An Approximate Decomposition Based Approach to the Variable-Sized 2-Dimensional Bin Packing Problem

Xiaojuan He
Supervised by Dr Sergey Polyakovskiy
Deakin University

Contents

Abstract

In manufacturing industries, the process of cutting variable-sized materials into required different sized smaller pieces is a critical step in production. This task involves solving the 2-dimensional bin packing problem, where items of different sizes and quantities need to be packed into a limited space. Finding optimal cutting patterns that minimize material waste and maximize production efficiency is of vital importance in industrial practice. In recent years, researchers have developed various optimization algorithms to address this problem, ranging from exact methods to heuristic approaches. However, there is still room for improvement, and new techniques are continuously being developed to further enhance the cutting efficiency and reduce costs. In this paper, we propose an approximate decomposition-based approach to solve the variable-sized 2-dimensional bin packing problem (VS-2D-BPP) efficiently. We adopt the Dantzig-Wolfe approach, which is a linear programming-based formulation that decomposes the problem into smaller subproblems. The idea is if we have a set of feasible packings, we can reduce the problem to select a subset of the packings that covers each item at least once and also minimize the total cost. However, there exist exponentially many feasible single bin packings that make this model too big to solve. To overcome this limitation, we propose a matheuristic approach that applies Mixed Integer Programming to find patterns heuristically. Specifically, we search for a set of patterns that can efficiently cover the items in the problem instance, and use these patterns to reduce the number of packings that need to be considered in the Dantzig-Wolfe model.

1 Introduction

The two-dimensional variable-sized bin packing problem (VS-2D-BPP) is an important optimization problem that arises in various industries, such as guillotine glass cutting and furniture production. The problem requires selecting a collection of bins of minimum total cost so that all items are packed without overlap and exceeding the boundaries of the bins.

In the manufacturing industry, optimizing the use of materials is crucial to keep production costs low and maintain competitiveness in the market. One of the most challenging problems that manufacturers face is the 2-dimensional bin packing problem, where they need to pack a variety of rectangular items with different sizes into a limited number of rectangular sheets of materials, such as paper, wood, glass or metal.

Consider the case of a furniture manufacturer that specializes in producing drawers of various sizes. The manufacturer receives orders for different quantities and sizes of drawers every day, and they need to cut the sheets of materials into the correct sizes and shapes to fulfill these orders. However, due to the varying sizes and shapes of the drawers, it is not always easy to find the optimal cutting pattern that minimize the waste of materials.

The problem is challenging due to the fact that each item can have many alternative positions within the bins, and there can be many solutions with equal costs. In this paper, we propose an approximate decomposition-based approach to solve the VS-2D-BPP efficiently.

1.1 Statement of Authorship

The work on this project was divided among the team as follows:

- Xiaojuan He was responsible for conducting the matheuristic part of the project.
- Adam McGregor, who is also a student at Deakin University, conducted the bin decomposition portion of the project.
- Dr Sergey Polyakovskiy provided supervision throughout the project, offering guidance and insights as needed.

2 Problem Formulation

The VS-2D-BPP is defined by a set T of m rectangular bin types, where each bin of type t is characterized by its width W_t, height H_t, and cost C_t for $t = 1, \ldots, m$. Additionally, a set I of n items is to be packed into/cut from these bins, where each item i is associated with its width w_i, height h_i, and area $a_i = w_i h_i$ for $i = 1, \ldots, n$. The objective is to select a collection of bins of the minimum total cost such that all items are packed without overlap and exceeding the boundaries of the bins.

There are many formulations for the VS-2D-BPP. In this paper, we adopt the Dantzig-Wolfe decomposition[1], which is a linear programming-based formulation that decomposes the problem into smaller subproblems.

This formulation has been shown to be effective in solving large-scale instances of the problem, and can provide valuable insights into the structure and properties of the solution space. The basic idea is if we have a set of feasible packings, we can reduce the problem to select a subset of packings which covers each item for at least once and minimize the total cost. In the following, we provide a detailed description of the Dantzig-Wolfe formulation for the VS-2D-BPP, and discuss its strengths and limitations.

In the formulation, we define P_t as a set of all feasible packings of bin type t and $P = \bigcup_{t=1}^{m} P_t$ to denote all feasible packings. Let x_p be a binary variable; $x_p = 1$ if packing $p \in P_t$ is selected in the solution, and let δ_i^p be a binary constant; $\delta_i^p = 1$ if packing $p \in P$ contains item $i \in I$. The Dantzig-Wolfe formulation of the problem is as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{m} \sum_{p \in P_t} C_t x_p && (1) \\ \text{subject to} \quad & \sum_{t=1}^{m} \sum_{p \in P_t} \delta_i^p x_p \geq 1, \quad i \in I && (2) \\ & x_p \in \{0,1\} \quad p \in P_t && (3) \end{aligned}$$

Eq. (1) defines the objective function of the MIP, which minimizes the total cost of the solution. This total cost is obtained by summing the cost of selected packing. Eq. (2) ensures that each item is packed in at least one of the selected packings in the solution. Eq. (3) declares the types of the decision variable.

One of the advantages of the Dantzig-Wolfe method is that the LP relaxation (i.e., $0 \leq x_p \leq 1$) of this formulation is tight. However, there exist exponentially many feasible single bin packings that make this model too big to solve. Fortunately, we only need to find those packings of P that constitute an optimal solution.

3 Dual Bounds

The Dantzig-Wolfe formulation provides an efficient approximate approach to find patterns heuristically in a reasonable time and still obtain a tight primal bound. Moreover, we can achieve a tight dual bound by applying the Column Generation approach. Specifically, we can follow the steps proposed by Pisinger and Sigurd[2] as follows:

Step 1. Solve the dual problem of the Restricted Master Problem (RMP):

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in I} \pi_i && (4) \\ \text{subject to} \quad & \sum_{i \in I} \delta_i^p \pi_i \leq C_t, \quad t = 1, \dots, m, \quad p \in P_t && (5) \\ & \pi_i \geq 0, \quad i \in I && (6) \end{aligned}$$

Eq. (4) defines the objective function of the dual problem of the RMP. It maximizes the sum of dual variables π_i over all items i in I. Eq. (5) and (6) are the dual constraints of the RMP. They ensure that the sum of dual variables of items in each packing p does not exceed the capacity of the corresponding bin t.

Step 2. Solve the Pricing Problem: Find a feasible packing for each type t that maximizes

$$Z_t = \sum_{i \in I} \delta_i^p \pi_i \tag{7}$$

The pricing problem is a 2D single bin packing problem that we solve exactly via a constraint programming approach. If $C_t - Z_t \geq 0$, then the linear program of the Dantzig-Wolfe model is solved to optimality and we have found the dual bound. Otherwise, we have found a violating constraint that we add to the RMP by extending the set P_t. We then need to resolve the RMP (Step 1) and the pricing problem again.

While iteratively solving the dual problem and the pricing problem may lead to finding feasible packings and improving the total cost, the process can be very time-consuming. As such, to address this issue, we propose to develop a matheuristic approach that can efficiently solve the problem at hand.

4 Matheuristic approach

Our approach aims to achieve a futuristic vision of reserving a free space for all unpacked items while exploring possible item-to-bin assignments. This allows us to focus on only potentially feasible packing alternatives, make the packing procedure less greedy, and decrease the number of iterations necessary to pack all items.

To guide the search process and enable current decisions to account for their impact on future ones, we propose using mixed-integer programming (MIP) augmented with redundant feasibility constraints. These constraints prohibit partial solutions that would lead to infeasible solutions in the future and enforce bounds on the objective function value, imposing an upper bound on the total packing cost.

Moreover, we aim to forbid the use of new bins when the sum of their costs plus the cost of already used bins exceeds the bound. This approach allows us to achieve a feasible packing and reduce the number of iterations required to find the optimal packing solution.

The matheuristic algorithm, as shown in Algorithm 1, is designed to solve the VS-2D-BPP through an iterative process that involves exploring a set of feasible packings and adjusting their configuration to optimize the cost function. The algorithm takes a VS-2D-BPP as input and initializes a feasible packings dataset using Dantzig-Wolfe decomposition. Then, it defines ceiling and floor ratios based on the dataset to identify the optimal packings and those that need to be released. Next, the algorithm generates a Pool of all non-optimal packings and iterates over all possible combinations of packings in the pool. For each combination, the algorithm computes the current cost of released packings and tentatively assigns the released items to new packings to generate a new tentative cost. If the tentative cost is less than the current cost, the tentative assignments guide the construction of a feasible packing. If any optimal packing found for the released items, the algorithm explores the Pool recursively to find better packings until no more improvements can be made.

As the number of combinations in a set of size n is 2^n, it can be computationally expensive to explore all possible combinations. Therefore, it is recommended to incorporate a timer parameter to halt the exploration process after a certain amount of time has passed.

Figure 1 illustrates one of the key steps in our matheuristic approach, which tentatively assigns the free items to bins while keeping the partial feasible solution fixed.

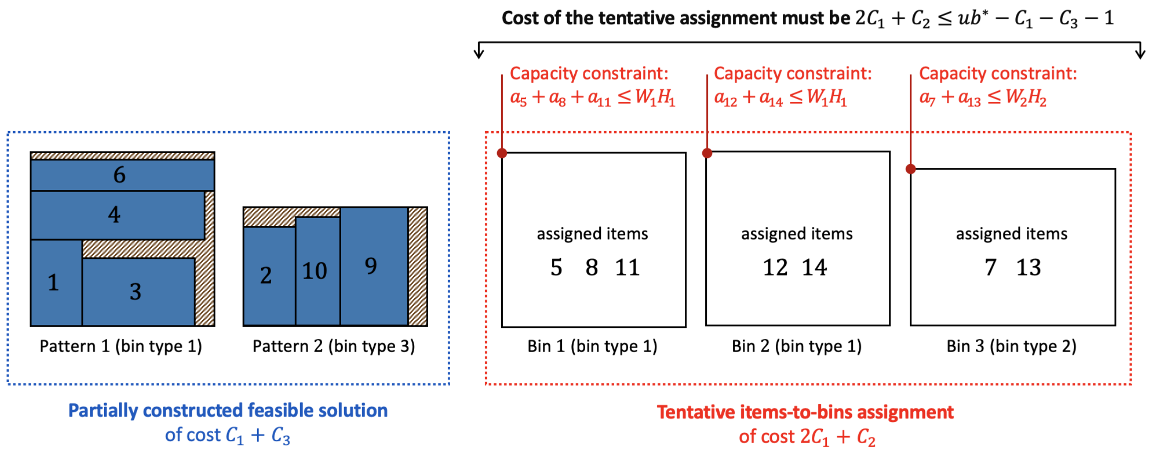
Cost of the tentative assignment must be $2C_1 + C_2 \leq ub^* - C_1 - C_3 - 1$
Capacity constraint: $a_5 + a_8 + a_{11} \leq W_1H_1$
Capacity constraint: $a_{12} + a_{14} \leq W_1H_1$
Capacity constraint: $a_7 + a_{13} \leq W_2H_2$
6
4
1
3
2
10
9
Pattern 1 (bin type 1)
Pattern 2 (bin type 3)
assigned items 5 8 11
assigned items 12 14
assigned items 7 13
Bin 1 (bin type 1)
Bin 2 (bin type 1)
Bin 3 (bin type 2)
Partially constructed feasible solution of cost $C_1 + C_3$
Tentative items-to-bins assignment of cost $2C_1 + C_2$

Figure 1: An example of matheuristic approach's repacking step

To ensure that the results of our algorithm can be used for future optimization purposes, we save all the feasible packings we found to our feasible packing dataset. These packings will be used in the Dantzig-Wolfe decomposition technique, allowing for even more efficient optimization in future iterations.

Algorithm 1 Matheuristic Algorithm

```
1:  function SOLVE(Problem)
2:      $feasible\_packings \leftarrow DW.init(Problem)$        ▷ Initialize feasible packings dataset with Dantzig Decomposition.
3:      $ceiling\_ratio, floor\_ratio \leftarrow compute\_ratios(feasible\_packings)$   ▷ Define ceiling and floor ratios based on the dataset.
4:      $Pool \leftarrow []$
5:      for $packing$ in $feasible\_packings$ do
6:          if $packing.ratio > ceilingRatio$ then
7:              $Fix(packing)$
8:          else if $packing.ratio < floorRatio$ then
9:              $Release(packing)$
10:         else
11:             $Pool.append(packing)$
12:         end if
13:     end for
14:     EXPLORE($Pool, ceilingRatio$)
15: end function
16: function EXPLORE(Pool, ceilingRatio)
17:     $n \leftarrow Pool.size$
18:     for $i$ in range$(1, n)$ do
19:         for $comb$ in all_combinations$(Pool, i)$ do
20:             RELEASEALL($comb$)
21:         end for
22:         $currentCost \leftarrow$ GETCURRENTCOST($released\_packings$)
23:         $tentativeCost, tentativeAssignment \leftarrow$ TENTATIVEASSIGN($released\_items$)
24:         if $tentativeCost < currentCost$ then
25:             PACK($tentativeAssignment, tentativePacking.items$)
26:         end if
27:         UNRELEASEALL($comb$)
28:     end for
29: end function
```

By setting an upper bound ub^* on the cost of new solution, we can obtain a tentative assignment of items to bins by solving the following MIP. Although ensuring optimality for this model is not required, it provides a useful initial configuration. To speed up the overall computation time, we may tolerate an MIP optimality gap of approximately 5%.

To formulate the model, we define two sets of decision variables: x_{ib}^t and y_b^t. The binary variable x_{ib}^t indicates whether item i is assigned to bin b of type t, while the binary variable y_b^t indicates whether bin b of type t is used in the solution.

To restrict the items that can be assigned to a bin of type t based on their size and the capacity of the bin, we define the set I^t, as shown in Eq. (8).

$$I^t = \{i \in I : w_i \leq W_t \wedge h_i \leq H_t\} \tag{8}$$

To determine the number of bins of type t that will be used in the packing solution, we set the value of n^t using Eq. (9).

$$n^t = \min\left(\frac{\lfloor ub^* - 1 \rfloor}{C^t}, |I^t|\right) \tag{9}$$

$$\begin{align}
\text{minimize} \quad & ub = \sum_{t=1}^{m} \sum_{b=1}^{n^t} C_t y_b^t \tag{10} \\
\text{subject to} \quad & \sum_{i \in I^t} w_i h_i x_{ib}^t \leq W_t H_t y_b^t, \quad b = 1, \ldots, n^t, \quad t \in T \tag{11} \\
& \sum_{t : i \in I^t} \sum_{b=1}^{n^t} x_{ib}^t = 1, \quad i \in I \tag{12} \\
& x_{ib}^t + x_{jb}^t \leq 1, \quad i, j \in I^t, \quad (w_i + w_j > W_t) \wedge (h_i + h_j > H_t) \tag{13} \\
& ub \leq ub^* - 1 \tag{14} \\
& y_b^t \leq y_{b-1}^t, \quad b = 2, \ldots, n^t, \quad t \in T \tag{15} \\
& x_{ib}^t \in \{0, 1\} \tag{16} \\
& y_b^t \in \{0, 1\} \tag{17} \\
& \tag{18}
\end{align}$$

In Eq. (10), the objective function minimizes the total cost of new bins used in the solution, where C_t denotes the cost of a bin of type t. Eq. (11) ensures that the sum area of items assigned to a bin do not exceed the area of the bin. Eq. (12) ensures that each item is assigned to exactly one bin. Eq. (13) ensures that two items cannot be assigned to the same bin if their combined weight and height exceed the capacity of the bin. Eq. (14) sets an upper bound on the cost of new solution. Eq. (15) enforces the use of bins in increasing order of their indices, which breaks symmetries and simplifies the search process by avoiding the consideration of equivalent solutions that differ only in the ordering of the bins. Finally, Eqs.(16) and (17) declare the types of the decision variables.

5 Bin decomposition

The tentative assignment MIP model provides a useful starting point for packing individual bins but is limited by its consideration of only the area of items and bins, which can result in generated assignments that are not feasible. In order to address this limitation, we focus on finding a feasible arrangement for a subset of rectangular items that maximizes the packed item's total area while minimizing waste, which is known as the 2D Knapsack Problem (2DKP). Our state-of-the-art constraint programming (CP) approach provides a precise solution for this problem by relying on propagation, inference, and domain reduction mechanisms intrinsic to constraint programming, along with cumulative scheduling relaxations and symmetry breaking constraints. While our approach has demonstrated high competitiveness on benchmark instances with up to 30 items, the use of CP as part of a heuristic solution may still result in slow processing. To overcome this challenge, we have developed a divide and conquer branching scheme that cuts the bin into several base case regions. Each base case region is assigned a subset of items that can be efficiently solved by the CP model. We acknowledge that tuning the base case figure is a topic of ongoing research, as balancing the work between the heuristic and the CP model affects both the quality and performance of the solution. Nonetheless, the advantage of our decomposition heuristic is that it effectively breaks down a complex cutting problem into several smaller, more manageable problems that can be efficiently solved using our CP approach.

There are many questions to consider when designing a decomposition heuristic. How might one choose which cutting positions to try? Furthermore, what positions should be considered valid/effective? The bin is being split into two regions, then those regions are potentially being split further, how might one choose which region to target for cutting? How are the items to be assigned to regions? Finally, how should one branch and descend in the search tree to get good bounds faster?

5.1 Cut Positions

Consider a region r of size (W_r, H_r) there are W_r, H_r possible cuts that can be made on r and all its sub-regions, of course that would leave W_r H_r $1{\times}1$ regions which would have limited utility except in a trivial edge case. In reality only one cut will be made on any region, then those sub regions can be further cut, so given a region r there would be $W_r + H_r$ possible positions to try, although not all those cuts will be worth considering.

For a cut to be useful each of the new regions must be able to fit some of the items. Since each of the items have their own spatial dimensions, it makes sense to choose cutting positions relative to those dimensions, such a method could guarantee that at least one item fits in each new region. To eliminate symmetrical cuts, each cut position to try should be less than half or equal to the respective dimension (width/height), this effectively halves the search positions without losing generality of solutions. To fulfil these requirements, one might use a Forward Dynamic Programming Algorithm to determine valid potential cut positions.

5.2 Assignment MIP

Through an assignment MIP we can allocate items into several regions to generate upper and lower bounds for each branching node to ensure faster solution converge. Our MIP makes use of Dual Feasible Functions as they are a proven effective way to generate faster lower bounds and valid inequalities for MIPs with knapsack constraints [3].

5.3 Decomposition Heuristic

Figure 2 illustrates the decomposition heuristic descending a tree to find solutions. The process which governs the actions taken at any given node is demonstrated in Figure 3. At some node we are given a set of regions and a set of items, if there is a target already defined, then simply proceed to the next cut, provided the next cut exists. The target is only pre-defined in the event of backtracking. The MIP tells us whether we can rely on our CP model to solve the problem, or that more cutting is required to determine a solution. To select a target simply choose the largest region. Cuts are obtained by the Forward Dynamic Programming Algorithm.

Within Figure 2 it is shown that the items get shuffled around as the tree is descended. Figure 2 only shows 2 possible cuts; however, the number of cuts is usually proportional to the number of items in the region. Upper bounds are produced by the assignments. As a solution is discovered a new lower bound is discovered, we enforce in our MIP that any assignment is better than these bounds to prune the search space.

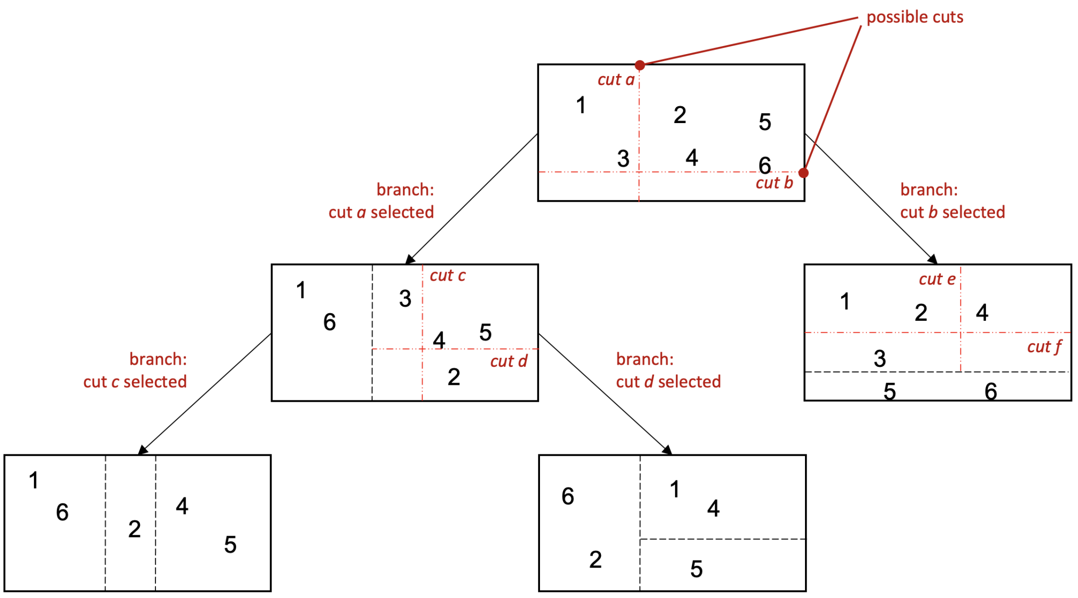

Figure 2: Bin decomposition with limited branching

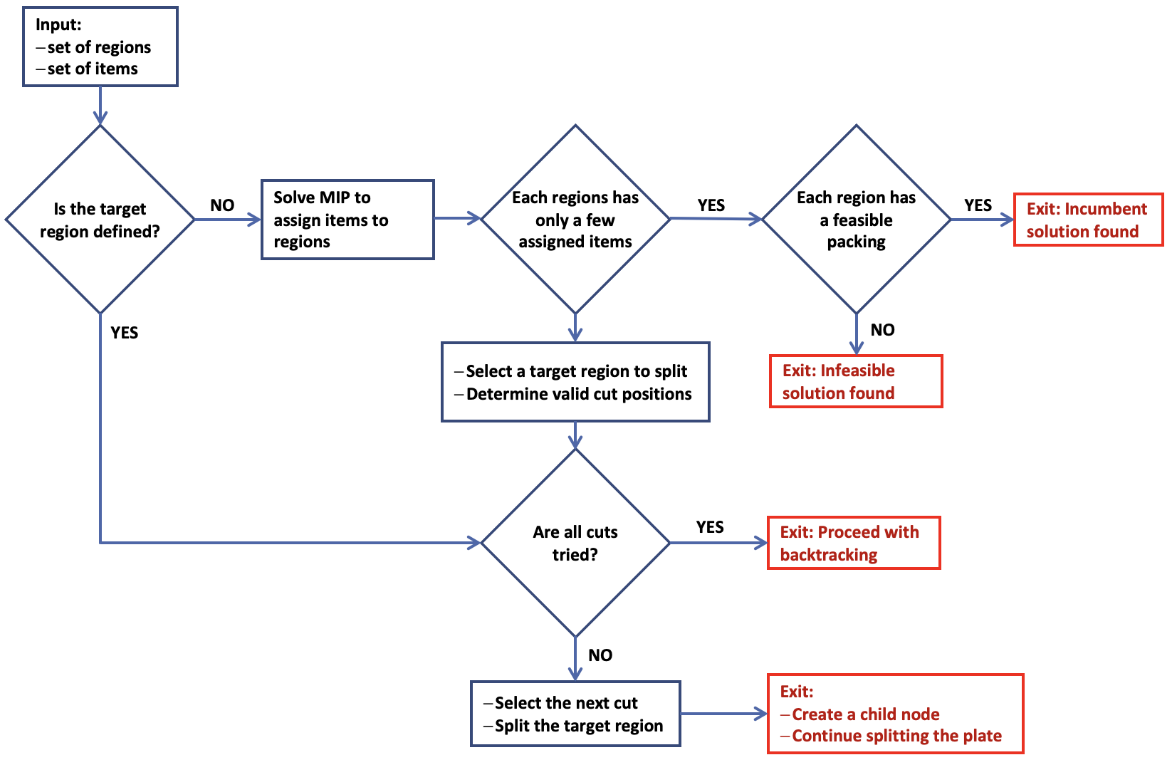
Input:
–set of regions
–set of items
Is the target region defined?
NO
Solve MIP to assign items to regions
Each regions has only a few assigned items
YES
Each region has a feasible packing
YES
Exit: Incumbent solution found
NO
Exit: Infeasible solution found
YES
–Select a target region to split
–Determine valid cut positions
Are all cuts tried?
YES
Exit: Proceed with backtracking
NO
–Select the next cut
–Split the target region
Exit:
–Create a child node
–Continue splitting the plate

Figure 3: Node Selection for Bin Decomposition

6 Discussion and Conclusion

The computed dual bounds have revealed that the state-of-the-art primal bounds are not optimal, indicating potential for further improvement. Our preliminary computational experiments on the test benchmark instances have shown that the approximate solutions generated by our approach outperform those reported in the literature, and often achieve optimality.

However, the scalability of our approach remains a challenge, particularly for larger problem instances. We need to strike a balance between the time spent on MIP subroutines and the time allocated for heuristic search.

In order to enhance the performance of our matheuristic, we aim to develop a learning mechanism that can capture the dependency between decision variables and preserve the packing of items into a single bin whenever such items exhibit synergy. Although an exact solution to this problem is unattainable, we plan to extend our ongoing work by incorporating a branch-and-price algorithm, which can be initialized using the solutions obtained from our heuristic approach.

In this paper, we proposed an approximate decomposition-based approach to efficiently solve the variable-sized 2-dimensional bin packing problem (VS-2D-BPP), which arises in various industries where materials need to be cut into smaller pieces of different sizes. We adopted the Dantzig-Wolfe decomposition, a linear programming-based formulation that decomposes the problem into smaller subproblems, and developed a matheuristic approach that applies Mixed Integer Programming to find patterns heuristically. Our method searches for a set of patterns that can efficiently cover the items in the problem instance and reduces the number of packings that need to be considered in the Dantzig-Wolfe model. Our approach showed promising results in terms of reducing total cost and increasing time efficiency, demonstrating its potential to be applied in real-world industrial settings. Further research can be done to improve the accuracy and scalability of the proposed method and to investigate its applicability in other related optimization problems.

References

[1] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, pp. 101–111, 1960.

[2] D. Pisinger and M. Sigurd, "Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem," *INFORMS Journal on Computing*, vol. 19, pp. 36–51, Feb. 2007. DOI: `10.1287/ijoc.1060.0181`.

[3] J. Rietz, C. Alves, J. Valério de Carvalho, and F. Clautiaux, "Constructing general dual-feasible functions," *Operations Research Letters*, vol. 43, no. 4, pp. 427–431, 2015, ISSN: 0167-6377. DOI: `https://doi.org/10.1016/j.orl.2015.06.002`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0167637715000759`.