Branch-and-Cut Based Method for the Two-dimensional Variable-Sized Guillotine Bin Packing Problem

Submitted as Honours Dissertation in SIT724

SUBMISSION DATE

T2-2021

Vireakpheakkdey Tiv

STUDENT ID 218012533

COURSE - Bachelor of Information Technology Honours (S470)

Supervised by: Dr. Sergey Polyakovskiy

Abstract

In this paper we study the use of Normal Pattern define by Christofides and Whitlock (1977)[4], to deal with the issue of the highly nested region in a matheuristic approach to solving the two-dimension variable guillotine bin packing problem (2DVSGBPP), by Dr Polyakovskiy and M'Hallah [15]. The 2DVSGBPP is a variant of the classical Bin Packing problem, where rectangular bins of different costs, opposed to identical bins, are available to pack a set of small rectangular items. And the guillotine constraint also dictates that each item must be obtained through edge-to-edge cuts. The objective is to pack all the items while minimizing the total cost of used bins. While the matheuristic proposed yields promising results, due to its nature of producing two additional regions for every item packed, large bins struggle to pack efficiently due to irregular packing patterns. On the other hand, small bin tends to produce tight packing as there is less opportunity to generate highly nested regions. The Normal Pattern principle would tackle this issue by informing the matheuristic of all the viable cuts, which allows for cutting early in the packing process to produce smaller packing regions from one bin. In addition, this paper also breaks the 2DVSGBPP into a master and slave problem, where the master is a relaxed variant of the original problem. It aims to solve the assignment problem of 2DVSGBPP without a geometric constraint to obtain a lower bound. Furthermore, the master problem also reduces 2DVSGBPP into a Two-Dimensional Guillotine Bin Packing Problem (2DGBPP). As for the slave problem, it aims to solve 2DGBPP with a geometric constraint as well as incorporate the Normal Pattern.

Contents

1 Introduction

The Two-dimensional Variable-Sized Guillotine Bin Packing Problem (2DVSGBPP) is a variant to the classical Two-dimensional Bin Packing problem (2DBPP), where rectangular bins of different costs, opposed to identical bins, are available to pack a set of small rectangular items. And the guillotine constraint also dictates that each item must be obtained through edge-to-edge cuts. The objective is to pack all the items while minimizing the total cost of used bins.

The 2DVSGBPP belongs to the area of the Cutting and Packing (C&P) problem, which is characterized by a common logical structure[7]. The structure usually entails two groups of elements of geometric bodies which are the "large objects" and "small items". And the cutting or packing aspect of the problem is the process of generating a pattern, which is a geometric combination of how small items are being assigned to large objects. Works on the C&P problem can be traced back to the early 70s such as the "Optimum packing and depletion", and "The Knapsack problem: A survey"[3, 9].

While both cutting and packing is the realization of patterns, these two processes are applicable in term of real-world application. As for 2DBPP, a great deal of work has been published on the subject as this problem can model various real-world applications. The application of the Two-dimensional Variable-Sized Bin Packing Problem can be widely applied in the industrial sector where the cutting application can assist in the process of cutting glass, steel, wood, plastic, and paper manufacturing[15]. It is fitting for the cutting application in glass due to the practical constraint of guillotine cutting. As the manufacturing process of glass utilizes this cutting method to the fragile nature of the material. Still, there are various benefits such as efficiency in operation and cost benefits. Firms can achieve maximum production of the material (outputs) by cutting in patterns that minimise waste and using the minimum number of sheets required (input)[5].

Another form of industrial application of the bin packing problem is for the supply chain and logistic industry, however using three-dimensional packing problems. In the packing process, it is applicable to find the optimal number of containers to pack goods of different sizes at different costs[1]. Consider a distribution company that packs products into boxes of different sizes, then putting them together into larger boxes (bins) for delivery. Since there are several types of bins, the benefits are to achieve the

most efficient way of loading such as determining the most appropriate bins (which size at what cost) to minimise the transportation costs.

2 Literature Review

The cutting and packing problem have been studied since the 1950s and since then researches on this topic have been growing. The momentum first started in operations research on one-dimensional and two-dimensional bin packing problems. Later, much research started assuming all the bins to be identical in size and focused on one dimension while others focused on two dimensions[11]. Most literature reviews on the Two-dimensional Variable-Sized Bin Packing Problem (2DVSBPP) have been conducted to study the non-guillotine variant of the problem, therefore, leaving research on the guillotine variant quite scarce.

For the non-guillotine problem, a study by Pisinger and Sigurd (2005)[14] developed an exact algorithm based on the branch-and-price algorithm. They decompose the two-dimensional problem into one-dimensional knapsack sub-problems. A valid constraint is added every time the one-dimensional problem is infeasible for the two-dimensional problem and repeated until a feasible solution is achieved. Overall, they utilized various methods in solving this problem from Dantzig-Wolfe decomposition and column generation to solve the decomposed problem. However, a drawback from this study is that some solutions are not solved optimally, it is demanding in memory space and long computational time (more than one hour with only 100 items).

Another research by Lui et al. (2011)[11] seeks to develop a heuristic algorithm for a general variable-sized problem to generate a more satisfying result in an acceptable amount of time. They also proposed a specific method that factors in different consumer demands such as rotation or orientation of items and cutting styles. Overall, they produced two heuristic algorithms (H1 and H2) intending to minimise the total space of the bin used. Both heuristics were able to achieve good quality solutions in a few seconds compared to existing algorithms at that time. Furthermore, the H2 algorithm can easily consider customer demands while both algorithms can also be employed to solve other combinatorial problems like set partitioning problems.

Wei et al. (2013)[16] also focused on examining the two-dimensional variable-sized bin packing problem (2DVSBPP) using a goal-driven approach. They partition the search space of the 2DVSBPP into sets by objective value then ensure all subset has

the same cost. Next, they impose an order to find a feasible solution. Overall, they are conducting a binary search on the objective value of the solution subsets. The solution obtained by their GDA outperformed the eight best-performing algorithms from Ortmann et al (2010)[13] with 120 seconds of computational time.

Meanwhile, Alvarez-Valdes et al. (2013)[1] present a metaheuristic algorithm for the three-dimensional and two-dimensional variable size bin packing problem. They incorporated path relinking strategies to combine the best solutions obtained in the iterative process. Overall, the hybrid of GRAPS/Path Relinking allowed the algorithm to obtain a more robust solution. When comparing their GRASP + PR algorithms to the eight best-performing algorithms from Ortmann et al (2010)[13], their algorithm obtains slightly better results than the maximum of the eight algorithms. This research was one of the first to form such a combination by using path relinking procedures and demonstrate success in its application to solve the bin packing problem.

As for the guillotine variant, Ortmann et al. (2010)[13] proposed a two constructive heuristic called stack ceiling (SC) and stack ceiling with re-sorting (SCR). Many restrictions were enforced such as items must be packed orthogonally into the bin, items may not overlap, no rotation may occur during packing and must be parallel to the edges of the bin. The first algorithm is aimed to solve the strip packing problem that aims to pack all items while minimising the resulting packing height. Then the second algorithm attempts to repacks the assignment into smaller bins to minimize wasted space. Overall, their proposed algorithm improved the solution quality, time and packing density of the 2DVSBPP. Furthermore, their algorithms have advantages in packing density of instances where items are the size varies by a great margin. Their result is later improved upon by Hong, Zhang, Lau, Zeng, and Si [10], as they utilize a mixed packing algorithm in combination with the Best Fit algorithm and as well as a backtracking algorithm.

Meanwhile, Polyakovskiy and M'hallah (2021)[15] employed three strategies to the packing procedures with a similar objective as other pieces of literature which are to minimise the waste of used bins. The first strategy is to reserve free space for unpacked items by using a sequence of low-level MIP. The second strategy employs a MIP model with a feasibility constraint, essentially providing a feasibility check on the reserved space from the first strategy. In addition, it rules out or eliminates infeasible solutions in future searches. Lastly, they further restrict searching for solutions that exceed the upper bound on the objective function which is set through known feasible solutions in previous cycles. Overall, their algorithm outperforms the state-of-the-art methods in

identical bin instances. While for variable size bin packing it matched 160 and improve 294 best-known results out of 500.

This paper aims to improve on the work of Dr Polyakovskiy and M'Hallah (2021) [15] in their paper "A Lookahead Matheuristic for the Unweighted Variable-Sized Two-dimensional Bin Packing Problem", which will be called Matheuristic for short. Where the known is an issue with their proposed approach is producing highly nested regions which make large bins struggling to pack efficiently. On the other hand, small bin tends to produce tighter packing as there is less opportunity to generate highly nested regions. In this paper, we will explore the use of the Normal Pattern define by Christofides and Whitlock (1977) [4], to deal with the issue of the highly nested region. The Normal Pattern principle would tackle this issue by informing the matheuristic of all the viable cuts, which allows for cutting early in the packing process to produce smaller packing regions from one bin. In addition, this paper also breaks the 2DVSGBPP into a master and slave problem, where the master is a relaxed variant of the original problem. It aims to solve the assignment problem of 2DVSGBPP without a geometric constraint to obtain a lower bound. Furthermore, the master problem also reduces 2DVSGBPP into a Two-Dimensional Guillotine Bin Packing Problem (2DGBPP). As for the slave problem, it aims to solve 2DGBPP with a geometric constraint as well as incorporate the Normal Pattern principle.

3 Problem Description

In the 2DVSGBPP, we are given a set of $I = \{1, 2, \cdot, n\}$ of rectangle items of width w_i , length l_i and area $a_i = l_i w_i$, to pack into non-identical bins or bins with different costs. There is a set $T = \{1, 2, \ldots, m\}$ of rectangular bin types where each bin type $t \in T$ has a length L_t, width W_t, and $A_t = L_t W_t$. The variable-sized aspect of this problem dictates that there are at least n bins for each bin type t. The problem also assumes that there's at least one bin type that each item $i \in I$ can fit into, which mean there is at least a bin set $B = \{1, 2, \ldots, n\}$ for any bin type $t \in T$, where $b \in B$. And for any bin type $t \in T$, it can at least fit on item. The aim is to pack all the items into the least amount of bins via guillotine cuts, in other words, minimize the total cost of used bins. And the total cost of used bins, as suggested is the total area of bins with items packed into them.

3.1 All possible cuts with Normal Pattern

The Branch-and-Cut based approached to this problem discussed in section 4, takes advantage of a cutting pattern utilized in most literature in C&P, where items are packed into the left-most bottom of the bin first, and it is called Normal Pattern. Normal Pattern is a set of all possible cuts in a bin accordingly to the items needed to be packed, it can be thought of as the various combinations of items being place into the bin along an axis side by side. Christofides and Whitlock (1977)[4] formally defined a set of normal patterns as:

$$N_0 = \{x = \sum_{i \in I} \omega_i \epsilon_i : 0 \leq x \leq W, \epsilon_i \in \{0,1\} for i \in I\} \tag{1}$$

This definition does not consider for models where items can only be packed in the lowest left corner, so for the case where $W = 20$ and $w = \{0, 4, 10, 13, 18\}$, N_0 would be $\{4, 10, 13, 14, 17, 18\}$. It is more appropriate to apply a reduction proposed by Beasley (1985)[2] where:

$$N = \{x \in N_0 : x \leq W - w_{min}\} \tag{2}$$

Conveniently when applied to the same instance above $N = \{0, 10, 13, 14\}$, but there's one more small reduction that is further applied. For our case, the "0" cutting position is irrelevant since no cut can be made on the edge of the bin, therefore:

$$N = \{x \in N_0 : 0 < x \leq W - w_{min}\} \tag{3}$$

From this normal pattern definition, the pseudo-code below is the implementation of it from Côté & Iori (2018)[6] with the reduction methods mentioned above incorporated.

The normal pattern algorithm requires a set of items, and either the bin's width or height depending on what cut orientation is being calculated. An array T from 0 to W is initialized to store the position of all viable cuts. Set $T[0]$ to 1 as it is the first viable cut on all patterns. Then set Threshold to $W - w_{min}$ accordingly to reduction by Beasley (1985)[2]. For each item, iterate from $W - w_i$ back to 0 with p index, wherever $T[p] = 1$ and less or equal to Threshold then set $T[p + w_i]$ to 1, essentially placing the item right next to the cut found, which result in a new viable cut. After

Algorithm 1 Normal Pattern

Require: I: a set of items, W: bin's width/length
  $T \leftarrow [0\ to\ W]$: an array with all the entries initialized to 0.
  $T[0] \leftarrow 1$
  $Threshold \leftarrow W - w_{min}$
  for $i \in I$ do
    for $p = W - w_i$ to 0 do
      if $T[p] = 1$ & $T[p + w_i] \leq Threshold$ then
        $T[p + w_i] \leftarrow 1$
    $T[0] \leftarrow 0$
    $N_0 \leftarrow \emptyset$
    for $p = W$ to 0 do
      if $T[p] = 1$ then
        $N_0 \leftarrow \cup\{p\}$

finding all the cut set $T[0]$ to 0 as according to our reduction method. The final step is storing and return viable cut positions. This normal pattern algorithm is heavily relied upon in section 4.2 Slave algorithm, as it is used to calculate all the possible cuts and determining which cut produce the best result.

3.2 Dual Feasible Functions Methods

The Dual Feasible Functions (DFF) introduced by Fekete and Schepers (2004)[8] in their work, "A General Framework for Bounds for Higher-Dimensional Orthogonal Packing problems" is used in the Minsum MIP in the next section as part of the Slave algorithm in section 4.2. Due to its ability to approximate the packing ratio of a given item to a bin without geometric constraints and as well the result presented in the Matheuristic[15] because of its inclusion, it is used here to relax the hard geometric constraints of the Branch-and-Cut approach in section 4.

Fekete and Schepers utilized three DFF which ultimately help obtain good classes of lower bound of bin packing problems[8]. The first class $u^{(k)}$ find a value that can fit as many items as possible in what they called the "win-zone", the sub-interval of [0,1]:

let $k \in \mathbb{N}$. Then $u^{(k)} : [0, 1] \leftarrow [0, 1]$

$$x \leftarrow \begin{cases} x, \ for\ x(k+1) \in \mathbb{Z} \\ \lfloor (k+1)x \rfloor \frac{1}{k}, \ else \end{cases}$$

As for the second class, the DFF is the basis found in the Matello and Toth (1990)[12] bin packing bound L2, which neglects all items smaller than a given ϵ value. But any savings done by omitting the size of the smaller items are accounted for by increasing the size of items larger than $1 - \epsilon$.

let $\epsilon \in [0, \frac{1}{2}]$. Then $U^{(\epsilon)} : [0, 1] \leftarrow [0, 1]$

$$x \leftarrow \begin{cases} 1, for\ x > 1 - \epsilon \\ x, for\ \epsilon \leq x \leq 1 - \epsilon \\ 0, for\ x < \epsilon \end{cases}$$

Lastly, the third class of DFF, similar to the previous class, ignores any item smaller than the threshold value ϵ. But at unlike the second class, from the interval $\left(\epsilon, \frac{1}{2}\right]$, the functions are constant. While from the $\left(\frac{1}{2}, 1\right]$, they assume the form of a step function.

let $\epsilon \in [0, \frac{1}{2}]$. Then $\phi^{(\epsilon)} : [0, 1] \leftarrow [0, 1]$

$$x \leftarrow \begin{cases} 1 - \frac{\lfloor (1-x)\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor}, for\ x > \frac{1}{2} \\ \frac{1}{\lfloor \epsilon^{-1} \rfloor}, for\ \epsilon \leq x \leq \frac{1}{2} \\ 0, for\ x < \epsilon \end{cases}$$

Finally, these three dual feasible functions are used to construct seven different functions which essentially transformed the packing ratio of an item i given a region k, where $d_1 = \frac{w_i}{W_k}$ and $d_2 = \frac{l_i}{L_k}$ [8].

For $p, q \in (0, \frac{1}{2}]$ let

$$\begin{aligned} w^{(1)(p)} &:= (u^{(1)} \leftarrow (w_1), U^{(p)} \leftarrow (w_2)) \\ w^{(2)(p)} &:= (U^{(P)} \leftarrow (w_1), u^{(1)} \leftarrow (w_2)) \\ w^{(3)(p)} &:= (u^{(1)} \leftarrow (w_1), \phi^{(p)} \leftarrow (w_2)) \\ w^{(4)(p)} &:= (\phi^{(p)} \leftarrow (w_1), u^{(1)} \leftarrow (w_2)) \\ w^{(5)(p)} &:= (\qquad\quad (w_1), U^{(p)} \leftarrow (w_2)) \\ w^{(6)(p)} &:= (U^{(P)} \leftarrow (w_1), \qquad\quad (w_2)) \\ w^{(7)(p,q)} &:= (\phi^{(p)} \leftarrow (w_1), \phi^{(q)} \leftarrow (w_2)) \end{aligned}$$

3.3 Minsum MIP

The Minsum MIP is a packing procedure demonstrated in Matheuristic by Dr Polyakovskiy and M'Hallah (2021)[15], which utilized a minimax model to maximize the packing density by minimising the cost. This model also includes the DFF method, to approximate the packing ratio of an item into a region, which is useful for estimating whether a subset of all the items can fit into the given region. In our case, the MIP is slightly tweaked to include an upper bound on C, to serve the purpose of feasibility checking the cuts calculated in section 4.2.

Let be $x_{ik} \in \{0,1\}$ the decision variables, if $x_{ik} = 1$ then item i assigned to region $k \in K$, of width W_k,length L_k and $A_k = W_k L_k$. Where regions are smaller sections in a given bin. While λ_{ik}^f is the $f^{th} \in F$ constraint generated from DFF methods of item i packed into region k, which is relevant for line 8.

MinsumMIP:

$$minimise\ C \tag{4}$$

$$\sum_{k \in k} x_{ik} = 1 \qquad , i \in I' \tag{5}$$

$$\sum_{i \in I'_k} x_{ik} \geq 1 \qquad , k \in K \tag{6}$$

$$\sum_{i \in I'_k} \frac{a_i}{A_k} x_{ik} \leq C \qquad , k \in K \tag{7}$$

$$\sum_{i \in I'_k} \lambda_{ik}^f x_{ik} \leq C \qquad , k \in K f \in F_k \tag{8}$$

$$C \leq 1 \qquad , C \in \mathbb{R}_{\geq \nvDash} \tag{9}$$

$$x_{ik} \in \{0,1\} \qquad , i \in I, k \in K \tag{10}$$

The Minsum MIP is a minimax model, where the objective function (4) aims to minimize the maximum C. Constraints (5) and (6) ensure that each item is packed while imposing that each region must have at least one item packed into them. Constraints (7) and (8) both ensure that C is greater or equal to the sum of the cost of packing items into the region. For constraints (7) the cost is proportional to the packing ratio a_i/A_k , where item i is assigned to region k. As for constraints (8), the cost is proportional to the transformed packing ratio of a_i/A_k, which is λ_{ik}^f and is calculated using DFF. As for constraint (9), this imposes an upper bound on C and

as well as defined its domain. And finally constraint (10) define the domain for the variable x_{ik}.

4 Branch-and-Cut Algorithm

The method proposed in this paper takes a decomposition-based approach, where the problem is split into a master-slave problem, where the master algorithm, from section 4.1, aims to solve the assignment problem of 2DVSGBPP with relaxed geometric constraint. Instead of imposing hard geometric constraints restricting solution to only be feasible if the items packed inside can fit. The master algorithm treats items as dimensionless entities, because the geometric constraints are dropped. This allows the assignment of items into any bin as long as the sum of items' area is less than the bin's area. And in this decomposition-based approach the packing constraints in slave algorithm are considered to be complicated as it is solving the heart of the problem. Each time the master algorithm branches out and explore and an assignment of items into different bins, the slave algorithm handles the packing constraint through the process of cutting those bins into smaller region, hence the named Branch-and-Cut based algorithm. Since these assignments are essentially instances of 2DBPP, the slave algorithm's, from section 4.2, goal is to solve a problem with less complexity, in turn, provide a feasibility check for the original problem. And for every infeasibility assignment, it is added to the master algorithm as constraints to prevent branching down an infeasible branch.

Algorithm 2 Branch-and-Cut Algorithm

Require: N: a set of items, M: a set of bin types
  $S_{best} \leftarrow \emptyset$: a set of best solutions, and a solution contains multiple assighments.
  $S_{best} = MasterMIP \leftarrow (N, M)$ from section 4.1, which iteratively explore each solution S and evaluate whether it satisfied the $MasterMIP$
  if S satisfied the $MasterMIP$ then
    for each assignment $s \in S$, where $s = \{n \subset N, m \in M\}$ do
      if $FeasibilityCheck \leftarrow (s) = False$ then
        Add s as constraints to $MasterMIP$
  return S_{best}

4.1 Master Algorithm

As mentioned, the master algorithm's goal is to solve the assignment problem of 2DVSGBPP with relaxed geometric constraints the MIP below. *Let*x_{ibt} be matrix of binary decision variables, where if $x_{ibt} = 1$ then item i is assigned to bin b of bin type t. Let y be denoted as another matrix of binary decision variable such that $y_{bt} = 1$ if bin b of bin type t is opened for packing.

MasterMIP:

$$min \sum_{t \in T} \sum_{i \in I} A_t y_{bt} \tag{11}$$

$$\sum_{b \in B} \sum_{t \in T} x_{ibt} \ = \ 1 \qquad , \ i \in I \tag{12}$$

$$\sum_{i \in I} x_{ibt} a_i \leq A_t y_{bt} \qquad , b \in B, t \in T \tag{13}$$

$$x_{ibt} \in \{0, 1\} \qquad , i \in I, b \in B, t \in T \tag{14}$$

$$y_{bt} \in \{0, 1\} \qquad , b \in B, t \in T \tag{15}$$

The objective function (11) minimizes the total area of bins used. Constraints (12) impose that each item is packed only once, and every item must be packed. Constraints (13) ensures that there are enough open bins for the assigned items, in other words, the sum of the items area does not exceed the bin area. In addition, the constraint also ensures that only bins are used. Constraints (14) and (15) define the variable domains.

For an instance P with $I = \{5, 1, 8, 7, 2, 2, 5, 5, 1, 1\}$ and $T = \{10, 10, 5, 5\}$, where length is listed before width, the Master MIP would assign all the items into one bin with the dimension of the $width \ = \ 10$ and $length \ = \ 10$. As the total of sum of all the items' area is only 91% of the opened bin's area, which regardless of whether it fit or not the master problem will assign it this way. As shown the instance P above is reduced to 2DBPP instance p with $I \ = \ \{5, 1, \ 8, 7, \ 2, 2, \ 5, 5, \ 1, 1\}$ and $b = \{10, 10\}$.

4.2 Slave Algorithm

This algorithm's objective is to provide a feasibility check for the assignments outputted from the master algorithm. The slave algorithm is an iterative approach that

is built from multiple core concepts such as Normal Pattern and Minsum MIP from section 3.1 and 3.3 respectively. As shown in the Slave algorithm pseudo-code below.

The algorithm requires the assignments from the master algorithm which consist of a set of items and the bin these items needed to be packed in. A queue Q_0 is created for storing regions that can still be cut, and its first region is the region made from the given bin as no cut has been made yet. And array Q_1 is initialized for storing any regions that cannot be cut anymore. In addition, the packing objective upper bound U_0 is set to 1 and this algorithm would try to minimize that. A flag $feasibilityCheck$ is set to $False$ and would only be $True$ if the algorithm finds the given assignment feasible for packing.

From the while loop, the iterative approach begins, and won't end until there is no more region in Q_0. One by one each element from Q_0 is selected and goes through a series of procedures to determine which cut would yield the best result. It starts by creating r'_{best} to store the best cut/regions. Then grouping all regions needed to be packed together into an array Q which consist of elements from Q_0 and Q_1. Then viable vertical and horizontal cuts are calculated from the Normal Pattern discussed in section 3.1. For each of the calculated cut, two regions $\{r'_0, r'_1\}$ are made from the original region r' used. These new regions are then combining with Q in the Minsum MIP calls which would produce an objective value L_0. Then L_0 is compare against U_0, if $L_0 < U_0$ then U_0 is set to L_0 and $\{r'_0, r'_1\}$ are saved to r'_{best}. This repeats until no cut is left, at the end if there is no feasible cut for r' then it moves to Q_1. Else $feasibilityCheck$ is set to $True$ and r'_{best}, which has been sorted by area in descending order, is added to Q_0. What this means is that the algorithm would consider an assignment feasibly as long as there is at least one feasible cut for the bin. Finally, the array Q is set to an empty set. This entire process is repeated until no more region in Q_0, and only then the algorithm will return its result as $feasibilityCheck$.

As an example, take the 2DBPP p from section 4.1 and feed it into the algorithm above it would conclude the assignment as feasible as it has at least one feasible cut. The bin is cut up into 3 regions $\{(10, 9),\ (2, 1),\ (8, 1)\}$, where the first region would have three items assigned into it $\{(8, 7),\ (2, 2),\ (5, 5)\}$, second region is assign the item $(2, 1)$, and $(8, 1)$ is assigned to the last region.

Algorithm 3 Feasibility Check Algorithm

Require: N: a set of items, m: a single bin to pack all the items in.
$Q_0 \leftarrow \emptyset$: a queue that stores all the regions that can still but cut.
$Q_1 \leftarrow \emptyset$: an array that stores all regions that cannot be cut.
$r \leftarrow$ a region from bin m (the region is as big as the bin since no cut has been made).
$Q_0 \leftarrow r$
$U_0 \leftarrow 1$
$feasibilityCheck \leftarrow$ false
while $Q_0 \neq \emptyset$ do
  $r'_{best} \leftarrow \emptyset$
  $r' \leftarrow Q_0$: dequeue one region from Q_0
  $Q \leftarrow Q_0 \cup Q_1$:an array of regions available to be packed in.
  $NormalPattern \leftarrow (N, r'_{width})$from section 3.1 to determine viable vertical cuts.
  $NormalPattern \leftarrow (N, r'_{length})$from section 3.1 to determine viable vertical cuts.
  for each viable verticle cuts do
    $\{r'_0, r'_1\} \leftarrow r'$: two regions are made from cutting region r'
    $L_0 = Minsum \leftarrow (N, Q \cup \{r'_0, r'_1\})$: Call $Minsum$ from section 3.3 to determine the cut's objective value.
    if $L_0 < U_0$ then
      $U_0 \leftarrow L_0$
      $r'_{best} \leftarrow \{r'_0, r'_1\}$
    else
      continue
  for each viable horizontal cuts do
    $\{r'_0, r'_1\} \leftarrow r'$: two regions are made from cutting region r'
    $L_0 = Minsum \leftarrow (N, Q \cup \{r'_0, r'_1\})$: Call $Minsum$ from section 3.3 to determine the cut's objective value.
    if $L_0 < U_0$ then
      $U_0 \leftarrow L_0$
      $r'_{best} \leftarrow \{r'_0, r'_1\}$
    else
      continue
  if $U_0 = 1$ then
    $Q_1 \leftarrow r'$
  else
    $feasibilityCheck \leftarrow$ true
    $Q_0 \leftarrow sortByDecending(r'_{best})$
  $Q \leftarrow \emptyset$
return $feasibilityCheck$

5 Conclusion

In conclusion, the proposed approach above aims to improve on the matheurtic method introduced by Dr Polyakovskiy and M'Hallah (2021) in solving the 2DVSGBPP[15]. Especially address the highly nested regions that exist in instances with large bin sizes relative to items' size. The Branch-and-Cut approach aims to tackles the aforementioned issue by employing a decomposition-based approach. As the method breaks the original problem down into two parts a master problem and a slave problem or subproblems, which help reduce the overall complexity of the problem. The master problem would handle the assignment problem of 2DVSGBPP, and the slave problem provides a feasibility check on the 2DBPP instances outputted from the master problem. This paper described in detail how one could model the 2DVSGBPP via a decomposition-based approach, while this approach is promising, extensive experimental work is required to obtain the experimental data and result. Especially focusing on test instances such as the ten-classes set used by Pisinger & Sigurd (2005) and Polyakovskiy & M'Hallah (2021), which provide a good benchmark as to how The Branch-and-Cut approach performed. [14, 15].

References

[1] R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit, A GRASP/Path Relinking algorithm for two- and three-dimensional multiple bin-size bin packing problems, Computers and Operations Research, 40 (2013), pp. 3081–3090.

[2] J. . E. . Beasley, Algorithms for Unconstrained Two-Dimensional Guillotine Cutting, Palgrave Macmillan Journals, 36 (1985), pp. 297–306.

[3] A. R. Brown, Optimum packing and depletion, American Elsevier, (1971), p. 106.

[4] N. Christofides and C. Whitlock, An Algorithm for Two-Dimensional Cutting Problems, Operations Research, 25 (1977), pp. 30–44.

[5] J.-F. Côté, M. Haouari, and M. Iori, Combinatorial Benders Decomposition for the Two-Dimensional Bin Packing Problem, INFORMS Journal on Computing, (2021).

[6] J.-F. Côté and M. Iori, The meet-in-the-middle principle for cutting and packing problems, INFORMS Journal on Computing, 30 (2018).

[7] H. Dyckhoff, A typology of cutting and packing problems, European Journal of Operational Research, 44 (1990).

[8] S. P. Fekete and J. Schepers, A general framework for bounds for higher-dimensional orthogonal packing problems, Mathematical Methods of Operations Research, 60 (2004), pp. 311–329.

[9] S. Harvey M and D. K. Cornelis A, The knapsack problem: a survey, Naval Research Logistics Quarterly, 22 (1975), pp. 127–144.

[10] S. Hong, D. Zhang, H. C. Lau, X. Zeng, and Y. W. Si, A hybrid heuristic algorithm for the 2D variable-sized bin packing problem, European Journal of Operational Research, 238 (2014), pp. 95–103.

[11] Y. Liu, C. Chu, and K. Wang, A dynamic programming-based heuristic for the variable sized two-dimensional bin packing problem, International Journal of Production Research, 49 (2011), pp. 3815–3831.

[12] S. Martello and P. Toth, Lower bounds and reduction procedures for the bin packing problem, Discrete Applied Mathematics, 28 (1990), pp. 59–70.

[13] F. G. Ortmann, N. Ntene, and J. H. van Vuuren, New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems, European Journal of Operational Research, 203 (2010), pp. 306–315.

[14] D. Pisinger and M. Sigurd, The two-dimensional bin packing problem with variable bin sizes and costs, Discrete Optimization, 2 (2005), pp. 154–167.

[15] S. Polyakovskiy and R. M'Hallah, A Lookahead Matheuristic for the Unweighed Variable-Sized Two-dimensional Bin Packing Problem, European Journal of Operational Research, (2021).

[16] L. Wei, W. C. Oon, W. Zhu, and A. Lim, A goal-driven approach to the 2D bin packing and variable-sized bin packing problems, European Journal of Operational Research, 224 (2013), pp. 110–121.