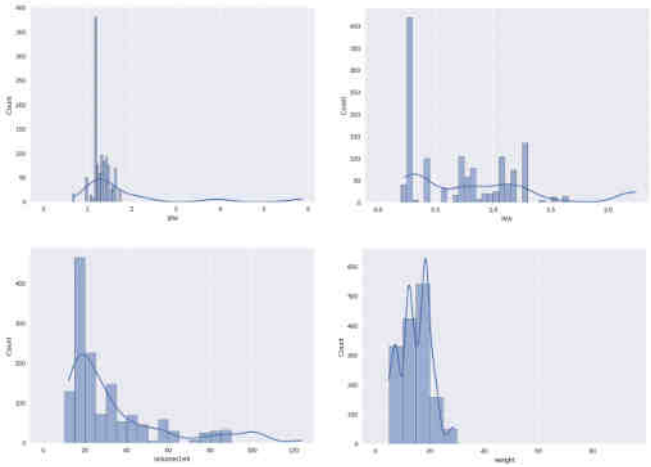

Figure 2: Item characteristics

According to the figure, there are a large number of the same items in the data, and their probability density is chaotic, that is, there is no rule to follow the characteristics of these items, and the experiment results are more universal with this real data set.

The enterprise data is boxed and the results are shown in Table 1:

Table 1: Average loading rate

Type of goods	Manual loading rate	Algorithm loading rate
Pure bulk cargo, cartons	91%	93.8%
Pure pallets, wooden cases	71%	82%
Mixed pallet bulk cargo	85%	86.6%

Compared with the manual loading rate, the algorithm has a certain improvement in the packing effect of pure bulk cargo and pure pallet goods, but it needs to be improved on the mixed goods.

IV. ConClusion

For the three-dimensional packing problem, a model of column generation algorithm is established, and a simple heuristic algorithm is used to pack a set of data, and a good loading rate is obtained. Compared with other algorithms, the column generation algorithm shows better optimization performance and faster convergence speed. However, the algorithm may face computational resource challenges in handling extreme cases and is sensitive to the quality of the initial solution. Future research could explore the parallelization of algorithms to improve their ability to handle large-scale problems, as well as combine machine learning techniques to improve the quality of initial solutions. In addition, the application of column generation algorithms to practical industrial problems, such as automated warehouse design and management, will be a promising research direction. In general, this paper provides a powerful tool for the solution of three-dimensional packing problem, and lays a foundation for future research.

References

[1] Martello S,Pisinger D,Vigo D.The three-dimensional bin packing problem[J].Oper.Res,2000,48(2):256~267
[2] Chen C,Lee SM,Shen Q.An analytical model for the container loading problem[J].Eur.J.Oper.Res,1995,80(1):68~76.
[3] Wu Y,Li W,Goh M,de Souza R.Three-dimensional bin packing problem with variable bin height[J].Eur.J.Oper.Res,2010,202(2):347~355.
[4] Hifi M,Kacem I,Negre S,Wu L.A liner programming approach for the three-dimensional bin-packing problem[J].Electronic Notes Discrete Math,2010,36:993~1000.
[5] Junqueira L .Three-dimensional container loading models with cargo stability and load bearing constraints[J].Comput.Oper.Res,2012,39(1):74~85.
[6] Paquay C,Schyns M,Limbourg S.A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application[J].Internat.Trans.Oper.Res,2016,23(1-2):187~213.
[7] Martello S,Pisinger D,Vigo D,Boef ED,Korst J.Algorithm 864:General and robat-packable variants of the three-dimensional bin packing problem[J].ACM Trans.Math.Software,2007,33(1):Article No.7.
[8] Crainic TG,Perboli G,Tadei R.Extreme point-based heuristics for three-dimensional bin packing[J].Comput,2008,20(3):368~384.
[9] Zhu W,Zhang Z,Oon WC,Lim A.Space defragmentation for packing problems[J].Eur.J.Oper.Res,2012b,222(3):452~463.
[10] Zhu W,Lim A.A new iterative-doubling Greedy-Lookahead algorithm for the single container loading problem[J].Eur.J.Oper.Res,2012,222(3):408~417
[11] Faroe O,Pisinger D,Zachariasen M.Guided local search for the three-dimensional bin-packing problem[J].INFORMS J.Comput,2003,15(3):267~283
[12] Lodi A,Martello S,Vigo D.Heuristic algorithms for the three-dimensional bin packing problem[J].Eur.J.Oper.Res,2002,141(2):410~420