COMPUTER-AIDED DESIGN & APPLICATIONS, 2018
VOL. 15, NO. 1, 11–24
https://doi.org/10.1080/16864360.2017.1353728

A two-dimensional bin packing-based split-and-pack approach for decomposing large three-dimensional structures into convex items

Erkan Gunpinar

Istanbul Technical University, Turkey

ABSTRACT

In this study, we present a technique to pack a three-dimensional (3D) structure into bins to minimize bin waste. The proposed approach is different from two-dimensional bin packing (2Dbp) methods: Rather than starting with fixed size items of the 3D structure (*with widths or heights smaller than those of the bin*), as is done in 2Dbp algorithms, *combined items* (*with widths or heights that can be greater than those of the bin*) are utilized. These items are obtained by combining the neighboring items. A method of generating combined items from a 3D structure is first explained. The packing approach for the combined items obtained is then described. Four operators are introduced for combined items' packing. Packing positions (P_p) on bins are computed by P_p finder. Following this, the orientations of combined items are changed via orientation modifier to find better packing orientations for combined items. Split operator splits combined items during packing if they exceed the bin boundary. Placement decision-maker (Pd-maker) determines which combined item is placed on which packing position via a cost function attaining lower values if packing produces less waste. A shipbuilding problem is utilized throughout the paper to explain the proposed method. A given ship hull is packed into bins, and experiments show that the proposed approach is advantageous in terms of minimizing bin waste.

KEYWORDS

Computational geometry; Two-dimensional bin packing; Optimization

1. Introduction

This study investigates an extended version of the two-dimensional bin packing (2Dbp) problem, which can be employed for packing large three-dimensional (3D) structures into bins. 2Dbp methods pack items into identical bins, each with length L and width W. Each item j is defined by a length l_j and width w_j, where $l_j \leq L$ and $W_j \leq W$ (i.e., *item size constraint*), for $1, \ldots, n$. Here, the overall goal is to minimize the total waste of bins, with no items overlapping with other items and all items contained in the bins. Here, rather than starting with fixed-size items (with lengths or widths that are smaller than those of the bin), as is done in 2Dbp algorithms, we argue that it is advantageous to utilize larger items (obtained by combining neighboring items) whose lengths or widths can be greater than those of the bin. These large items can be dynamically split into smaller items while packing.

Figure 1 illustrates the basics of the proposed approach. Let us consider a two-dimensional (*2D)* structure (Fig. 1(a)) that has to be produced using *2D* flat bins (Fig. 1(b)). One way to accomplish this is to decompose this structure into small items manually while considering shape of the given *2D* flat bins. These items can then be packed into the bins. The gray items shown in Fig. 1(c) and (d) are packed into the bins without producing any bin waste. In contrast, the items in pink are packed into the bins with some amount of bin waste (Fig. 1(e)). When the proposed approach in this study is applied, the items from (1) to (12) are combined while considering their neighboring relationships; thus, three combined plates (in blue) are obtained (Fig. 1(d)). These combined plates are split at the required location during packing, which can decrease bin waste production. The items (2) and (3) are split wherever required, and no bin waste is produced (Fig. 1(f)).

The remainder of the paper is organized as follows. Section 2 reviews relevant literature, while Section 3 illustrates a shipbuilding application and details the generation of combined items. Section 4 describes the Split-and-Pack approach, and numerical results of the proposed method are given in Section 5. Concluding remarks and opportunities for future work are presented in Section 6.

2. Related literature

2Dbp techniques pack a given set of 2*D* rectangles io rectangular bins while minimizing the number of bins used. Gilmore et al. [12] proposed the first model for

CONTACT Erkan Gunpinar gunpinar@itu.edu.tr

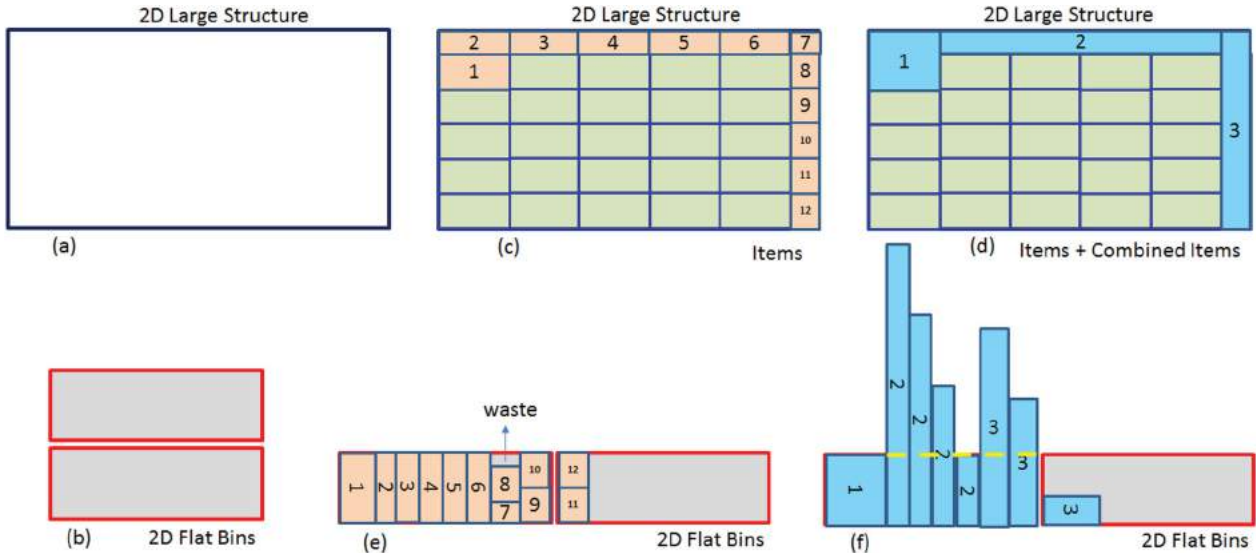

Figure 1. A two-dimensional (2*D*) structure (a) is produced using 2D flat bins (b). Items shown in gray and pink (c) are packed into bins with some bin waste (e). Once combined items (in blue) obtained by merging neighboring items (d) are utilized, no bin waste is produced, as the items can be split at the required location (see dashed yellow line in (f)).

2Dbp problems. Beasley [3] considered the cutting stock problem, which is a variant of the 2Dbp problem, and formulated it based on an integer linear programming formulation. Other studies [2, 17] proposed a method for 2Dbp problems where guillotine cuts are utilized. Guillotine cuts are edge-to-edge cuts parallel to the edges of bins.

There are off-line and on-line 2Dbp algorithms: Off-line algorithms have full knowledge of the input item set, as in the present work; in contrast, on-line algorithms [9] pack items as soon as they are encountered without knowledge of the subsequent items. In the literature, on-line algorithms are classified into two categories, namely one-phase and two-phase algorithms. One-phase algorithms (i.e., 2Dbp) pack the items into finite bins, whereas two-phase algorithms [2, 8, 11] (i.e., strip packing) pack items into a single strip (i.e., bin with a specific width and infinite height). The problem of the present study requires a one-phase algorithm. *Level algorithms* are generally used for bin or strip packing; here packing is achieved by placing items, from left to right, in rows forming levels. Three methods for level packing have been proposed for the one-dimensional (1*D*) case. Next-fit Decreasing Height (NFDH) packs items in a left-justified manner if they fit. Otherwise, a new level is formed and a new item is packed left justified to that. In the First-fit Decreasing Height (FFDH) approach, items are left justified on the first level where they fit. If there is no fit, a new level is initialized, as in NFDH. Best-fit Decreasing Height (BFDH) packs items left justified on the current level, among those that fit, for which the unused horizontal space is minimized. If there is no level where the items fit, a new level is initialized, as in NFDH. Frenk et al. [11] proposed the Hybrid Next-fit (HNF) algorithm, where NFDH is utilized in the first phase. The 1*D* bin-packing problem is then solved by the Next-fit Decreasing (NFD) algorithm in the second phase, which packs an item to the current bin if it fits and starts a new bin otherwise. Several one-phase algorithms were also proposed by Berkey et al. [4]. Finite Next-fit (FNF) directly packs the items into finite bins in the same way as HNF. Finite First-fit (FFF) packs items on the lowest level of the first bin where they fit; if no level can accommodate an item, a new level is formed in the first bin with sufficient vertical space. Otherwise, the new level is formed in a new bin. Finite Bottom-left (FBL) and Next Bottom-left (NBL) are two other one-phase bin-packing approaches mentioned in the paper. Readers can refer to the work of Lodi et al. [21] for additional 2Dbp techniques.

More recently, Honga et al. [19] presented a mixed packing algorithm that combines a heuristic with Best Fit and is based on simulated annealing and binary searching. Beyaz et al. [5] used state-of-the-art meta-heuristics and local search techniques to minimize the number of bins while maintaining the load balance. Martinez Sykora et al. [23] packed a set of irregular pieces into bins with fixed dimensions in such a way that the utilization was maximized. Their procedure allows free orientation for the pieces. Zhao and Shen [26] presented a 2*D*. online rectangular packing algorithm with three bits of advice per item. Lai et al. [20] proposed a heuristic approach for 2*D* bin-packing problems with a guillotine-cut constraint while considering three

criteria, including packing priority and packing spaces. Bansal and Khan [1] studied packing of 2*D*. rectangular items with and without rotations. As stated in the paper, their technique achieves 1.405 approximation for the bin-packing problem. The bin-packing technique of Camachoa et al. [7] handles both 1*D* and 2*D* problems involving irregular concave polygons. The method was based on a hyper-heuristic methodology. Trivella and Pisinger [25] formalized the load-balanced, multi-dimensional bin-packing problem using mixed-integer linear programming models.

Nesting algorithms [10, 13, 18, 27] are used for packing items into bins. These all start with a fixed size of items, which are then packed into bins. Zheng [27] approximated blocks as isosceles trapezoids, right trapezoids, parallelograms, and triangles before the packing process and utilized these approximated blocks during the packing process. This study explored the effective block spatial layout under the condition of spatial constraints. Hana et al. [18] generated guillotine-cutting layouts of irregular shapes. Pieces were first enclosed within rectangle shapes, and then the rectangles were packed. However, additional waste was produced using this methodology. Griffith [13] provided techniques for decomposing 3D geometry into developable surface patches and cut patterns. In this method, a decomposition application receives a triangulated 3D surface as input, and approximately developable surface patches are determined from the 3D surface using a type of k-means clustering. Dash et al. [10] proposed an automated method that optimally designs plates to satisfy an order book at a steel plant. Here, the yield of the designed plates can be maximized while using capacity fully to satisfy order deadlines.

We formulate the bin-packing problem differently from the way it is considered in 2Dbp techniques. Instead of starting with a given set of items, neighboring items are combined, and items with large sizes are obtained (called *combined items*). In 2Dbp techniques, the width and length of items should be smaller than those of the bins before packing. Hower, combined items may have larger sizes than the bins. Therefore, items are split wherever required while satisfying the minimum length and angle criteria. Here, we propose a new set of algorithms for decomposing a large *3D* structure into convex items. Combined items are produced first and these are utilid as a set of input items for an extended version of the 2Dbp technique. Items are dynamically split into sub-items during packing. We think that utilizing this approach can better minimize bin waste, since the splitting process splits combined items at the required locations.

3. An application and generation of combined items

3.1. A shipbuilding problem

A ship hull is a large 3*D* structure and cannot be produced all at once. Therefore, it must be decomposed into producible plates that are manufactured individually. These plates are then assembled and welded to produce the hull. Producible plates are generated using rectangular flat plates, and waste minimization of these plates is one of the most important criteria. There are inherently more criteria, such as cost estimation and minimization of welding length between the producible plates, for ship hull decomposition. However, only the waste minimization criterion of the rectangular flat plates is considered for the validation of the proposed approach.

A *ship hull, H*,. is given as a 3*D* triangular mesh model and consists of 3*D producible plates* on *H*. A producible plate can be a triangular region (with three corners and three boundaries) or a quad region (with four corners and four boundaries), as plates in triangular and quad geometries are easy to bend. Let *Q* be a set of producible plates, and $H = \cup_i Q_i$, $Q_j \cap Q_k = \emptyset$. (i.e., no overlap between Q_j. and Q_k), $Q_j, Q_k \in Q.$, $j, k = 1, \ldots, a$, $j \neq k.$, and *a* be the number of producible plates. A *flattened plate*, called a *hull item*, is a planar triangular or quad region, which is produced by flattening the producible plate to 2*D*. The geometric characteristics (i.e., *quality criteria*) of hull items are as follows:

- *The maximum length constraint:* Each edge length of a hull item should be smaller than l_{max}, which varies according to the item position. Hull items residing on less curved portions of the ship hull have greater values of l_{max} than those on highly curved portions. Note that hull items with greater edge lengths are difficult to bend and transport;
- *The minimum length constraint:* Each edge length of a hull item should be greater than l_{min}. Items with smaller edge lengths are difficult to weld and bend;
- *The minimum angle constraint:* The angle between consecutive edges of a hull item should be greater than α_{min}. Hull items with smaller angles of α_{min} are difficult to weld.

A ship hull is produced from rectangular flat steellates called *bins*, and items are packed into these bins while minimizing waste. The research question here is how to obtain a set of hull items for a given ship hull while minimizing bin waste. Starting from a mesh model (*H*), the ship hull is first segmented into non-overlapping *mega blocks* as shown in Fig. 2 user constraints depicted in

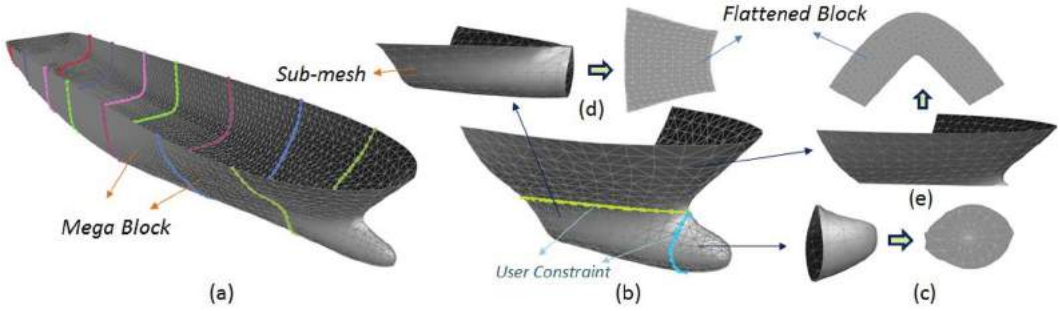

Figure 2. Required preprocessing steps before the generation of combined items and Split-and-Pack approach: Mega blocks are generated from a given mesh model of a ship hull (a) (of about 320 *m* in length). The mega block is cut into several sub-meshes based on the user constraints in (b), and the 3*D* mesh models of these blocks are flattened to 2*D* (c, d, e).

Fig. 2(b) are defined by the manufacturing or design engineer to determine the sub-mesh boundaries of the mega block (see the 3*D* mesh models in Fig. 2(c, d, e)). For the sake of simplicity, only a few user constraints are utilized in this study, but there can be more in practice. Finally, sub-meshes between these constraints (boundaries) are flattened to 2*D* using the ReSurf MeshFlatten standalone application [24], and *flattened blocks* are obtained; these are represented using planar *2D* meshes (see the 2*D*mesh models in Fig. 2(c, d, e)). These flattened meshes will be utilized to produce combined items.

3.2. Combined items' generation

3.2.1. Bin template formation

Flattened blocks are covered using bin templates of equal size in this step. Bin templates are produced from bins in such a way that they do not generate bin waste while packing. Assume that we have a bin with length L. and width W. Bin templates with length l. and width w. are generated from bins, where $l \leq L$ and$w \leq W$. Bin templates will be used to cover the flattened blocks. Here, l. and w are determined based on the minimum and maximum length constraints (l_{min} and l_{max}, respectively) described in Section 3.1. Figure 3 illustrates the generation of bin templates from bins. Starting from a bin (see Fig. 3(a)), horizontal and vertical cuts are generated (in pink and green in Fig. 3(b)). These cuts divide the bin into equally-spaced regions, each of which will be a bin template, as shown in Fig. 3(c). To compute l and w., the number of required cuts in the horizontal and vertical directions (n_h and n_v) are first determined, and these are calculated as follows: $n_h = f(W/l_{max})$ and $n_v = f(L/l_{max})$., where $f(x)$ maps a real number x to the largest previous integer. Finally, l and w are computed using n_h and n_v: $l = L/(n_v + 1)$. and $w = W/(n_h + 1)$. This formula favors the generation of larger length and width of equal-sized bin templates. Bins with a width of 2.5 m and a length of 12.0 m are utilized for this study's problem. In Fig. 2(c, d), l_{max} is defined as 2.0 *m* for blocks, which have highly curved regions. Fig. 2(e), l_{max} is set to 3.0 *m* for the block. For all models throughout this paper, l_{min} and α_{min} are set to 0.5. *m* and 0.35. radian (about 20. degrees), respectively. In these parameter settings, bin templates with a length of 2.0 *m* and a width of 1.25 *m* are generated for the blocks in Fig. 2(c, d). In Fig. 2(e), a length of 3.0 *m* and a width of 2.5 *m* the bin templates are generated for the block.

3.2.2. Initial covering of flattened blocks

this step, bin templates are placed on each flatteneock, and a regular grid consisting of the bin templates will be obtained (see Fig. 4(c)). In this grid structure, bin templates residing on the inner portion the Flattened block always have four neighbor bin templates (see rectangles in green). First, the orientation of the flattened block in Fig. 4(a) is changed so that its boundary is as much as parallel to X, Y axes (Fig. 4(b)), as bin templates are also

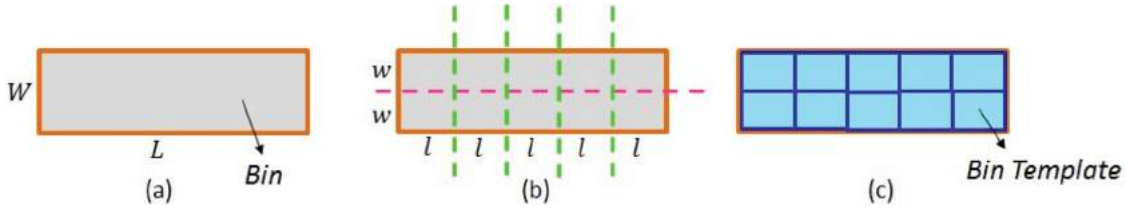

Figure 3. Equally-spaced horizontal and vertical cuts shown with pink and green dashed lines in (b) divide the input bin in (a) and bin templates (c) are produced.

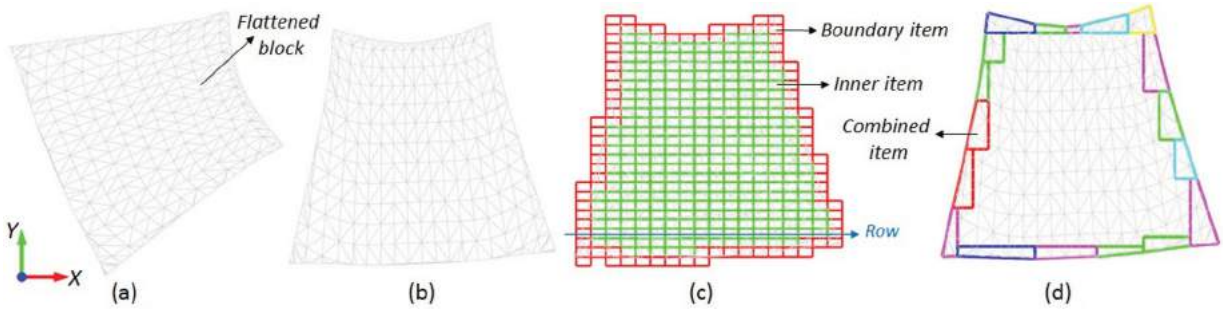

Figure 4. The flattened block in (a) is oriented according to *X*- and *Y*- coordinate axes to make the boundary as parallel as possible to these axes (b). The flattened block is covered with bin templates; boundary and inner items are marked in red and green (c), respectively. The combined items are generated by merging boundary items with their neighboring items (d).

oriented according to the X, Y axes (see Fig. 4(b)). The oriented flattened block is then covered using bin templates, as depicted in Fig. 4(c). Bin templates are placed on the flattened block without overlapping each other, and a $2D$ regular grid consisting of these bin templates with uniform grid spacing in the X or Y direction is obtained. Starting from the left-st vertex (*initial vertex*) of the flattened block, bin templates are placed from left to right (i.e., in the positive X direction) until the right edge of the placed bin template passes the right edge of the flattened block's bounding box. Thus, a single row of the flattened block is covered with the bin templates. Uppower rows of the flattened block are similarly covered by moving the initial vertex by the width W of the bin template in the positive/negative Y direction. Covering the flattened block with the bin templates (in the positive/negative Y direction) stops when there is no intersection between the placed bin templates and the flattened block. Finally, bin templates with no intersection with the flattened block are removed; those that intersect with the flattened blk boundary are marked as *boundary items*, and those without intersection with the boundary are marked as *inner items*. Figure 4(c) shows the results oe flattened block covering method; the boundary and inner bin templates are depicted with red and green rectangles, respectively.

3.2.3. Generation of combined items

Boundary items generate bin waste, since they are located on the flattened block boundary. *Combined items*—triangular or quad convex regions—will be replaced with boundary items to reduce waste on the flattened block boundaries. For each boundary item, an overlapping region between the item and the flattened block is first computed. An enclosing region of the overlapping region is then found, which is a minimum enclosing triangular or quad convex region [22] of the overlapping region; this is called a *trimmed item*. Combined items will be generated by merging trimmed items with their neighboring items. Figure 5(a) depicts a trimmed item (in yellow) with its neighboring items (in blue). Suppose A and B are two neighboring items (see Fig. 5(b)), and these are merged to obtain a single combined item, C (Fig. 5(d)). The resulting combined item C is a minimum enclosing triangular or quad convex polygon that encompasses A and B. Let D be a polygoan be either convex or concave, and encompasses A and B without any waste (Fig. 5(c)). The waste W occurring after merging A and B to obtain the combined item C is computed as follows: $W = \varphi(D)\text{-}\varphi(C)$, where $\varphi(D)$ and $\varphi(C)$ represent the area of D and C, respectively (Fig. 5(e)).

Let τ_B and τ_I. denotes the set of trimmed and inner items, respectively. List T stores these items. Here, τ_B^u is

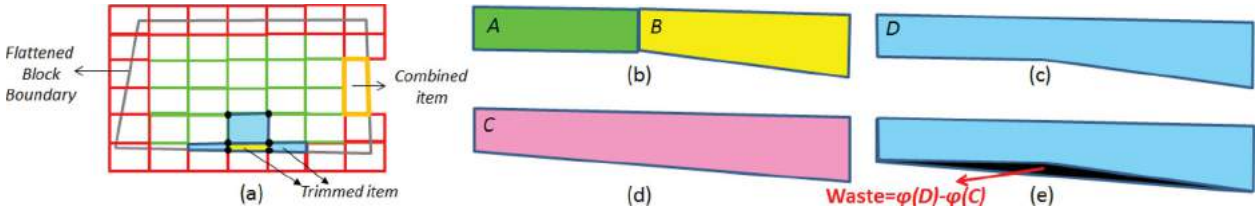

Figure 5. Trimmed items are obtained by trimming the boundary items in red (a). Two neighboring items (b), one of which should be a trimmed item, are merged to obtain a combined item (d). The waste (e) after merging is computed by subtracting the combined item from the polygon encompassing the neighboring items (c).

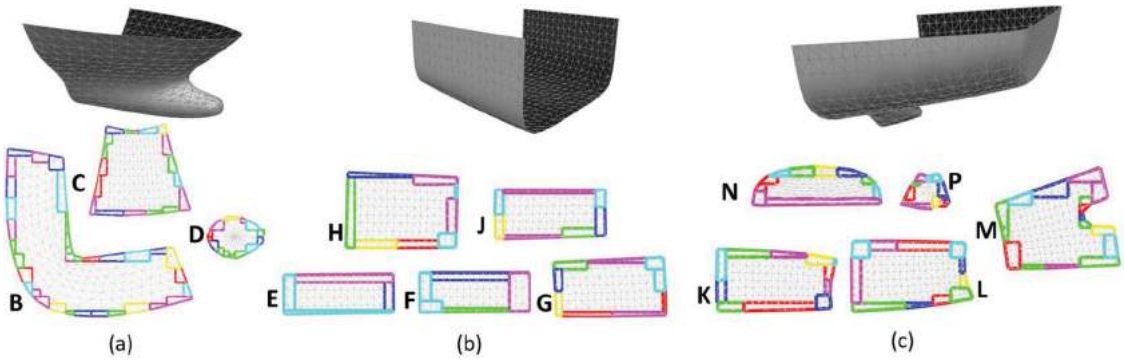

Figure 6. Combined items (in colored boundaries) of mega blocks residing in the front (a), middle (b), and back (c) portion of the ship hull model in Fig. 2(a).

Table 1. *g*, *n*, *A* and *W* denote Flattened block name shown in Fig. 6, total number of the generated combined items, total area of the generated combined items in m^2, total waste area in m^2 after merge respectively. l_{max} is the maximum length constraint in *m*.

g	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*J*	*K*	*L*	*M*	*N*	*P*
l_{max}	3	2	2	4	4	3	3	3	3	3	4	4	2
n	34	14	18	4	5	9	8	7	11	11	16	10	11
A	510.7	86.0	186.7	217.0	267.0	206.1	289.8	198.1	250.3	279.8	332.6	194.4	68.3
W	2.2	2.6	1.0	0.003	0.001	0.4	0.96	0.58	1.96	2.97	0.51	3.65	2.0

the *infeasible trimmed item* set, where items do not satisfy the minimum edge length and angle criteria introduced in Section 3.1. Other items are stored in a *feasible trimmed item* set denoted by τ_B^f. The item merging rules are as follows:

- To avoid merging, items in τ_B^f only merge with items in τ_B. and do not merge with items in τ_I.
- Items in τ_B^u. are allowed to merge with both items in τ_I and τ_B to eliminate infeasible trimmed items.

The list O stores all possible merging item pairs, and if the computed waste is greater than a user defined parameter ξ, the merging item pair is not added to the list O. Merging operations in O are grouped into two categories, namely O_1 and O_2. Here, O_1 stores the merging item pairs where a feasible combined item (i.e., satisfying the minimum length and angle criteria) is obtained after merging items, one of which is an infeasible item; O_2 contains the rest of the merging item pairs in O. In addition, O_1 and O_2 are sorted by waste occurring after the merge operation in an increasing order. Merging item pairs in O_1 are appended to O first, and then those of O_2 are appended. In this way, items in τ_B^u can be merged rapidly. The first item pair in O is merged first, and then the list O is rearranged similarly to the method described above. Items are merged until no item pair in O is merged and combined items are finally obtained. Figure 4(d) and Fig. 6 show that the combined items generated using the proposed approach. ξ is set to 0.5 in this study and can be adjusted to other values. Setting it to higher values allows the generation of large-size combined items, but waste will probably increase. Readers should refer to Table 1 for the further results of the combined item generation process.

4. The split-and-pack approach

The combined items produced in the previous step will be packed into bins while minimizing bin waste. Note that inner bin templates do not produce any bin waste as they are obtained by dividing bins into equal widths and lengths of regions without a gap between them. Figure 7 depicts the generated combined items (Fig. 7(a)) and inner items (Fig. 7(b)). The black polygon Fig. 7(b) encloses inner items, and a single bin is sufficient when 12 inner items are packed into it, as shown in Fig. 7(c).

4.1. Method overview

Initially, we have a set of combined items ψ and a finite number of equal-sized bins. A combined item is a quad or a triangle represented by its corner vertices. A bin ρ is a polygon initially having four corner vertices in $2D$ with length L and width W, and the number of corner vertices changes while packing combined items into the bin. Combined items in ψ are placed on ρ separately, and the cost for each placement is computed using the cost function introduced in Section 4.5. The

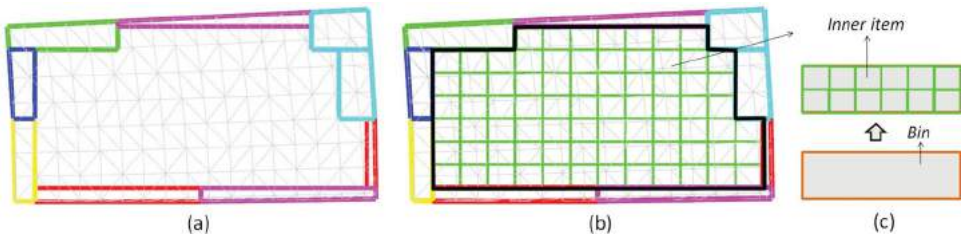

Figure 7. Combined items (a) and inner items enclosed by a black polygon (b). Inner items can fit perfectly into the bins without producing any bin waste (c).

item with a minimum cost value is placed first. The shape of ρ is modified by subtracting ρ from the placed combined item.

Candidate points where combined items are placed and their corresponding *candidate regions*, which are convex triangle or quad regions; are found using ρ. As combined items can be larger than candidate regions, some portions of these items can exceed the candidate regions. In this case, the intersection between the combined items and the candidate region is found, and the intersected portion is packed into ρ. The exceeded portion is then split into quad or triangular items, each of which will be separately packed into ρ in the next steps. If ρ does not have enough space to insert combined items, a new (unused) bin is utilized (set ρ. as an unused bin) for the further combined item placements. Packing of combined items is performed until all combined items are placed. Figure 8 shows a flow chart for the proposed approach. Readers should refer to the pseudo-code below for further details on the Split-And-Pack approach.

SplitAndPack(ψ, ρ) /* Inputs; ψ. : A set of combined items, ρ: Bin. Output; ς: Packed item list*/

(1) Set ρ as an *unused* (new) bin.
(2) Find all candidate points and their corresponding candidate regions in ρ.
(3) Place all combined items ψ. on each candidate region separately and compute cost value for each placement.
(4) Select the combined item ι having minimum cost.
(5) Find the intersected polygon φ between $\iota$and the selected candidate region. Place φ on ρ.
(6) **If** ι exceeds the candidate region, **then**
- Add φ to ς.
- Find exceeded portion ι' by subtracting $\iota$from the candidate region.
- Set ι as ι'.
- Split ι' into quad or triangular items, each of which will be separately placed on candidate regions in the next steps.

(7) Elseρ
- Add ι to ς.
- Mark ι as *placed*.

(8) Update ρ by subtracting ρ from φ.**If** has insufficient space to place the combined items in ψ, **then**
- Set ρ as a new (unused) bin.

(9) Repeat lines 2 to 10 until all combined items are placed on ρ.

4.2. Packing position (P_p) finder

P_p finder computes packing positions and packing regions of the bin ρ. A *candidate point* is a point on which combined items are placed. A *candidate region* is a corresponding convex quad or triangularegion for a candidate point into which combined items are packed, and it is generated from ρ; ρ is a polygon consisting of a sequence of points from P_1 to P_t in a counter-clockwise order (with edges connecting the consecutive vertices). That is, $\langle P_1, P_2, \ldots, P_t \rangle$, where P_j represents a point in $2D$

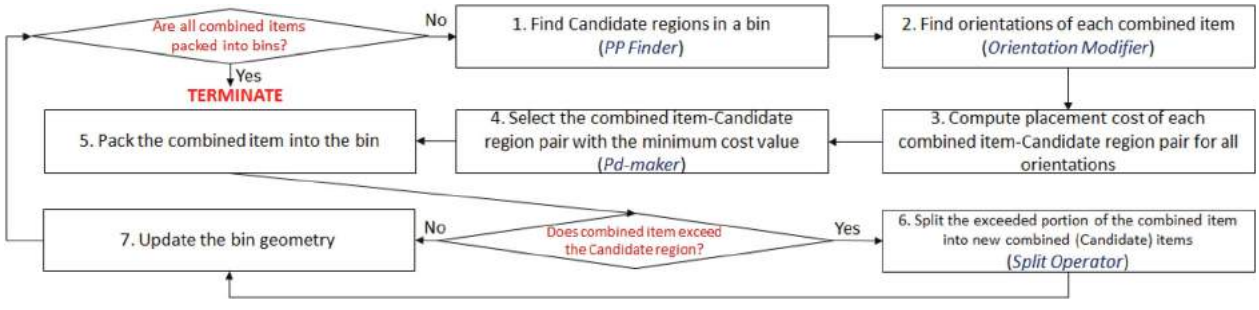

Figure 8. Flow chart for the proposed approach.

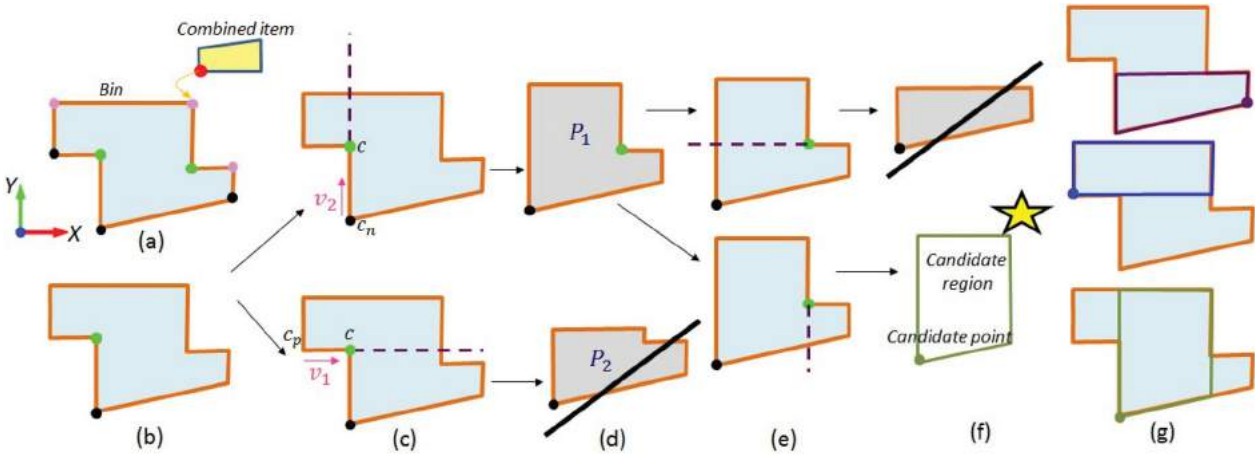

Figure 9. For each candidate point (see black points in (a)) in a bin, candidate regions are generated. Concave vertices of the bin are eliminated one by one. In each elimination, the resulting polygon with a greater area is selected (b-f). Finally, a candidate region is found for the candidate point (f). Three candidate regions with their corresponding candidate points (boundaries are shown with dark green, purple, and blue boundaries) are obtained (g).

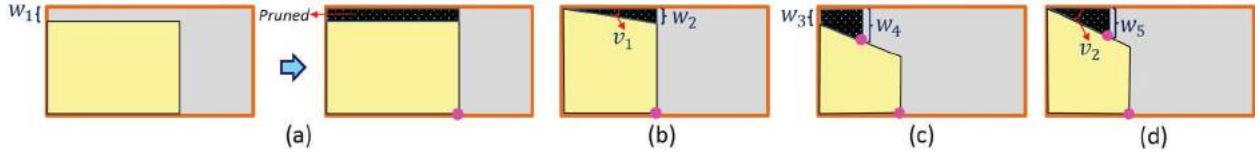

Figure 10. Pruning of bins: (a) $w_1 < l_{min}$, (b) $w_1 < \alpha_{min}$ and $w_2 < l_{min}$, (c) $w_3 < l_{min}$and $w_4 = 0.5$, (d) $v_2 < \alpha_{min}$ and $w_5 = 0.5$.

and t denotes the number of vertices in ρ where $j \in [1, t]$. Every combined item has a *main point* (see Fig. 13(a)) which is defined as the closest vertex to the left-bottom vertex of a combined item's bounding box and coincides with the Candidate point when placed on the candidate region.

A candidate point has the following characteristics:

- Let v_j^n be a vector from a vertex (P_j) in ρ to its next vertex, and let v_j^p denotes a vector from P_j to the previous vertex of P_j. If the cross product $v_j^n \times v_j^p$ is positive, the vertex is called a *convex vertex*. Otherwise, it is a *concave vertex*. A candidate point is a convex vertex. Convex and concave vertices of are depicted with black/pink and green dots, respectively, in Fig. 9(a).
- When combined items are placed on convex vertices in ρ, there is either no or a less amount of intersection area between the combined item and ρ (see the upper image of Fig. 9(a)). We do not accept such vertices (see pink points in Fig. 9(a)) as candidate points. The other remaining convex vertices are candidate points (in black).

We take a greedy approach to generate a candidate rρegion for each candidate point of the bin ρ, as shown in Fig. 9(b-f). The concave vertices of ρ are eliminated one by one. In each elimination step, the resulting polygon with a greater area is selected. Decomposition of the given polygon (ρ) is performed until a convex polygon is obtained. If the obtained convex polygon is not a triangle or a quad (e.g., a pentagon, hexagon), it has to be further decomposed into triangular or quad polygons. Section 4.3 outlines a method for this decomposition. Figure 9(f) shows three candidate regions with their corresponding candidate points (boundaries in dark green, purple, and blue) generated from the bin in Fig. 9(a), where the obtained candidate regions cover the bin completely. Note that it also possible to enumerate all candidate regions of a candidate point without pruning the smaller area branches (polygons). However, such a complete enumeration will result in increased computational time.

Figure 10 shows a bin (in gray) and some of its portion is pruned. After placing a combined item (in yellow) on the bin, bin portions that do not satisfy the minimum length and angle constraints are pruned. Figures 10 (a) and (c) show a bin containing an edge whose length (w_1 and w_2) is smaller than l_{min}, and therefore, it is pruned. If the bin contains an angle (v_1 and v_2) smaller than α_{min}, it is also pruned, as shown in Fig. 10(b) and (d). Candidate points of the bin after the pruning operation are depicted with pink dots.

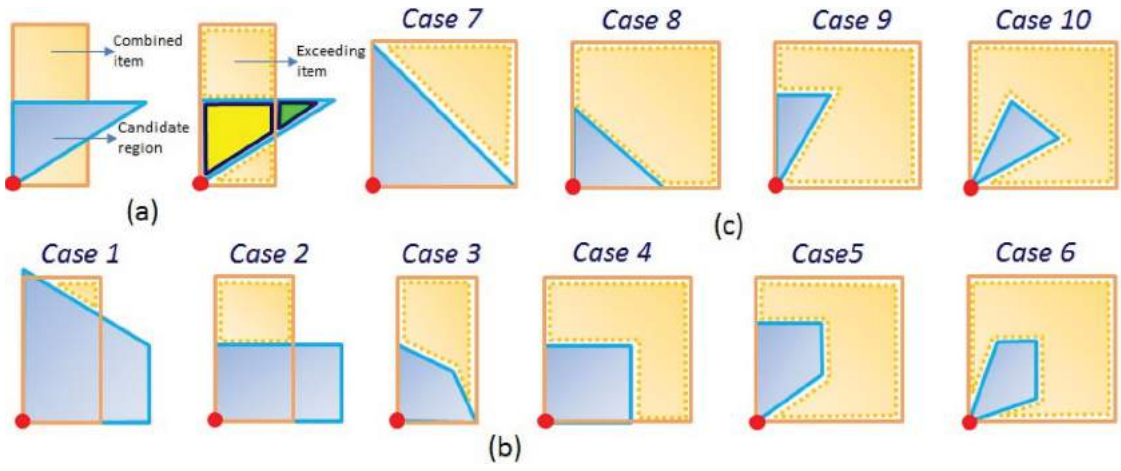

Figure 11. (a) Four separate regions are formed after placement of a combined item (in orange) on a Candidate region (in blue): packed (in yellow) and remaining portions (in orange) of the combined item, and the remaining portion of the combined item (in green). (b, c) Several polygon types (cases 1-10) are formed after subtracting the combined item from a candidate region with a quad (b) and a triangular shape (c).

4.3. Split operator

The split operator splits a combined item into sub-items if it exceeds the candidate region. Figure 11(a) shows a combined item that exceeds the candidate region after placement. The exceeded portion of the combined item is termed as *exceeding item*. An exceeding item can have a shape other than a quad or triangle. Figures 11 (b) and (c) list the possible exceeding item shapes. If the candidate region is a quad, the exceeding item can be six types of different polygons, as follows: a triangle, quad, pentagon, hexagon, heptagon or octagon (see cases 1–6 in Fig. 11(b)). If the candidate region is triangular, the exceeding item can have triangular, pentagonal, hexagonal, or heptagonal shapes (see cases 7–10 in Fig. 11(c)). If an exceeding item has a shape other than a triangle or quad, it has to be decomposed into a convex quad or triangular items before the packing process. Each decomposed item is packed separately during packing. To decompose a non-triangular or non-quad exceeding item with concave geometry into triangular or quad items, edges containing concave vertices are extended, and convex polygons containing the edges are obtained (see Fig. 12(a)). These polygons are *candidate items* (see polygons $C_1 - C_4$ in Fig. 11(a)) of the exceeding item, each of which is placed on the candidate region while checking its placement cost.

4.4. Orientation modifier

We take the orientation of the combined items as they are formed in the combined items' generation step. The *main point* of a combined item should coincide with the candidate point of a candidate region when combined items are packed. Several orientations of a combined item are produced by utilizing rotation and mirror *orientation modifiers* (OMs). Rotation OM rotates a combined item around its main point in the positive Z axis by θ. until obtaining a full rotation. In this study, θ is set to $\pi/2$ radians, and four different combined item orientations with their main points are obtained, as shown in Fig. 13(a). Note that setting θ to smaller values will increase the number of generated orientations; therefore, better packing of combined items (i.e., minimizing bin waste) can probably be obtained. Mirror OM changes the orientation of the combined items by mirroring them in the X-axis. Figure 13(b) depicts the combined items with their main points obtained after applying the mirror OM.

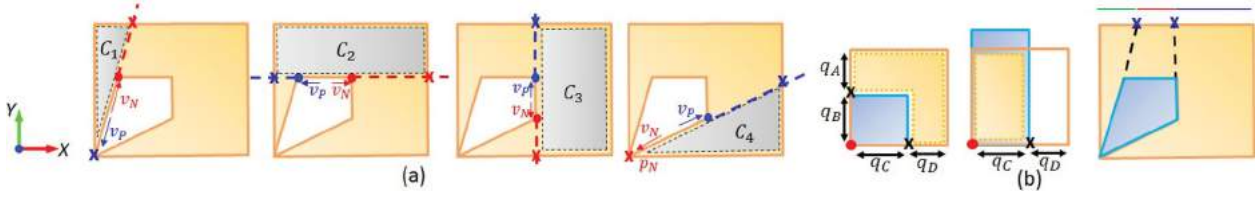

Figure 12. (a) Generation of Candidate items ($C_1 - C_4$) from a concave Exceeding item (b) Images to help understanding the Y(.) term of the cost function.

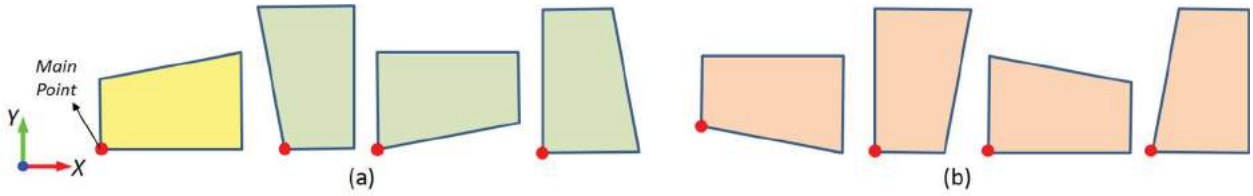

Figure 13. (a) Given a combined item (in yellow) with its main point, three more orientations of the combined item are generated by rotating it around its main point by $\pi/2$ radians. (b) Orientation of four combined items (a) are modified by mirroring them in the *X*-axis and four more orientations are obtained.

The main point of the combined item is updated for both OMs and is the closest vertex to the left-bottom vertex of the combined item's bounding box.

4.5. Placement decision maker (Pd-maker)

Placement of combined items on candidate regions are performed based on a cost function. Priority is given to the combined item and candidate region pair with minimum placement cost. The cost function $F(\iota, B, \rho) = X(\iota, B, \rho) + \beta * Y(\iota, B)$ consists of two terms and the β parameter, which adjusts the weight of the second term. The term $X(\iota, B, \rho)$ is computed based on the resulting bin ρ' after packing a combined item, ι, into a candidate region, B, of a bin ρ where $B \subset \rho$. The resulting bin $\rho' = \rho - \iota_I$ is obtained by subtracting the intersection $\iota_I = \iota \cap B$ between and B from ρ. Here, ρ' can be represented as a counter-clockwise ordered vertices and edges described by $\rho' = \langle v_1, e_1, v_2, e_2, \ldots, v_u, e_u \rangle$., where u is an integer. The last edge e_u is the edge connecting the last vertex $v_u$and the first vertex v_1. Thedge lengths for the edges in ρ' are denoted by $L' = \langle l_1, l_2, \ldots, l_u \rangle$, and the angles between the edges connected at the vertices are $\langle \alpha_1, \alpha_2, \ldots, \alpha_u \rangle$. The term $X(\iota, B, \rho)$ is computed as follows: $X(\iota, B, \rho) = \sum_{j=1}^{u} (g(l_j) + h(\alpha_j))$. If the length l_j for the j^{th} edge ($j = 1, 2, \ldots, u$) is smaller than the minimum edge length constraint l_{min}, the resulting bin is penalized by setting $g(l_j)$ to l_j/l_{min}. Otherwise, $g(l_j)$ is zero. Similarly, if the angle α_j at the j^{th} vertex is smaller than the minimum angle constraint α_{min}, the resulting bin is penalized by setting $h(\alpha_j)$ to α_j/α_{min}. Otherwise, $h(\alpha_j)$ is zero. Note that a cost value is computed for each combined item-candidate region pair so that the pair with minimum placement cost can be selected.

The left and middle images of Fig. 12(b) illustrate two combined items (in orange) placed on the same candidate region (in blue). If the distance q_A is greater than l_{min}, placement of both combined items on the candidate region will have the same value for the term $X(\iota, B, \rho)$. The term $Y(\iota, B)$ will give placement priority to one of these two combined items.o compute the value for the term $Y(\iota, B)$, decomposition positions and probable decomposition positions have to be first found (see black and blue crosses, respectively, in Fig. 12(b) and called *R points*) from where the combined item has to be decomposed or probably decomposed. These points divide edges into several pieces (see green, red, and blue lines on the top edge). Let the edge e_g be an edge containing a single R point v_g. Then, e_{g1} and e_{g2} e the edges where $e_g = e_{g1} \cup^{e_{g2}}$, and v_g is the intersected point of e_{g1} and e_{g2}. $Y(\iota, B)$ is computed as follows: $Y(\iota, B) = \sum_{e_g \in E_G} Z(e_g)$, where E_G consists of set of edges, each of which contains an R point, and $Z(e_g) = (v(e_{g1}) + v(e_{g2}) + 1) - v(e_g)$. Furthermore, $v(x)$ is the number of cuts required for the edge x based on the maximum length constraint l_{max} and is calculated as follows: $v(x) = c(L(x)/l_{max})$, where $L(x)$. is the length of edge x and $c(y)$ rounds y downward. For the sake of simplicity, we only explain computations on an edge containing a single R point. If there is more than one R point on an edge, as shown with black crosses in the right image of Fig. 12(b), computations will also be done similarly. $Y(\iota, B)$ favors the generation of combined items (occurring after placement on candidate regions), where a minimum number of decompositions are required to obtain hull items (see Section 3.1). We set β to 0.2 in this paper, which gives higher priority to the $X(\iota, B, \rho)$ term. Finally, note that if an exceeding item, whose candidate items do not satisfy the quality criteria, appears after placing a combined item on a candidate region, a higher cost is assigned to the placement of that combined item, and it will not be packed in the current step.

5. Results and discussion

Figure 14 shows the results of the proposed packing method for the combined items depicted in Fig. 6. Regions with colorful boundaries and shaded with gray are the combined items placed on bins (shown with black boundaries) with size of $12\,m \times 2.5\,m$. Areas in white between the combined items are the unused portions of

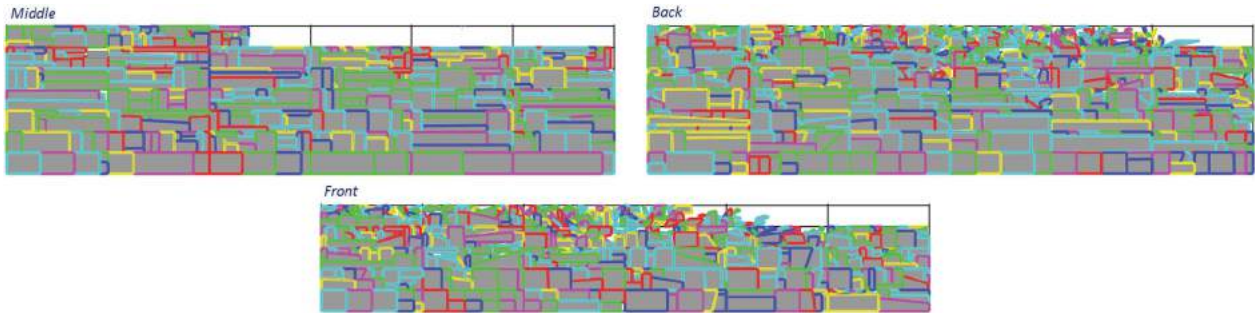

Figure 14. Packing of combined items depicted in Fig. 6 using the Split-and-Pack approach.

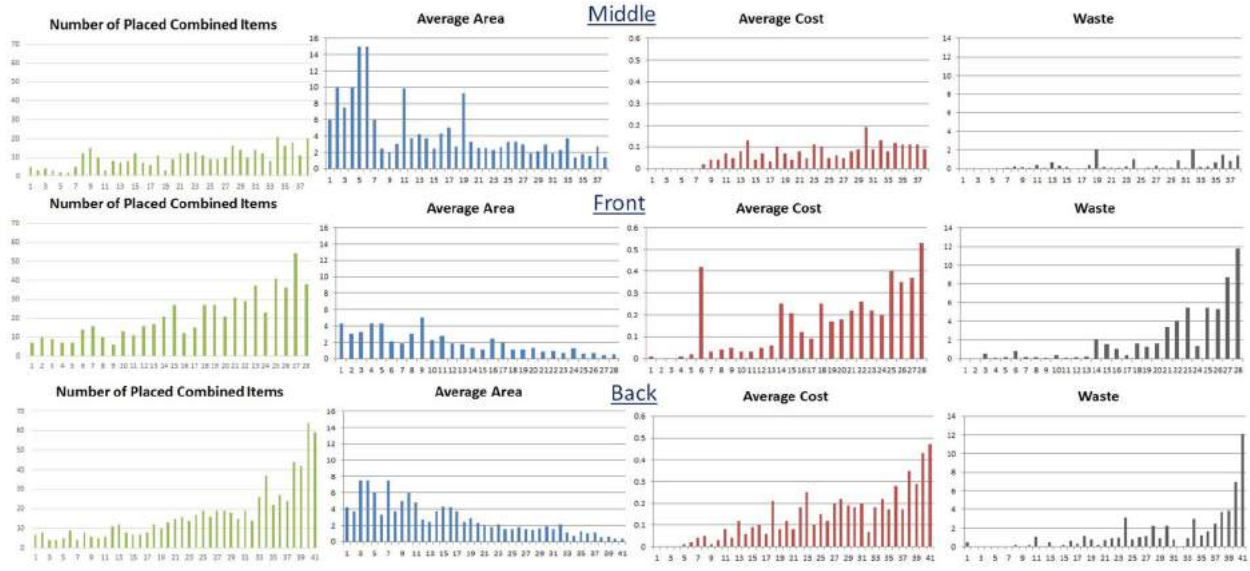

Figure 15. Detailed results of the proposed technique for the packing shown in Fig. 13.

the bins and can be called *waste*. Since combined items generated from the flattened blocks residing on the middle portions of the ship hull are close to rectangular, packed items are also rectangular, as seen in the upper image of Fig. 14. When packing results of the front and back portions for the ship hull are analyzed, triangular items can be seen, particularly in the final stages, because initial combined items (before packing) are not close to rectangular for the front and back hull portions.

Figure 15 shows the detailed results for the placeme of combined items on bins except for the placement on the final bins. Horizontal axes denote the bin numbers. Vertical axes denote the number of placed combined items, average area of placed combined items, average cost, and waste per bin. The number of placed combined items is less at the initial placement and increases at the final stages. This is due to the formation of small-size candidate items after the split process. The average area of placed combined items per bin also decreases for the same reason. At the initial stages, the cost of placement and waste of bins are close to zero, which increases as placement steps proceed. Average amounts of waste per bin are about 0.45, 2.06, and 1.38 m^2 for the middle, front, and back ship hull portions, respectively, where the area of a bin is 30 m^2. It has been observed that the waste amount of bins is large for the final stages of the front and back ship hull portions. This is because of the generated combined items in the final stages, which are not close to rectangular quads or are skinny triangles. Such combined items do not fit the candidate regions properly (i.e., while satisfying the minimum length and angle constraints). To place them on candidate regions properly, rotation OM (described in Section 4.4) should be applied with a smaller rotation angle (i.e., smaller θ values), particularly for the last placement steps.

5.1. Computational time

A PC with an Intel Core *i*7 4820 3.7 GHz processor and 8 GB memory is used for the experiments in this study, and the implementation is single-threaded. Table 2 shows the computational time taken for the proposed algorithms in this work. The processing time for the initial covering of flattened blocks was less than 1 second. The

Table 2. Processing time in seconds for the proposed algorithms of this study.

Hull Portions	Initial Covering of Flattened Blocks	Generation of Combined Items	Packing of Combined Items
Middle	0.172	8.088	15.03
Front	0.47	10.731	87.636
Back	0.126	3.915	99.198

generation of combined items took less than 11 seconds for all hull portions, which depended on the maximum length constraint (i.e., l_{max}). For smaller values of l_{max}, many bins were generated, and therefore many merging operations for neighboring bin templates can be listed. The computational time for the Split-and-Pack approach was about 15 seconds for the middle hull portion where the generated combined items were closer to be a rectangular, whereas the times taken for the ship hull front and back portions were about 88 and 99 seconds, respectively. Since the combined items of these hull portions are generally not rectangular, more candidate items could be generated from the combined items than in the middle hull portion. The overall processing time for all hull portions was less than 2 minutes, and it can be said that the proposed approach of this study has low computational cost.

5.2. Parameter tuning

Figure 16 shows the bin waste amount and number of placed combined items of the middle ship hull portions for different β (adjusting the weight of the $Y(\cdot)$ term in the cost function) settings. According to these experiments, bin waste was 5.48 when $\beta = 0.0$, 17.4 when $\beta = 0.2$, and 21.19 when $\beta = 0.4$. A lower amount of bin waste is obtained for the $\beta = 0$ setting, which assigns the $X(\cdot)$ term full weight. Recall that the $X(\cdot)$ term favors the generation of bin shapes satisfying the quality criteria (i.e., minimum length constraint), and therefore, the waste amount of bins is minimized. It was also observed that the number of placed combined items decreased when β increased, as follows: 393 for $\beta = 0.0$, 371 for $\beta = 0.2$ and 317 for $\beta = 0.4$. When a smaller weight was given to the $Y(\cdot)$ term, the Split-and-Pack algorithm minimized the bin waste with less consideration of the number of cuts on the combined items resulting in the generation of more combined items.

5.3. Results using different combined item sets

Different combined item sets were generated by setting the allowable waste ξ during the generation of combined items. Note that setting ξ to greater values allows the generation of a smaller number of combined items with greater size and waste. Table 3 shows the waste amount at the end of the combined items' generation and packing steps. It was observed that although setting ξ to 0.5 generated combined items with greater waste than that of the $\xi = 0.1$ setting for the ship hull back portion, smaller waste occurred after the combined items' packing. Thus, there was less overall waste for the $\xi = 0.5$ setting than when ξ was 0.1. Since large combined items were produced in the $\xi = 0.5$ setting, they had much more splitting flexibility (i.e., they could be split many times) than smaller items; thus, less waste occurred compared to that of the $\xi = 0.1$ setting. We observed a similar overall waste amount for the ship hull front portion.

5.4. Results when using items and combined items

Let us take 18 items of $2 \times 2\,m$ width and length, 30items of $1 \times 1.6\ m$, 15items of $2.5 \times 3\ m$ and 21items of $1.2 \times 1.5\,m$. These items are combined and the following combined items are obtained: 6combined items of $2 \times 6\ m$, 10combined items of $1.6 \times 3\ m$, 5combined items of $2.5 \times 9\ m$ and 7combined items of $1.5 \times 3.6\,m$. Experiments were conducted to compare the results when using the non combined item set, like that of the

Table 3. W_{BC}, W_{BS}, and W_B denote waste during the combined items' generation step, the packing step and the total waste for the ship hull back portion, respectively, for the user-defined allowable waste ξ. W_{FC}, W_{FS}, and W_F denote waste for the ship hull front portion during these steps.

ξ	W_{BC}	W_{BS}	W_B	W_{FC}	W_{FS}	W_F
0.1	3.9	75.7	79.6	3.3	60.0	60.3
0.5	11.2	56.8	68	6.0	57.8	63.8

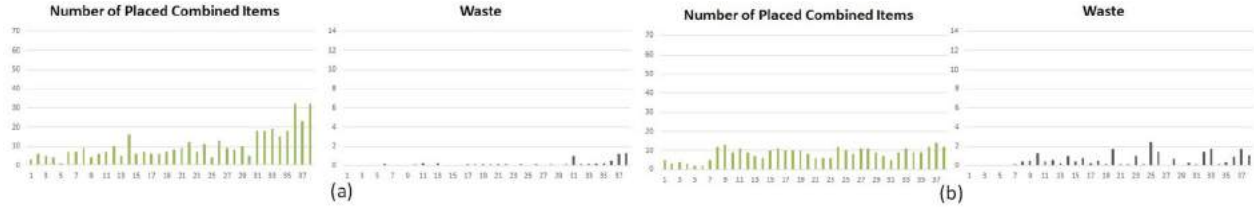

Figure 16. Waste and number of placed combined items for the middle ship hull portions when (a) $\beta = 0.0$ and (b) $\beta = 0.4$.

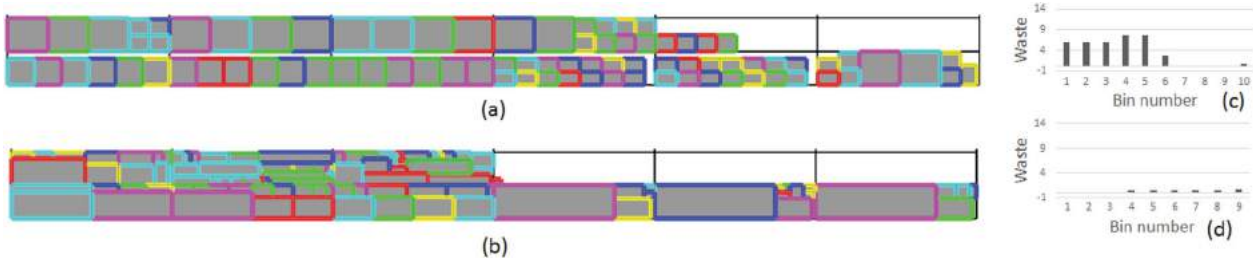

Figure 17. Results obtained using non-combined items without applying the split concept (a, c) and combined items with the split concept applied (b, d).

2Dbp approaches and the combined item set. Split concept in Section 4.3 is applied for the combined item set, whereas it is not applied for the non-combined item set. Figures 17(a) and (b) show the packing results. The amount of bin waste for the non combined item set packing was observed to be 36.9 m^2 while it was 3.23 m^2 for packing the combined item set. Thus, utilizing the combined item set with the split concept can reduce bin waste.

5.5. Post processing of combined items

Since packed combined items can have edge lengths greater than the maximum length constraint (i.e., l_{max}), they have to be further decomposed into hull items satisfying the quality criteria in Section 3.1. Horizontal and vertical cuts, introduced in Section 3.2.1, can split these combined items into hull items.

6. Conclusions and future studies

This paper proposes a new set of algorithms to pack a large *3D* structure into bins while minimizing bin waste. The main approach involved initial covering of the flattened block using bin templates, generating combined items by joining neighboring items of the block, and packing these combined items in bins by splitting them if required. The experiments showed that the proposed approach is effective in terms of minimizing bin waste, as combined items can be split from wherever required without producing items that violate the defined quality criteria. The proposed algorithm can be enhanced in several ways. For example, guillotine cuts [2, 17] (i.e., a set of edge-to-edge cuts parallel to the edges of bins) can also be used during packing. Moreover, utilizing a different set of combined items could further minimize bin waste. Therefore, mesh segmentation methods can be integrated into the proposed approach to generate a better set of combined items enabling minimal waste. Quad meshing [6] and quad partitioning [14–16] techniques can be utilized to obtain a better set of flattened blocks from which combined items are generated.

Acknowledgements

The author would like to thank Hiromasa Suzuki, Masaki Moriguchi and Hikmet Kocabas for valuable discussions. This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK BIDEB – Project Number: 114C067).

Funding

This work was supported by The Scientific and Technological Research Council of Turkey [TUBITAK BIDEB – Project Number: 114C067].

ORCID

Erkan Gunpinar http://orcid.org/0000-0002-0266-5546

References

[1] Bansal, N.; Khan, A.: Improved Approximation Algorithm for Two-Dimensional Bin Packing, Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 2014.

[2] Bansal, N.; Lodi, A.; Sviridenko, M.: A tale of two dimensional bin packing, *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, 2002, 657–666.

[3] Beasley, J.: An exact two-dimensional non-guillotine cutting tree search procedure, *Oper. Res.*, 33, 1985, 49–64. https://doi.org/10.1287/opre.33.1.49

[4] Berkey, J.; Wong, P.: Two dimensional finite bin packing algorithms, *Journal of Operational Research Society*, (2), 1987, 423–429.

[5] Beyaz, M.; Dokeroglu, T.; Cosar, A.: Hybrid Heuristic Algorithms for the Multiobjective Load Balancing of 2D Bin Packing Problems, *Information Sciences and Systems Conference*, 2015, 209–220. https://doi.org/10.1007/978-3-319-22635-4_19

[6] Bommes, D.; Levy, B.: Pietroni, N.; Puppo, E.; Silva, C.; Tarini, M.; Zorin, D.; State of the art in quad meshing, Eurographics STARS, 2012.

[7] Camachoa, E. L.; Marina, H. T.; Rossb, P., Ochoa, G.: A unified hyper-heuristic framework for solving bin packing problems, *Expert Systems with Applications*, 41(15), 2014, 6876–6889. http://doi.org/10.1016/j.eswa.2014.04.043

[8] Chung, F.-R.-K.; Garey, M.-R.: Johnson, *D.-S.; On packing two-dimensional bins, Journal on Algebraic and Discrete Methods*, 3(1), 1981, 66–76. https://doi.org/10.1137/0603007

[9] Csirik, J.; Woeginger, G.: On-line packing and covering problems, in: Lecture notes in computer science, *Springer*, 144, 1998, 147177. https://doi.org/10.1007/bfb0029568

[10] Dash, S.; Kalagnanam, J.-R.; Reddy, C.-K.: Method for production design and operations scheduling for plate design in the steel industry, US Patent US20060100727 (B2), 2006.

[11] Frenk, J.-B.-G.; Galambos, G.: Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem, *Computing*, 39(3), 1987, 201–217. https://doi.org/10.1007/BF02309555

[12] Gilmore, P.-C.; Gomory, R.-E.: Multistage cutting problems of two and more dimensions, *Oper. Res.*, 13, 1965, 94–119. https://doi.org/10.1287/opre.13.1.94

[13] Griffith, S.: Decomposition of 3d geometry into developable surface patches and 2d cut patterns, US Patent 20130297058 (A1).

[14] Gunpinar, E.; Suzuki, H.; Ohtake, Y.; Moriguchi, M.: Generation of bi-monotone patches from quadrilateral mesh for reverse engineering, *Computer-Aided Design*, 45(2), 2013, 440–450. https://doi.org/10.1016/j.cad.2012.10.027

[15] Gunpinar, E.; Moriguchi, M.; Suzuki, H.; Ohtake, Y.: Feature-aware partitions from motorcycle graph, *Computer-Aided Design*, 47, 2014, 85–95. https://doi.org/10.1016/j.cad.2013.09.003

[16] Gunpinar, E.; Moriguchi, M.; Suzuki, H.; Ohtake, Y.: Motorcycle graph enumeration from quadrilateral meshes for reverse engineering, *Computer-Aided Design*, 55, 2014, 64–80. https://doi.org/10.1016/j.cad.2014.05.007

[17] Hadjiconstantinou, E.; Christofides, N.: 2-D cutting problems using guillotine cuts, *Eur. J. Oper. Res.*, 83, 1995, 21–38. https://doi.org/10.1016/0377-2217(93)E0277-5

[18] Hana, W.; Bennell, J.-A.; Zhao, X.; Song, X.: Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints, *European Journal of Operational Research*, 230(3), 2013, 495–504. https://doi.org/10.1016/j.ejor.2013.04.048

[19] Honga, S.; Zhanga, D.; Laub, H. C.; Zenga, X.Sic, Y. W.: A hybrid heuristic algorithm for the 2D variable-sized bin packing problem, *European Journal of Operational Research*, 238(1), 2014, 95–103. http://dx.doi.org/10.1016/j.ejor.2014.03.049

[20] Lai, C.; Cui, Y.; Yao, Y.: A constructive heuristic for the two-dimensional bin packing based on value correction, *International Journal of Computer Applications in Technology*, 55(1), 2017. http://doi.org/10.1504/IJCAT.2017.082263

[21] Lodi, A.; Martello, S.; Vigo, D.: Recent advances on two-dimensional bin packing problems, *Discrete Applied Mathematics*, 123(13), 2002, 379–396. https://doi.org/10.1016/S0166-218X(01)00347-X

[22] Lowa, K. L.; Ilie, A.: Computing a view frustum to maximize an object's image area, *Journal of Graphics Tools*, 8(1), 2003, 3–15.

[23] Martinez Sykora, A.; Alvarez-Valdes, R.; Bennell, J.; Ruiz, R.; Tamarit, J. M.: Matheuristics for the irregular bin packing problem with free rotations, *European Journal of Operational Research*, 258(2), 2016, 440–455. http://dx.doi.org/10.1016/j.ejor.2016.09.043

[24] Technology, T.-N.: MeshFlatten standalone application 1.0, Nanjing, Jiangsu, China (http://www.resurf3d.com) (2007).

[25] Trivella, A.; Pisinger, D.: The load-balanced multi-dimensional bin-packing problem, *Computers & Operations Research*, 74, 2016, 152–164. http://doi.org/10.1016/j.cor.2016.04.020

[26] Zhao, X.; Shen, H.: Online algorithms for 2D bin packing with advice, *Neurocomputing*, 189, 2016, 25–32. http://doi.org/10.1016/j.neucom.2015.11.035

[27] Zheng, D.: Study on the layout optimization of platform based on simulated annealing algorithm, *Journal of Software*, 8(7), 2013, 1793–1800. https://doi.org/10.4304/jsw.8.7.1793-1800