An Improvement of a Heuristic Algorithm for 3D Bin-Packing Problem

Sarun Krisadee
Department of Computer Science, Faculty of Science
Chiang Mai University
Chiang Mai, Thailand, 50200
E-mail sarunkri9@gmail.com

Wattana Jindaluang[1]
Department of Computer Science, Faculty of Science
Chiang Mai University
Chiang Mai, Thailand, 50200
E-mail wjindaluang@gmail.com
[1]corresponding author

2024 28th International Computer Science and Engineering Conference (ICSEC) | 979-8-3503-6686-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSEC62781.2024.10770638

***Abstract*— We are given the large 3-dimensional rectangular items, which we refer to as the bins, in the 3D bin-packing problem. They have specific weight, length, height, and depth measurements. Additionally, we have various small, rectangular, 3-dimensional items that we call boxes. These boxes also have specific weights, lengths, heights, and depths. The goal of this problem is to pack all the small boxes into the bins so that their combined weight does not exceed the weight limit of each bin. Additionally, we aim to use as few bins as possible. In this paper, we refine the computational processes to enhance an algorithm by Youssef Harrath [1]. Experimental results indicate that as the number of boxes increases, our improvements can reduce the number of bins used, decrease running time, increase efficiency, and reduce memory usage. That is, the number of bins used, processing time, and memory use were all decreased to 38.68%, 58.42%, and 63.33%, respectively. In the meantime, bin space utilization efficiency scaled to 35.38%.**

Keywords — 3D bin-packing problem, spatial layer-based heuristic, Three-Stage Layer-Based Heuristic (TSLBH)

I. Introduction

According to [2], the 3-dimensional bin-packing problem, also known as the 3D bin-packing problem, is categorized as an NP-Hard problem. It arises from issues with storage management in a variety of constrained space situations, including box packing, shelf stocking, and packaging inside freight containers. The primary objective of this problem is to compute efficient methods to store small, closed 3D rectangular objects (called 'boxes') inside a larger 3D rectangular space (called 'bins'). Fig.1 illustrates an instance of the problem.

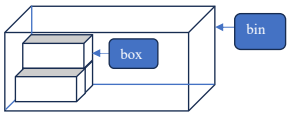

Fig.1. Characteristics of box packing in the 3D bin-packing problem

There are several possible goals for the 3D bin-packing problem, including reducing the overall packing time, utilizing fewer bins, and reducing the total amount of empty space inside the bin. For example, achieving these goals can help cut down on expenses and resources associated with the usage of different packing materials or space for storing items during the shipping and storage of goods. As demonstrated by the research of [3], who developed and adapted an algorithm to the scenario of packing goods in a container in the online case—where information about items is unknown in advance and they must be packed in real-time upon receipt—this results in maximizing the cost-effectiveness of expenses related to renting space or transportation. Using a set of created data, this paper also improved the online packing heuristic's efficacy. Another paper by [4] investigates how the 3D bin-packing problem is applied while packing luggage into ULDs, which are shaped and sized differently than conventional airplane containers. The analysis takes into consideration the maximum weight that each container can hold in addition to the factors of safe placement and uniform weight distribution within the container.

In this paper, we aim to improve the heuristic approach for Y. Harrath's 3D bin-packing problem in order to enhance overall bin usage efficiency and reduce the number of bins and time consumption.

The paper is organized as follows: Section 2 contains a review of the literature. Section 3 reviews the Y. Harrath's algorithm and presents our improvement strategy. Section 4 describes the experimental setup and results. Finally, Section 5 presents the conclusion.

II. Literature Review

There are numerous approaches to solving the 3D bin-packing problem. One of the basic concepts for solving this problem is the Layer-Based Heuristic approach, as described by [5]. This method uses heuristics to find solutions in each layer within the bin, as illustrated in Fig.2. The approach involves selecting the appropriate type and number of boxes for each layer until no more boxes can fit, indicating the completion of packing for that bin.

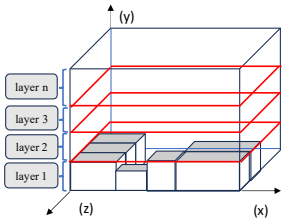

Fig.2. Solving problems using a heuristic approach by dividing the task into layers within the bin's internal space.

The online 3D bin packing problem is the focus of the research in [6]. Since each item in this scenario is unknown until bin packing is finished, bin packing needs to be done as soon as it comes. They offered a greedy multi-indicator fusion technique as a solution to this problem. This entails specifying several assessment indicators during the bin packing process, utilizing the SVR algorithm and quasi-Newton methods to determine how much of these indicators to fuse, and selecting the site that will produce the fused indications with the highest score. The researchers in [7] merged the greedy algorithm with the chemical reaction optimization technique. The chemical reaction optimization approach, which has been proposed recently, has a faster convergence time and all the benefits of the classic simulated annealing algorithm. In order to learn a policy that resembles

a heuristic and produces the orientations and sequence of objects to be packed simultaneously, the researchers in [8] devised a unique multi-task framework based on Selected Learning. A neural model learns a policy that resembles a heuristic, and quantitative experiments are created and carried out to show how effective this policy is. In order to combine a heuristic and greedy approach, [9] used a straightforward strategy based on integer linear programming (ILP) heuristics to solve the three-dimensional single bin-size bin packing problem without the need for any metaheuristic improvement. Initially, a complementary greedy solution strategy in two phases is suggested. Second, the improved version of the heuristic was enhanced with a simplified re-optimization process to enhance the quality of the results, especially for large-scale examples. Finally, the research conducted by Y. Harrath, to which this paper is referred, focuses on creating a heuristic algorithm for addressing the 3D bin-packing problem by means of a layer-based packing method. Since the approach we propose is based on Y. Harrath's method, the particulars of this algorithm will be described in the section on the proposed algorithm.

III. Proposed Algorithm

Since this paper enhances Y. Harrath's algorithm, we will start with a review. Y. Harrath presented a three-stage layer-based heuristic (TSLBH) strategy that solves the 3D bin-packing problem. This involves recording every configuration under which boxes could be arranged in each of the 6 orientational scenarios, as seen in Fig.3. The algorithm consists of three phases. The first stage is to determine the largest horizontal arrangement that is practical for each type of box and orientation. The second phase looks for workable vertical configurations that combine several box types without going beyond the bin's height. The third stage fills the bins based on weight balance while taking into account the patterns found in the first and second stages.

Several restrictions must be considered when managing packing boxes inside bins in accordance with the 3D bin-packing problem's principles. These include:

- Boxes have to be closed and packed inside the bins.
- Every type of box needs to have its surfaces lined up with the surfaces of the bin.
- The total weight of all boxes which contained in a bin must not exceed a weight of bin.

In order to describe our proposed algorithm, we will next go over the symbols in more detail as follows:

- The bin's dimensions (depth, height, length) are denoted by *D, H,* and *L*, respectively.
- The bin's maximum weight is denoted by *W*.
- The number of types of boxes is denoted by *t*.
- The box's dimensions (depth, height, length) of each type of box are denoted by $d_{1\ldots t}, h_{1\ldots t}, l_{1\ldots t}$, respectively.
- The weight of each type of box are denoted by $w_{1\ldots t}$.
- The total number of bins used for packing is denoted by m.
- The 6 possible orientations for placing a box are denoted by $p_{1\ldots 6}$.
- The number of each type of box are denoted by $n_{1\ldots t}$.
- The total number of all boxes is denoted by *n*.
- The number of boxes according to the placement pattern recorded for each type is denoted by $c_{1\ldots t}$.
- The total number of all boxes according to the recorded placement patterns is denoted by *cn*.
- The variable recording the placement pattern of each type of box and its orientation along the horizontal axis is denoted by *s1*.
- The variable recording the mixed-type placement pattern of boxes and their orientations is denoted by *s2*.

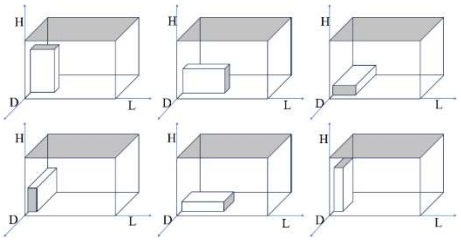

Fig.3. Characteristics of the 6 box orientations within a bin

Next, we will explain how we improved the TSLBH process. In the first stage, which involves finding the horizontal placement patterns, we adhered to the original method from a TSLBH algorithm. In the second and third stages, we enhanced the efficiency to achieve better results. The details are as follows:

Stage 1 (Finding the horizontal placement patterns):

As shown in Fig.4, this phase involves determining the maximum number of boxes that can be positioned horizontally. The boxes must be of the same type and orientation. The maximum number, volume, and orientation pattern are recorded in the variable *s1*. Algorithm 1 illustrates how to determine the horizontal box placement patterns.

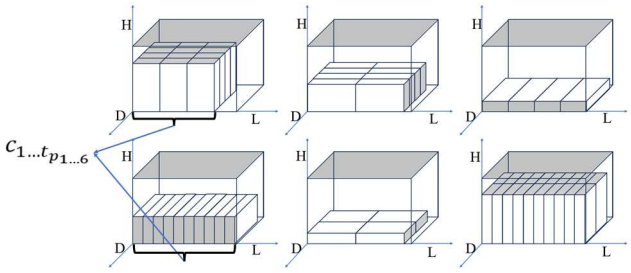

Fig.4. Horizontal placement patterns for each box type and orientation.

Algorithm 1: Stage1 Determine the horizontal box placement patterns.

Step 1
1: *for* $i \leftarrow 1$ *to* t:
2: *for* $p \leftarrow 1$ *to* 6:
3: $c_{i,p} = \left\lfloor \frac{D}{d_{i,p}} \right\rfloor \times \left\lfloor \frac{L}{l_{i,p}} \right\rfloor$
4: s1 $\leftarrow$ s1 $\cup$ $c_{i,p}$

Stage 2 (Finding the vertical placement patterns):

This entails figuring out how to arrange boxes vertically so that various types of boxes can be arranged under each of the six orientation scenarios. As seen in Fig.5, the overall height of all the positioned boxes cannot be greater than the height of the bin. For instance, the vertical placement patterns in this figure may indicate that we can pack type 1 boxes with a type 2 rotation in the first two layers, type 2 boxes with a type 1 rotation in the third layer, and type 3 boxes with a type 2 rotation in the final layer. Furthermore, because the

algorithm's first stage had already recorded aspects of each layer's packing, we know about the overall volume and number of boxes in this packaging layout. Let's say type 1 boxes with a type 2 rotation are packed with stage 1 of the algorithm, having a volume of 20 and an amount of 10. Type 2 boxes with a type 1 rotation have a volume of 25 and an amount of 20, and type 3 boxes with a type 2 rotation have a volume of 30 and an amount of 15. Then, in this packing layout, the total volume and number of boxes are 20 + 25 + 30 = 75 and 10 + 20 + 15 = 45, respectively. This stage is represented by algorithm 2.

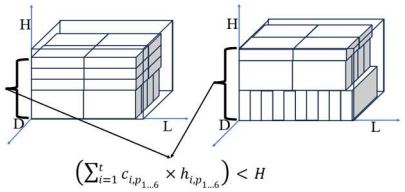

Fig.5. Vertical box placement patterns where the total height does not exceed the bin height.

Assuming the total height of the boxes does not exceed the height of the bin, Y. Harrath's research collects patterns at this stage for each instance, starting with box type 1, orientation 1, and quantity 1. If the weight exceeds the limit, the pattern will still be stored in an array. However, by retaining only patterns with maximum values, we reduce the number of patterns in the array, thereby shortening the time required to search for solutions in array.

Two factors are used to determine these maximum values. Patterns are retained in the array if either of the following conditions is met:

- The pattern before adding another box is regarded as having the highest box volume if adding more boxes causes the total height to exceed the bin's height.
- The pattern before adding more boxes is regarded as the maximum if adding them causes the total weight to exceed the bin's weight capacity.

Algorithm 2: Stage2 Determine the vertical box placement patterns and minimize the amount of data stored.

Step 2

1: $for\ p \leftarrow 1\ to\ 6$:
2: $for\ c_{1,p} \leftarrow 1\ to\ \left\lfloor \frac{H}{h_{1,p}} \right\rfloor$:
3: $for\ c_{2,p} \leftarrow 1\ to\ \left\lfloor \frac{H}{h_{2,p}} \right\rfloor$:
⋮
4: $for\ c_{i,p} \leftarrow 1\ to \left\lfloor \frac{H}{h_{i,p}} \right\rfloor$:
5: $maxHeight \leftarrow \sum_{i=1}^{t} c_{i,p} \times h_{i,p}$
6: $maxWeight \leftarrow \sum_{i=1}^{t} c_{i,p} \times c_{s1_{i,p}} \times w_i$
7: $if\ maxHeight < H\ and\ maxWeight < W$:
8: $verBoxes_i \leftarrow c_{s1_{i,p}}$
9: $hoBoxes_i \leftarrow c_{i,p}$
10: $maxBoxes_i \leftarrow hoBoxes_i \times verboxes_i$
11: $maxVolume \leftarrow \sum_{i=1}^{t} (d_{i,p} \times h_{i,p} \times l_{i,p} \times c_{i,p})$
12: $act \leftarrow 0$
13: $else$:
14: $if\ act == 0$:
15: $s2 \leftarrow s2 \cup (\cup_{i=1}^{t} p, hoBoxes_i, verBoxes_i, maxBoxes_i)$,
15: $maxVolume$
16: $act \leftarrow 1$

The goal of this step is to store the fewest possible patterns. By retaining just those patterns where the weight of all the boxes added together is less than or equal to the weight of the bin, we enhance the TSLBH algorithm. In the same manner, the boxes' overall height must be as high as feasible without going beyond the bin's height. Conversely, all of the earlier patterns involving lower total weight and height are eliminated. As shown in line 7 of Algorithm 2, which only chooses patterns that indicate the maximum number of boxes that cannot be added vertically, provides more information on this. The following is how the algorithm works:

If the height and weight of boxes do not exceed the bin's weight and height, the variable will be updated and stored temporarily (lines 8-11 of Algorithm 2), and the variable named act will be set to 0 (line 12 of Algorithm 2). When the total height of boxes exceeds the bin's height or the total weight exceeds the bin's capacity, the previously updated variable representing the maximum height or weight the bin can hold will be stored in the array *s2*. After storing the pattern in *s2*, the variable named act is updated to 1 (line 16 of Algorithm 2) to prevent the storage of patterns that exceed the bin's capacity, as this data is unusable.

Each time the box rotation or box type changes, the count resets to 1, ensuring that the patterns stored in *s2* are feasible for each box type and rotation configuration. This method guarantees that *s2* contains only relevant and usable patterns. The patterns stored in *s2* include the maximum vertical and horizontal box placements, the total number of boxes for each type, and the total volume of all boxes (line 15 of Algorithm 2).

Stage 3 (Finding the pattern for filling boxes into the bins):

At this stage, Y. Harrath's research process sorts the variables stored in array *s2* by pattern, from the highest to the lowest volume. Then, the patterns are packed with boxes starting from the highest volume until all boxes have been packed. The packing process for each pattern takes load balancing into consideration. In the proposed method, the patterns are sorted to find the one that yields the highest volume and that pattern is used until a condition arises that necessitates a change in the pattern. Afterward, a new highest volume pattern suitable for the remaining boxes is selected. The selection of the highest volume pattern will be based on the following criteria:

A. The issue of using the maximum volume pattern.

From the list sorted by total volume from highest to lowest recorded in *s2*, selecting the filling pattern that yields the highest volume first may cause a problem. As seen in Fig.6, selecting a layout with the largest volume only might not achieve the goal of utilizing the fewest number of bins. There may be only one type of box in the pattern with the maximum volume. As shown in this figure, a pattern with the maximum volume indicates that we should pack a box of type 2 with an amount of 147. Meanwhile, in a configuration, we have 3 types of boxes with amounts 17, 7, and 76, respectively. Because of this, packing every box may not be feasible with this pattern—even if the required amount is minimal and can fit in a single bin. Therefore, we should consider the following pattern, which can fit more types of boxes into a bin and has a lower order in the sorted list. After that, evaluate them and select the finest one. That is, the actual number of boxes that remains should be considered when choosing a pattern from a sorted list. The pattern with the highest volume in this case is problematic since it states that packing a box type 2 with an amount of 147 using this pattern will yield the maximum volume. However, we cannot achieve a maximum value because there are only 7 type 2 boxes in the actual configuration.

Required number of boxes for packing		
Boxtype1	Boxtype2	Boxtype3
17	7	76

Max volumetric pattern #1		
Boxtype1	Boxtype2	Boxtype3
0	147	0

Remaining number of boxes after using pattern #1(1 Bin)		
Boxtype1	Boxtype2	Boxtype3
17	0	76

Fig.6. An example of the problem when using the maximum volume pattern.

B. *Choosing a format that fits the remaining boxes*

In this state, Y. Harrath's TSLBH model will loop through all patterns that have already been sorted by maximum volume and use each pattern to pack until all boxes are packed. To improve this stage, after sorting by maximum volume, we change the process to use only the first maximum pattern to pack until all boxes are packed. Nevertheless, if we use the pattern until one of the box types becomes 0, continuing to use it would cause the volume of the pattern to be lost, as Fig.7 illustrates. The optimal pattern now states that we must stuff a type 1 box, a type 2 box, and a type 3 box with amounts of 35, 30, and 25, respectively. This is shown in the figure. There are currently 300, 150, and 50 pieces of types 1, 2, and 3 in the remaining boxes. We must therefore pack 70, 60, and 50 pieces of box types 1, 2, and 3 into a bin in this round, correspondingly. Because of this packing, there are no more type 3 boxes. Because a type 3 box is empty, we are unable to use this pattern in the upcoming round. Therefore, it becomes necessary to find a new pattern suitable for the remaining quantity, as illustrated in Fig.8. The boxes for types 1, 2, and 3 must be packed with amounts of 50, 45, and 0, respectively, according to the next placement pattern in a sorted list. We shall employ this pattern twice. The final 3 types of boxes that will be packed in this round are 130, 0, and 0. Finally, we suppose that the following placement pattern in a sorted list instructs us to fill the boxes with 80, 0, and 0 amounts of type 1, 2, and 3, respectively. When we pack using this pattern, the algorithm reaches its final stage.

Required number of boxes for packing		
Boxtype1	Boxtype2	Boxtype3
300	150	50

Max volumetric pattern #1		
Boxtype1	Boxtype2	Boxtype3
35	30	25

Remaining number of boxes after using pattern #1(2 Bins)		
Boxtype1	Boxtype2	Boxtype3
230	90	0

Remaining number of boxes after using pattern #1(3 Bins)		
Boxtype1	Boxtype2	Boxtype3
195	60	-25

Fig.7. Wastage of space due to using the same pattern when the remaining volume of a particular type of box becomes zero.

An operation process for Stage 3 is shown in Algorithm 3. It is decomposed into 2 sub-steps.

1. Selecting the pattern from the variable *s2* in accordance with the number of boxes that remain, considering the case where there is zero of any type of box, as line 7 of Algorithm 3 illustrates. In this round, information to be stored includes the maximum values for boxes in the vertical orientation, the maximum values for boxes in the horizontal orientation, the total number of boxes placed in the bin, and the total maximum volume of all boxes as line 10 of Algorithm 3.

Remaining number of boxes after using pattern #1(2 Bins)		
Boxtype1	Boxtype2	Boxtype3
230	90	0

Max volumetric pattern #2		
Boxtype1	Boxtype2	Boxtype3
50	45	0

Remaining number of boxes after using pattern #2(2 Bins)		
Boxtype1	Boxtype2	Boxtype3
130	0	0

Max volumetric pattern #3		
Boxtype1	Boxtype2	Boxtype3
80	0	0

Remaining number of boxes after using pattern #3(2 Bins)		
Boxtype1	Boxtype2	Boxtype3
0	0	0

Fig.8. Changing to a new suitable pattern when the value of a certain type of box becomes zero.

Algorithm 3: Stage 3 Select the maximum volume to store in the bin and ensure load balance.

Step 3

1: $Bin \leftarrow 0$
2: *while* $n > 0$:
3: *for* $k \leftarrow 1$ *to* $(|s2|)$:
4: $act \leftarrow 1$
5: $canBin \leftarrow \{\}$
6: *for* $i \leftarrow 1$ *to* t:
7: *if* $(n_i > 0$ *and* $maxBoxes_{k,i} == 0)$*or* $(n_i == 0$ *and* $maxBoxes_{k,i} > 0)$:
8: $act \leftarrow 0$
9: *if* $act == 1$:
10: $canBin \leftarrow canBin \cup (\cup_{i=1}^{t} t_i, p_{k,i}, hoBoxes_{k,i}, verBoxes_{k,i}, maxBoxes_{k,i}), maxVolume_k$
11: $canBin \leftarrow \overleftarrow{vols}_{maxVolume}$
12: $actS \leftarrow canBin_1$
13: $actS \leftarrow \overleftarrow{actS}_{verWeight}$
14: $usedBin \leftarrow 0$
15: $lastBin \leftarrow 0$
16: $a \leftarrow 0, b \leftarrow 0$
17: *for* $i \leftarrow 1$ *to* n:
18: *if* $n_i > 0$:
19: $a \mathrel{+}= 1, b \mathrel{+}= 1$
20: *while* $a == b$:
21: $v \leftarrow 0$
22: $allHoBoxes \leftarrow \cup_{i=1}^{actS} hoBoxes_i$
23: *while* $\sum allHoBoxes > 0$:
24: *for* $i \leftarrow 1$ *to* $(|actS|)$:
25: *if* $hoBoxes_i > 0$:
26: $n_{actS_{t,i}} \leftarrow max(0, n_{actS_{t,i}} - verBoxes_{actS_i})$
27: $hoBoxes_i \mathrel{-}= 1$
28: $usedBin \mathrel{+}= 1$
29: *for* $i \leftarrow 1$ *to* n:
30: *if* $n_i \leq (hoBoxes_{actS_{t,i}} \times verBoxes_{actS_{t,i}})$:
31: $lastBin \mathrel{+}= 1$
32: *if* $n_i > 0$:
33: $b \mathrel{+}= 1$
34: *if* $lastBin == t$ *and* $b > 0$:
35: $usedBin \mathrel{+}= 1$

2. Select the pattern that yields the maximum volume first, as indicated in line 12 of Algorithm 3, and sort the weights of the boxes within the selected pattern for load balancing during placement, as specified in line 13 of Algorithm 3. Then, pack the boxes (n), arranging them in descending order of weight, as detailed in lines 21-28 of Algorithm 3. In each round, check the remaining quantity of each type of box, as outlined in lines 16-20 of Algorithm 3. If the remaining quantity of any type of box is zero, find a new pattern that yields the maximum volume for the remaining number of boxes. Continue this process until the total number of boxes (n) equals zero, which will complete the process.

IV. Experimental Setup And Results

The testing data used in this research is identical to that used by Y. Harrath and includes information on the weight and volume of boxes and bins. Table I provides specifics on the dimensions of the bins and boxes. A total of 27 samples, ranging from 100 to 1,000,000 boxes, were randomly created using Python's 'random' module. Python was also employed to measure the time taken for each stage, from the start of the procedure to its end, utilizing the 'time' module. To calculate memory usage, the 'memory_profiler' module in Python was used to monitor changes during code execution. This was done by calculating the difference between the maximum and minimum values at each stage. The 'memory_profiler' module provides detailed reports of memory consumption, allowing for accurate tracking and comparison of memory usage across different stages of the process. This method ensures that memory usage patterns are precisely captured and analyzed. Using Python 3.8.6 and Visual Studio Code version 1.86, the packing process in Stages 1-3 was tested on an 11th Gen Intel (R) Core (TM) i5-1135G7 @ 2.40GHz Windows 11 64bit Asus Vivobook with 8 GB RAM.

TABLE I. Volume And Weight Of Bins And Boxes

Type	Cm			Kg
	Depth	Height	Length	Max.weight
Bin	243.8	178	317.5	6804
BoxType1	40.16	32.86	25.88	10
BoxType2	54.8	42.1	33.5	25
BoxType3	22.2	19.7	28.7	13

Table II displays the outcomes of Stage 1, which involves calculating the maximum number of boxes that may be arranged horizontally based on type and orientation and entering that information in variable *s1*. The outcomes of this stage aligned with the methodology outlined by Y. Harrath.

TABLE II. Experimental Results Of Stage1

Package Type	Dimension			Maximum number of boxes in horizontal orientation packed into the bin		
Bin	D	L	H	Box type 1	Box type 2	Box type 3
Box rotation 1	d	h	l	72	36	110
Box rotation 2	d	l	h	54	28	160
Box rotation 3	h	d	l	84	45	132
Box rotation 4	h	l	d	49	25	168
Box rotation 5	l	d	h	81	49	128
Box rotation 6	l	h	d	63	35	112

Stage 2 of the proposed method involved figuring out how to arrange mixed-type boxes vertically. In order to cut down on unnecessary data storage, this research approach introduced a process that only included patterns in which the overall weight and height of the data did not exceed the maximum capacity of the bin. Only the highest-value patterns were chosen to be stored. As a result, compared to Y. Harrath's second-stage method, the number of patterns saved in variable *s2* was reduced from 13,480 to 3,744.

The following is an experimental result from Stage 3 that compares the packing procedure with Y. Harrath's method (TSLBH):

- Comparative analysis of bin usage: The outcomes were comparable for quantities between 100 and 10,000 boxes. However, Y. Harrath's TSLBH procedure tended to require more bins for quantities over 25,000, as it used a sequential volumetric arrangement strategy, causing patterns recorded subsequently to lose volume, as indicated in bold type in Table III. The proposed method, on the other hand, increases efficiency per cycle by selecting the optimal pattern and filling bins until a particular type of box runs out. The best result is a reduction from 4,488 to 2,752 bins, a decrease of 38.68%, as shown in Table III.
- Time efficiency comparison: When testing with different numbers of boxes, as shown in Table III, the proposed method's total execution time is also improved as it can decrease the number of patterns saved in variable *s2* from 13,480 to 3,744, from our experiment finding. Furthermore, our proposed method's overall time consumption is significantly less than the TSLBH when the total number of boxes exceeds 10,000, as indicated in bold type in Table III. The best result is a reduction from 543.02 to 225.78, which is a decrease of 58.42%.
- Comparative Analysis of Space Efficiency: The efficiency of bin usage is determined by equation (1).

$$\text{Efficiency} = \left(\frac{Total\ volume\ of\ all\ boxes}{Total\ volume\ of\ all\ bins\ used}\right) x100 \quad (1)$$

An illustration of effective bin usage would be if we were to pack 10 boxes, each of which has a volume value of 6, in the bins, which have a volume value of 40. As a result, we will fill the first bin with 6 boxes and the second bin with 4 boxes. Thus, the total volume of boxes is 60, which is equivalent to 36 in the first bin plus 24 in the second bin. Additionally, the total volume of the bin is 40 + 40 = 80. In conclusion, the efficiency of bin usage is (60/80)*100 = 75. We also see that in this instance, the weight of each bin is not less than the sum of the weights of all the boxes inside.

This measure shows how well the space is used inside the bins. The efficiency is initially lower because there is unused space in the bins, particularly in the last bin, which is not completely filled. On the other hand, efficiency rises and differences get smaller as the number of bins grows. When the quantity is between 100 and 10,000 boxes, the differences are not significant. However, Y. Harrath's TSLBH procedure tends to use more bins for quantities exceeding 25,000, as illustrated in all rows with bold font in Table III. This is due to its sequential volumetric arrangement strategy, which leads to inefficiencies in later stages of the packing process. Y. Harrath's approach stores patterns in the variable *s2*, sorting them by volume from highest to lowest. The packing process proceeds sequentially, using each pattern until all boxes are packed. While this approach works well for smaller volumes, as the number of boxes increases, the efficiency drops because later patterns tend to have less volume. Like Y. Harrath's method, the proposed approach, also sorts the patterns in *s2* by volume. However, it diverges by selecting only the maximum-volume pattern and using it exclusively until all boxes are packed. This ensures that the bins are filled with the highest possible volume, increasing efficiency per packing cycle. With the proposed method, the efficiency remains

higher, particularly as the quantity of boxes grows. The best result is an increase in efficiency of 35.38%, as shown in Fig.9, which illustrates this tendency. Lastly, Fig.9 compares the space usage efficiency of our proposed approach with the TSLBH method. It is evident that our proposed strategy works better than the TSLBH method when there are more than 25,000 boxes, as indicated by every curve on the right side of the blue bold line.

TABLE III. COMPARISON TABLE OF TIME RESULTS AND NUMBER OF BINS USED BETWEEN PROCESS TSLBH Y. HARRATH AND THE PROPOSED METHOD

Number of boxes obtained from randomization				TSLBH,Y. Harrath			The proposed method		
ɔtal boxes	Box type 1	Box type 2	Box type 3	Number of bins used	Total time used(ms)	Total efficiency (%)	Number of bins used	Total time used(ms)	Total efficiency (%)
100	8	16	76	1	398	17.88	1	182	17.88
500	102	164	234	2	401	69.30	2	190	69.30
1,000	13	386	601	5	401	54.90	5	192	54.90
2,000	92	1,535	373	12	401	76.48	12	193	76.48
2,500	1,358	38	1,104	6	416	76.42	6	193	76.42
3,000	1,460	1,523	17	15	401	81.18	15	202	81.18
4,000	2,309	1,238	453	16	401	81.75	16	193	81.75
5,000	4,866	30	104	15	399	82.16	15	193	82.16
6,000	2,893	1,341	1,766	19	401	85.80	19	192	85.80
7,000	1,077	723	5,200	16	416	71.64	15	193	76.41
7,500	1,654	5,359	487	42	401	82.39	42	197	82.39
8,000	2,400	753	4,847	18	420	81.05	18	189	81.05
9,000	5,016	2,455	1,529	32	401	86.24	32	193	86.24
10,000	5,898	2,954	1,148	37	402	87.12	37	197	87.12
25,000	**3,988**	**4,309**	**16,703**	**61**	**417**	**80.77**	**57**	**199**	**86.44**
50,000	**5,715**	**22,470**	**21,815**	**196**	**407**	**81.67**	**190**	**194**	**84.25**
75,000	**3,422**	**15,917**	**55,661**	**193**	**418**	**76.93**	**174**	**194**	**85.33**
100,000	**57,266**	**8,580**	**34,154**	**260**	**405**	**85.07**	**241**	**196**	**91.78**
150,000	**114,522**	**960**	**34,518**	**406**	**419**	**78.99**	**352**	**199**	**91.11**
200,000	**23,968**	**41,091**	**134,941**	**534**	**422**	**77.31**	**477**	**197**	**86.55**
300,000	**52,154**	**212,640**	**35,206**	**2,453**	**491**	**55.20**	**1,617**	**205**	**83.74**
350,000	**33,925**	**144,009**	**172,066**	**1,705**	**452**	**61.50**	**1,240**	**206**	**84.57**
500,000	**26,971**	**85,455**	**387,574**	**1,389**	**441**	**64.74**	**1,116**	**208**	**80.58**
650,000	**261,151**	**130,932**	**257,917**	**2,348**	**453**	**68.85**	**1,814**	**215**	**89.12**
800,000	**550,853**	**47,681**	**201,466**	**3,123**	**491**	**58.16**	**1,982**	**230**	**91.64**
900,000	**244,862**	**69,289**	**585,849**	**2,282**	**441**	**67.02**	**1,831**	**210**	**83.52**
1,000,000	**791,401**	**77,784**	**130,815**	**4,488**	**543**	**56.09**	**2,752**	**226**	**91.47**

TABLE IV. MEMORY USAGE ANALYSIS BY COMPARING EACH STAGE BETWEEN TSLBH,Y. HARRATH AND THE PROPOSED METHOD

Memory Usage Analysis by Comparing Each Stage (MiB)						
	TSLBH,Y. Harrath			The proposed method		
	Stage1	Stage2	Stage3	Stage1	Stage2	Stage3
Max memory used after test 10 times	0.00198	0.08203	0.35156	0.00199	0.08984	0.12891
Max memory Used	0.35156			0.12891		
Reduction from Y. Harrath's TSLBH Patterns: **63.33%**						

- The memory usage was measured in Mebibytes (MiB). At Stage 1 and Stage 2, the memory usage did not differ significantly between the processes. However, at Stage 3, the proposed method demonstrated lower memory usage Overall, the proposed method reduced memory usage by 63.33%, as shown in Table IV.

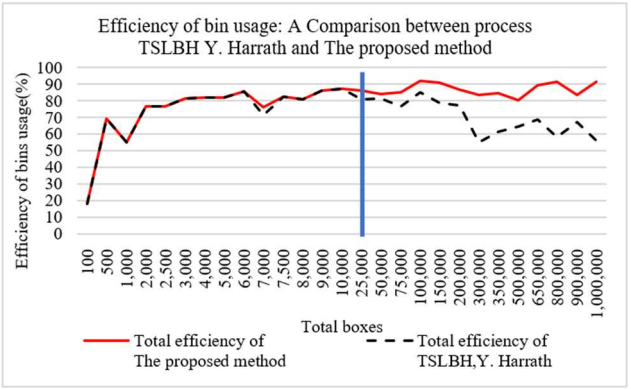

Fig.9. Comparison of space utilization efficiency between process TSLBH Y. Harrath and the proposed method.

V. CONCLUSION

The conceptual framework of the Three-Stage Layer-Based Heuristic (TSLBH) technique for the bin-packing problem by Y. Harrath is improved by this paper. The algorithm we propose uses the same method as Y. Harrath's TSLBH in the first stage. However, in the second and third stages, we made improvements. Y. Harrath's TSLBH method keeps all possible placement patterns, while our proposed method retains only the patterns that result in the maximum volume for each placement pattern that can be packed into the bin according to the packing conditions. The experiment demonstrates that our proposed method can reduce the number of bins used. Additionally, it also reduces the storage space, total experiment time and memory usage. However, this paper does not consider scenarios where we have non-cubic box shapes or complex packing restrictions, such as not being able to stack or rotate boxes, which we will address in our further work.

VI. REFERENCES

[1] Y. Harrath, "A Three-Stage Layer-Based Heuristic (TSLBH) to Solve The 3D Bin-Packing Problem Under Balancing Constraint," Journal of King Saud University, vol 34, Issue 8, Part B, Sep.2022.

[2] B.Korte and J.Vygen, "Combinatorial optimization," vol. 2. Springer ,2012.

[3] C. T. Ha, T. T. Nguyen, L. T. Bui and R. Wang, "An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the Physical Internet," in European Conference on the Applications of Evolutionary Computation, Evo Applications 2017, Amsterdam, The Netherlands, 2017.

[4] C. Paquay, M. Schyns and S. Limbourg, "Three Dimensional Bin Packing Problem applied to air cargo," Open Repository and Bibliography – Liège, Belgian city Belgium, 2011.

[5] J.Ebrahim, J.Kaabi and Y. Harrath, "Multilayers placement algorithm to solve multiple objectives 3d bin-packing problem," in Proceeding of the 8th International conference on modeling, Simulation and Applied Optimization. University of Bahrain, IEEE, Kingdom of Bahrain, 2019.

[6] M. Lixin, W. Wei, Z. Tong, T. Xincheng and J. Yong, "A greedy online 3D bin packing algorithm based on multi-indicator fusion," SPIE - The International Society for Optical Engineering, vol 13171, id. 1317103, pp 7, Jun.2024.

[7] S. Yaru, Y. Yanming, C. Shuhao and Y. Wenshuo, "A Hybrid Chemical Reaction Optimisation Algorithm for Solving 3D Packing Problem," The International Journal of Autonomous and Adaptive Communications Systems, vol 14, No. 1/2, pp 117 – 131, Apr.2021.

[8] L. Duan, H. Hu, Y. Qian, Y. Gong, X. Zhang, J. Wei and Y. Xu, "A Multi-task Selected Learning Approach for Solving 3D Flexible Bin Packing Problem," the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), pp 1386-1394, May.2019.

[9] M. Hifi, S. Negre and L. Wu, "Hybrid greedy heuristics based on linear programming for the three-dimensional single bin-size bin packing problem," International Transactions in Operational Research, vol 21, Issue 1, pp 59-79, Dec.2013.