Collision free region determination by modified polygonal Boolean operations

Computer-Aided Design 45 (2013) 1029–1041

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Collision free region determination by modified polygonal Boolean operations

André Kubagawa Sato, Thiago Castro Martins, Marcos Sales Guerra Tsuzuki [*]

Escola Politécnica da Universidade de São Paulo, Computational Geometry Laboratory, Department of Mechatronics and Mechanical Systems Engineering, Av. Prof. Mello Moraes, 2231 São Paulo - SP, Brazil

HIGHLIGHTS

- An algorithm to determine the collision free region is proposed.
- The collision free region is a useful tool for cutting and packing problems with irregular items.
- Degenerated elements (edges and vertexes) represent local compaction situations.
- The collision free regions determines the presence of local compaction for the current item.

ARTICLE INFO

Article history:
Received 6 May 2011
Accepted 25 March 2013

Keywords:
Cutting and packing problems
Boolean operations

ABSTRACT

Cutting and packing problems are found in numerous industries such as garment, wood and shipbuilding. The collision free region concept is presented, as it represents all the translations possible for an item to be inserted into a container with already placed items. The often adopted nofit polygon concept and its analogous concept inner fit polygon are used to determine the collision free region. Boolean operations involving nofit polygons and inner fit polygons are used to determine the collision free region. New robust non-regularized Boolean operations algorithm is proposed to determine the collision free region. The algorithm is capable of dealing with degenerated boundaries. This capability is important because degenerated boundaries often represent local optimal placements. A parallelized version of the algorithm is also proposed and tests are performed in order to determine the execution times of both the serial and parallel versions of the algorithm.

1. Introduction

Cutting and packing problems involving irregular shapes arise in a wide variety of industries, including shipbuilding, garments, sheet metal cutting, plastics and shoe manufacturing. These problems usually consist in placing a number of irregular items into one or more containers in such way that the layout is the most efficient possible; all items are assigned and do not overlap. The two-dimensional stock cutting problem was shown to be NP-hard and is therefore intrinsically difficult to solve.

Bennell and Oliveira [1] showed that the manipulation of items and containers' geometry is a key point to determine whether a layout is feasible or not. Several approaches to ensure that, in the resulting layout, items do not overlap and fully fit inside the container have been proposed in the literature. Adamowicz and Albano [2] chose to nest items into simpler shapes in which the interference can be more easily calculated. Babu and Babu [3] approximated the container and the items by grid squares represented by a matrix. Lee et al. [4] used direct trigonometry to determine the interference among items and the container. Recently, the nofit polygon (NFP) has been used by several researchers [5–9] to ensure feasible layouts.

The NFP is the set of feasible locations for one polygon with respect to another polygon, such that the polygons do not overlap. Feasible locations are required for most of the solutions to two-dimensional packing problems, and also for other problems such as robot motion planning. Different approaches to generate the NFP have been proposed in the literature. Minkowski sums were proposed by Ghosh [10], and were later applied to cutting and packing problems by Dean et al. [11] and Bennell and Song [12]. Agarwal et al. [13] compared different algorithms of convex subpolygon decomposition, such that the Minkowski sum can be directly determined for each convex subpolygon pair. Li and Milenkovic [14] decomposed the items into star-shaped polygons. Burke et al. [15] proposed an orbiting algorithm in which the movable item slides along the fixed item.

When sequential placement of items is adopted, the placement heuristic must take into account previously placed items, as well as the container in order to obtain a feasible layout. The concept of collision free region emerges from this heuristic, and it represents

[*] Corresponding author. Tel.: +55 11 3091 5759.
E-mail address: mtsuzuki@usp.br (M.S.G. Tsuzuki).

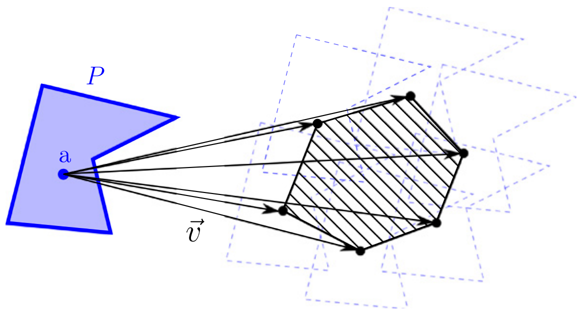

Fig. 1. Set of translations of a shape P represented as a region containing all possible translations from its reference point a.

the set of translations that, when applied to the movable item, places it in the interior of the container without colliding with the already placed items. The collision free region is determined by unifying the NFPs determined by the movable item in respect to the already placed items.

The NFPs union must be made through non regularized Boolean operations, in which degenerated edges and vertexes are considered. A degenerated edge represents a sliding fit (in which the item position is constrained in all but one direction) and a degenerated vertex represents an exact fit (in which the item position is fully constrained by its surroundings).

Gomes and Oliveira [16] evaluated the point intersections between NFPs and selected the intersection points that are not internal to any NFP and, simultaneously, internal to the inner fit polygon. Such an approach cannot classify the vertexes in conventional boundary, sliding fit and exact fit. The vertex classification can be made by analyzing the vertex neighborhood with appropriate rules. Martins and Tsuzuki [9] used regularized Boolean operations to determine the collision free region, thus ignoring exact and sliding fit configurations.

Here, an algorithm to determine the collision free region is proposed. The paper is structured as follows. The concepts of NFP and collision free region are defined in Section 2. Section 3 explains the importance of determining the collision free region degenerated elements, using specific non-regularized Boolean operations. Section 4 shows the proposed algorithm. The proposed algorithm was implemented in two versions: serial and parallel. Finally, computational results are presented and conclusions are drawn.

2. Nofit polygon and collision free region

In this section, the collision free region and its primitive components, the nofit and inner fit polygons, are defined.

2.1. The nofit polygon

The NFP represents a set of translations of an item and is mathematically represented by a set of vectors. For a better understanding of the NFPs properties, the set of translations of an item are represented by polygons in the plane. Every item has a reference point that can be internal or external to it. The NFP represents the set of forbidden translations that, when applied to the item, moves the reference point to the interior of the NFP, as shown in Fig. 1. For an item P, which is a closed data set, let $i(P)$ be its interior, ∂P be its boundary and $c(P)$ be its complement.

Definition 2.1. The NFP induced by item P_i to item P_j, noted as $\Upsilon(P_i, P_j)$, is the set of translation vectors that, when applied to P_j, makes it collide with P_i. Thus,

$$\Upsilon(P_i, P_j) = i(P_i) \ominus i(P_j) = \left\{\vec{\boldsymbol{v}} \mid \exists \boldsymbol{a} \in i(P_j), \boldsymbol{a} + \vec{\boldsymbol{v}} \in i(P_i)\right\}. \quad (1)$$

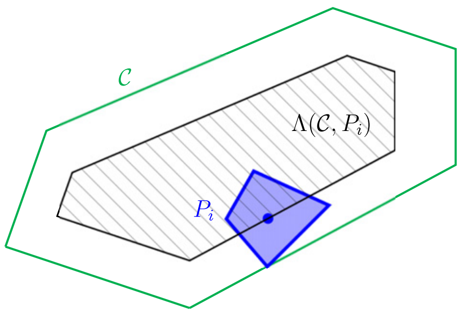

Fig. 2. The inner fit polygon for a given item and container.

Another way to define the NFP is [8]

$$\Upsilon(P_i, P_j) = i(P_i) \oplus (-i(P_j)) = \left\{(\mathbf{v} - \mathbf{w}) \mid \mathbf{v} \in i(P_j), \mathbf{w} \in i(P_i)\right\}. \quad (2)$$

The NFP can be obtained by the Minkowski sum algorithm [13], which can be calculated very efficiently for convex polygons. The Minkowski sum result of two convex polygons is a convex polygon built from the original polygon edges sorted in counterclockwise order. Non-convex polygons can be decomposed into convex polygons in a preprocessing step, as the transformations applied (rotations and translations) do not affect such decomposition.

Definition 2.2. The Minkowski sum of two polygons P_i and P_j, noted $P_i \oplus P_j$, is defined as the set of points $\{\boldsymbol{O} + \vec{\boldsymbol{v}} + \vec{\boldsymbol{w}} \mid \boldsymbol{O} + \vec{\boldsymbol{v}} \in P_i, \boldsymbol{O} + \vec{\boldsymbol{w}} \in P_j\}$.

Definition 2.3. The opposed polygon for a given polygon P_j, noted as $-P_j$, is defined as the set of points $-P_j = \left\{\boldsymbol{O} - \vec{\boldsymbol{w}} \mid \boldsymbol{O} + \vec{\boldsymbol{w}} \in P_j\right\}$.

The opposed polygon is obtained by inverting the signal of all the coordinates of the original polygon. From the above definitions, one can see that

$$i(P_i) \ominus i(P_j) = i(P_i) \oplus \left(-i(P_j)\right) \quad (3)$$

meaning that the NFP is produced by the Minkowski sum of the fixed item with the opposed item to be placed.

An important property is that $i(\Upsilon(P_i, P_j))$ represents colliding placements. $\partial(\Upsilon(P_i, P_j))$ and $c(\Upsilon(P_i, P_j))$ represent feasible placements.

2.2. The inner fit polygon

The inner fit polygon is another important frequently used concept, which is derived from the NFP and represents a set of translations for the placement of items inside a container $\mathcal{C}$. The inner fit polygon can be computed by sliding an item along the internal contour of the container [5] (see Fig. 2).

Definition 2.4. The inner fit polygon induced by container $\mathcal{C}$ to item P_j, noted as $\Lambda(\mathcal{C}, P_j)$, is the set of translation vectors applied to P_j that leaves it inside the container. Thus,

$$\Lambda(\mathcal{C}, P_i) = c(c(\mathcal{C}) \oplus (-i(P_i))) = \left\{\vec{\boldsymbol{v}} \mid \forall \boldsymbol{a} \in i(P_i), \boldsymbol{a} + \vec{\boldsymbol{v}} \in \mathcal{C}\right\}. \quad (4)$$

An important property is that $c(\Lambda(\mathcal{C}, P_i))$ represents invalid placements. $\partial(\Lambda(\mathcal{C}, P_i))$ and $i(\Lambda(\mathcal{C}, P_i))$ represent feasible placements.

2.3. The collision free region

Consider a container $\mathcal{C}$ and a set of already placed items $\mathcal{P} = \{P_1, \ldots, P_n\}$, as shown in Fig. 3. A new item P_{n+1}, will be placed inside the container without colliding with the already placed items. The feasible set of translations for item P_{n+1} is given by the collision free region. A similar concept was previously used in robot motion planning [17, sec. 13.4], and it was originally applied to irregular packing by Martins and Tsuzuki [18].

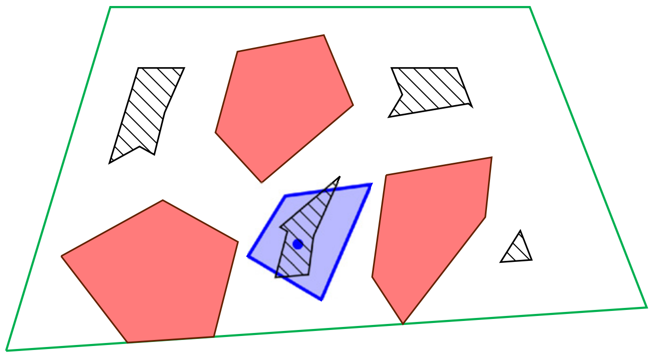

Fig. 3. The collision free region is filled with a hatch pattern. The item to be placed is filled in blue with the reference point inside, and the already placed items are filled with a different color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Definition 2.5. Collision free region is the set of all translations that, when applied to a specific item, places the specific item inside a container without colliding with the already placed items.

When the container is empty, the collision free region represents all the translations that place the item completely inside the container. In this particular case, the collision free region is the already defined inner fit polygon [5]. For any given item, the calculation of the inner fit polygon is the first step in the determination of the collision free region.

The collision free region for a specific item is determined by removing the NFPs generated by the already placed items, from the inner fit polygon.

$$\Pi(\mathcal{C}, \mathcal{P}, P_m) = \Lambda(\mathcal{C}, P_m) \boxminus \biguplus_{P_i \in \mathcal{P}} i(P_i) \ominus i(P_m) \quad (5)$$

where $\boxminus$ and $\biguplus$ are specific Boolean operations to manipulate NFPs and inner fit polygons. It is worth noting that only the NFP interior must be removed from the inner fit polygon (interior and boundary). This is why specific non-regularized Boolean operations are necessary.

By analyzing expression (5), it is possible to define at least two possible algorithms to compute the collision free region. In the first algorithm, all the NFPs are removed from the inner fit polygon and, therefore, only difference operators are used. The difference operations considered result in collision free regions. In the second algorithm, the unions of all NFPs are calculated and then removed from the inner fit polygon. The unions of NFPs result in NFPs and, after the final difference operator is applied, the collision free region is obtained. The implementation of the specific non-regularized Boolean operations is not an easy task.

3. Basic concepts

In Section 3.1, it is explained that, in this work, to implement robust Boolean operations, the intersections between segments are calculated with finite precision. The difference between regularized and non-regularized Boolean operations is explained in Section 3.2. Section 3.3 shows an example in which the regularized union misses some degenerated edges. Section 3.4 shows some necessary characteristics of the non-regularized union and difference Boolean operators. Section 3.5 explains the data structure used.

3.1. Segment intersection with finite precision

Boolean operations over polygons have the problem of lacking robustness. They face numerical instability and theoretical difficulties during geometric computations. These difficulties occur in boundary evaluations involving ill-conditioned geometric intersections [19]. There is a great amount of research on robust geometrical representations and computations. In the context of floating point arithmetic, a threshold $\epsilon > 0$ is used to compare two numbers. Hoffmann [19] presented the incidence asymmetry problem in which a vertex can be incident to another vertex but not vice versa, and the incidence intransitivity problem, which considers three vertexes, $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, where $\mathbf{a} = \mathbf{b}$ since $|\mathbf{a} - \mathbf{b}| < \epsilon$, $\mathbf{b} = \mathbf{c}$ since $|\mathbf{b} - \mathbf{c}| < \epsilon$, but $\mathbf{a} \neq \mathbf{c}$, since $|\mathbf{a} - \mathbf{c}| > \epsilon$.

Bentley and Ottmann [20] used finite precision to achieve robust algorithms for intersecting line segments. Agarwal et al. [13] used CGAL to implement the Minkowski sum algorithm with exact rational numbers, and they reported execution times that range from a few seconds for shapes involving a small amount of concavities, and up to twenty minutes for highly irregular shapes. Hu et al. [21] used interval arithmetics to ensure robustness. Wallner et al. [22] showed that interval arithmetic is not geometric in the sense that it does not give exact error bounds. Several researchers used finite precision to implement Boolean operations over polygons [23,24], which was adopted in this work, too.

3.2. Regularized Boolean operations

Conventional polygons are expected not to contain isolated points or lines. The regularization of a point set A, $r(A)$, is defined by $r(A) = \partial(i(A))$. Sets that satisfy $r(A) = A$ are said to be regular [25]. Some combinations of polygons do not quite satisfy the regularity concept. Consider, for instance, the case shown in Fig. 4. According to the ordinary definition of intersection, the intersection between the two polygons consists of a rectangular polygon plus a degenerated edge. The Boolean operation over conventional polygons needs to preserve the regularity property. The regularized intersection is defined as $A \bigcap^* B = \partial(i(A \bigcap B))$, where $\bigcap$ denotes the ordinary set operation. In the literature, several proposals to implement regularized Boolean operations were proposed [23,24,26–28].

3.3. Nofit polygon and collision free region boundaries

The collision free region cannot be determined using regularized Boolean operations because it will miss eventual degenerated elements that represent local minima for the packing problem. Fig. 5(a) shows a critical example in which four rectangular items are already placed and a fifth rectangular item is to be placed. The reference is the central point of the movable rectangular item. This example shows the difference between regularized and non regularized unions. Fig. 5(b) shows the union of the four NFPs, represented by a rectangle with two internal degenerated edges. In this case, the degenerated situation refers to a situation in which the item can slide within a segment. If regularized union is applied to the four NFPs, the two degenerated edges are lost (see Fig. 5(c)).

3.4. Collision free region and NFP determination

This section explains how to modify regularized Boolean operations to correctly implement the specific non-regularized Boolean operations which can manipulate NFPs and collision free regions. Implementations of regularized Boolean operations over conventional polygons have the following steps [23,24,26–28]: intersection determination, classification of boundaries and collection of appropriate boundaries to compose the result.

Fig. 6 shows two conventional polygons A and B with their boundary orientations. The intersections between the conventional polygons are determined and the edges are divided. The intersection determination is a common module in all types of

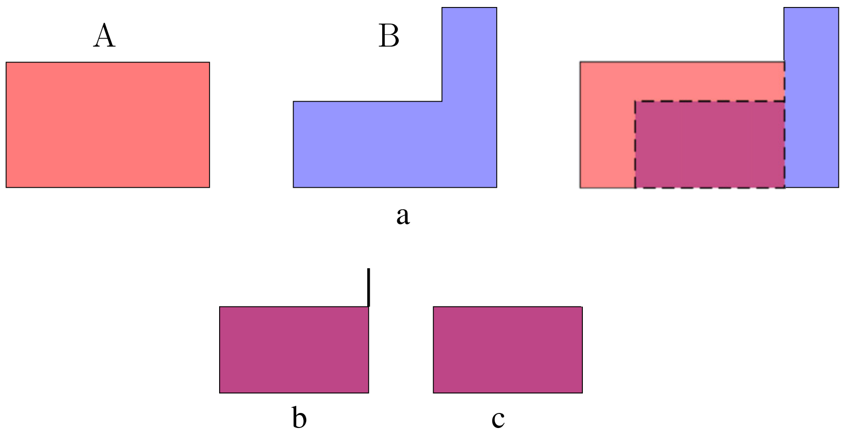

Fig. 4. (a) Polygons *A*, *B* and $A \bigcap B$ are shown. (b) Non regularized intersection with a degenerated edge. (c) Regularized intersection with no degenerated element.

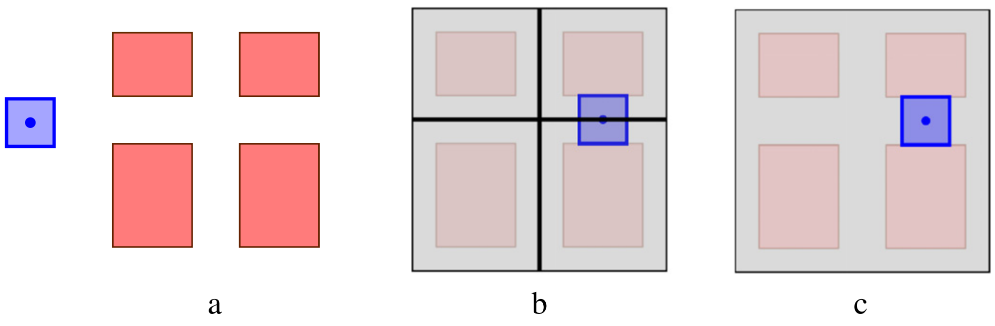

Fig. 5. Critical placement example in which the regularized union of NFPs does not result in a correct NFP. (a) On the right, four rectangular items are already placed and the movable item is shown on the left. (b) NFPs for each fixed item. The union of the four NFPs is a rectangle with two internal degenerated edges. The movable item can be placed between the fixed items as its reference point lays on a degenerated edge. (c) When the regularized union is used to combine the NFPs the degenerated edges are missed as a consequence of the regularization.

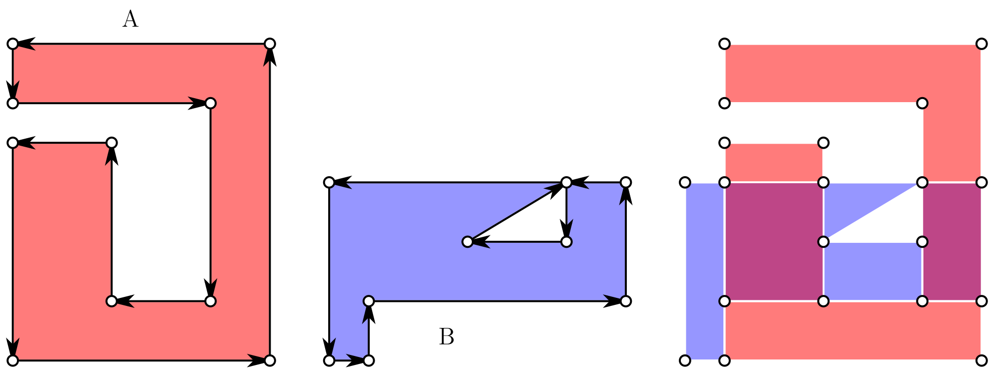

Fig. 6. Two conventional polygons *A* and *B* are shown, they have one coincident vertex and three partially coincident edges. The intersections are determined and the edges are divided.

Boolean operation implementations [23,24,26–28]. New vertexes are created at the intersecting positions and the intersecting edges are divided in both conventional polygons. Subsequently, vertexes and edges must be classified.

The vertexes of a conventional polygon are classified as internal, external and on boundary. The classification of edges must consider the case in which two polygons share an edge. The shared edge may be in opposed orientations on the original polygons or coincident orientations. Consequently, the edges of a conventional polygon can be classified according to four attributes: internal, external, coincident shared and opposite shared. Fig. 7 shows the classifications of the all edges for the example described in Fig. 6.

The collection of the appropriate edges occurs differently depending on the Boolean operation type: subtraction or union. This module is responsible for determining which edges from the original conventional polygons will be used and which will be discarded. The edges can be used to define a conventional boundary or to define a degenerated element. The rules to define a conventional boundary are the same as for regularized Boolean operations. New rules are defined to create degenerated elements from conventional polygons.

The union is exclusively used to combine NFPs, and the subtraction is exclusively used to combine a collision free region with NFP. Fig. 8 shows both cases; on the left, the result of the union of two NFPs and, on the right, the difference between a collision free region and an NFP. The edges classified as shared play a very important role, creating degenerated placements in both results. In the union case, opposed shared edges generate internal degenerated

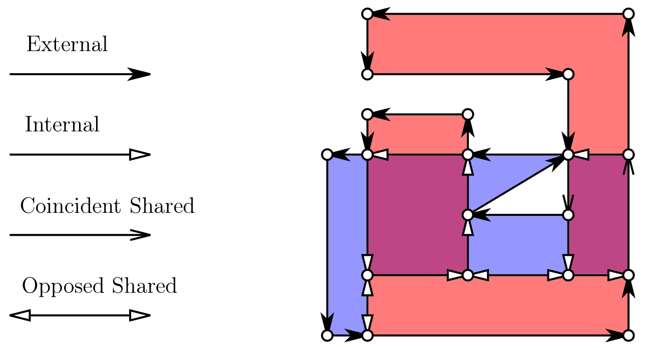

Fig. 7. Example in which the edges from conventional polygons A and B are classified according to the four attributes: external, internal, coincident shared and opposite shared.

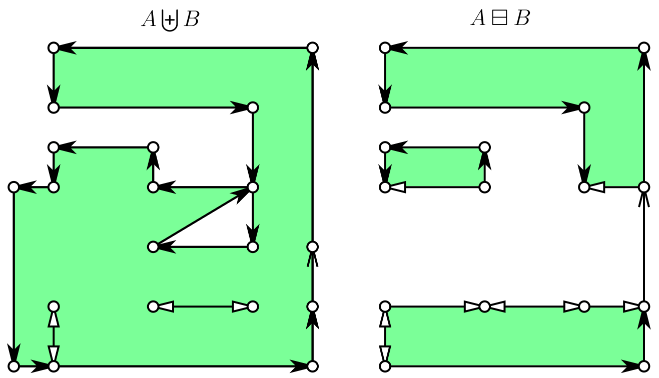

Fig. 8. On the left, the result of $A \uplus B$, where A and B are NFPs, and internal degenerated edges are present. On the right, the result of $A \boxminus B$, where A is a collision free region and B is a NFP, and an external degenerated edge is present.

edges. In the subtraction case, coincident shared edges generate external degenerated edges.

3.5. Data structure

The NFP (represented in Fig. 8 by $A \uplus B$) and the collision free region (represented in Fig. 8 by $A \boxminus B$) have two types of boundaries: conventional polygons, and degenerated edges and/or vertexes. Conventional polygons are represented by an oriented sequence of vertexes. In this work, the convention that the left side of the oriented edge is inside and the right is outside is adopted. Degenerated edges that are associated with sliding fits are represented by two vertexes without orientation. Degenerated vertexes that are associated with exact fits are represented by single vertexes. Degenerated boundaries are of special interest as they represent desirable placements that can produce local optima layouts. Degenerated elements are stored in a separated data structure. Thus, the placement solver is capable of accessing these elements whenever necessary.

4. Proposed algorithm

As previously explained, NFPs and collision free region are generically represented by conventional polygons and degenerated elements. The implementation of the specific non-regularized Boolean operations over NFPs and collision free regions is executed in two steps (see Fig. 9). The first step is based on regularized Boolean operations [23,24,26–28]. Initially, the intersection between conventional and degenerated boundaries are determined. Afterwards, conventional and degenerated boundaries are classified. Selected elements from the classified conventional

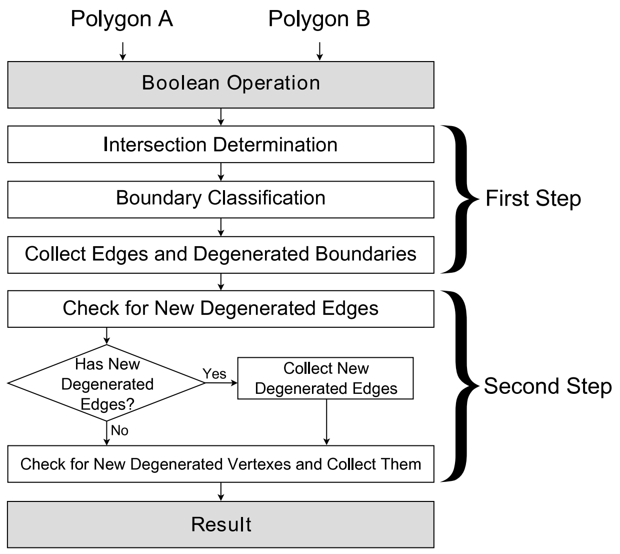
Polygon A
Polygon B
Boolean Operation
Intersection Determination
Boundary Classification
Collect Edges and Degenerated Boundaries
First Step
Check for New Degenerated Edges
Has New Degenerated Edges?
Yes
No
Collect New Degenerated Edges
Check for New Degenerated Vertexes and Collect Them
Second Step
Result

Fig. 9. System developed to determine the union of NFPs and the difference between inner fit polygons and NFPs. The intersection and classification among conventional polygons and degenerated boundaries are simultaneously determined. The collection of boundaries occurs in appropriate modules.

boundary are selected to compose the result conventional boundary; and, selected elements from the classified degenerated boundary are selected to compose the result degenerated boundary. In the second step, new degenerated elements are created. New degenerated edges can be originated from the conventional boundary classification (see Fig. 8), and new degenerated vertexes can be originated from special configurations. The main modules of the proposed algorithm are explained as follows.

4.1. Intersection determination

As conventional polygons and degenerated elements might intersect each other, all boundaries are simultaneously processed and their intersections are determined. The intersection determination is based on the Bentley and Ottmann sweep line algorithm [20]. In this work, the sweep line algorithm is modified to simultaneously classify degenerated vertexes and isolated degenerated edges; i.e., determine if degenerated vertexes and edges are internal, external or boundary.

In the sweep line algorithm, an imaginary vertical sweep line moves from left to right across edges and vertexes, halting at event points. As the sweep line proceeds, the intersections restricted to the left of the sweep line are determined. There are four kinds of event points: left end points, right end points, crossings and isolated vertexes. The edges and vertexes that intersect the sweep line s_1 are stored in a list S, which is ordered from bottom to top (see Fig. 10). When a left end point event happens, the edge is inserted in S. In the example of Fig. 10, edges L_2, L_5 and L_6 were inserted in S. When a right end point is reached, the edge is removed from S. In the example of Fig. 10, edges L_1, L_3 and L_4 were removed from S. Adjacent edges are processed to verify if they intersect. If they intersect, the intersection point is determined and the intersecting edges are divided. In the example of Fig. 10, edges L_B and L_7 intersect and they will be divided. Fig. 11 shows the types of intersections that causes edge division. The edges in S have the information to which polygon (A or B) and to which type of boundary (regular or degenerated) they belong. In the example, $\{L_1, L_2, L_3, L_4, L_5, L_6, L_7\}$ are edges from polygon A with conventional boundaries, and $\{V_1, L_A, L_B\}$ are elements from polygon B with conventional and degenerated boundaries.

One simple way of finding whether the degenerated vertex is inside or outside a conventional polygon is to test how many times

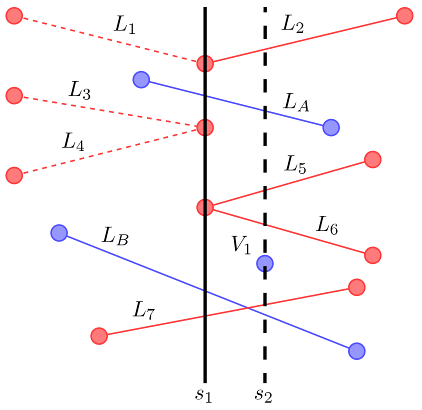

Fig. 10. The thick vertical line s_1 represents the current position of the sweep line. Considering that $S = \{L_2, L_A, L_5, L_6, L_B, L_7\}$. L_1, L_3, L_4 are not in S. Supposing that $\{L_1, L_2, L_3, L_4, L_5, L_6, L_7\}$ are edges from polygon A with exclusively conventional boundary, and $\{V_1, L_A, L_B\}$ are elements from B (that has conventional and degenerated boundaries), when the sweep line s_2 is processed, $S = \{L_2, L_A, L_5, L_6, V_1, L_7, L_B\}$ and as the number of edges from the conventional polygon A above V_1 is odd, one can conclude that V_1 is internal to A.

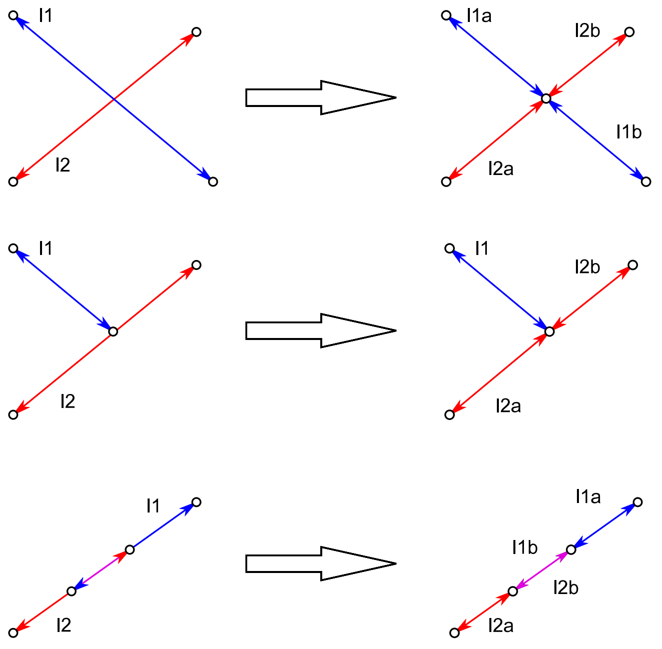

Fig. 11. Types of intersections that cause edge division.

a ray, starting from the degenerated vertex and going any fixed direction, intersects the edges of the conventional polygon. The vertical sweep line s_2 plays the ray role, initially the degenerated vertex V_1 from B is checked to lie on A's conventional boundary. If the degenerated vertex in question is not on the boundary, the number of intersections is even if the degenerated vertex is outside, and it is odd if inside (see Fig. 10).

In the case of a right end point event occurring on a degenerated edge whose left and right end points are isolated, without any contacting edge, such right edge point is classified in similar way as a degenerated vertex. Thus, the intersection module also includes degenerate vertexes and isolated edges classification. Fig. 12 shows an example in which all intersecting vertexes are determined.

4.2. Boundary classification

Conventional and degenerated boundaries from polygon A are classified against polygon B conventional boundary, and vice versa. The edges from a conventional boundary can be classified as: internal, external, coincident shared and opposed shared. As degenerated boundaries do not have a direction, they can be classified as: external, internal and boundary.

The classification starts by analyzing intersecting vertexes with coincident coordinates, they are collected in a circular list D similar to the one proposed by Leonov and Nikitin [24]. Fig. 13 shows an example in which vertexes A_4, B_9 and B_{12} have the same coordinates. The circular list D contains all the edges emanating from the coincident vertexes, and they are ordered according to their horizontal angle. If a vertex belongs to the regular boundary, then it has two edges in D; otherwise, it has just one edge. The circular list D is used to determine counter clockwise and clockwise adjacency and to identify coincident edges.

Coincident edges have the same vertexes coordinates. For conventional polygon edges, based on both edges orientation, the edges are classified as coincident or opposed shared. For degenerated edges they are classified as boundary. Conventional boundaries have sectors in the circular list D. Edges that are internal to the sector are classified as internal. Special care needs to be taken, because one conventional boundary can have more than one sector in D. The edges that were not classified as internal are classified as external. Fig. 12 shows an example in which all the possible intersections were determined and all the edges were classified as internal, external, coincident shared or opposite shared.

4.3. Boundary collection

The boundary collection step is the one in which the result is finally obtained. It has to deal with two important problems: which edges should be collected and in which order. The first problem is solved by adopting a set of boundary collection rules, which determine the inclusion of edges in the final result. For conventional polygons, the rules were defined by Leonov and Nikitin [24] and are operation-specific. Consider an operation A *op* B, in which *op* can be the union or difference operator. A boundary collection rule specifies which label is applied to the edge and to which polygon it belongs (A or B). Also, boundary collection rules determine if the edge will be included with its original orientation or with inverted orientation. Collection rules are also defined for degenerated edges and are naturally equal to the rules of regular boundaries. The only difference is that, as degenerated edges have no orientation, inclusion orientation is not specified. Table 1 shows the boundary collection rules for union and difference operators. It can be observed that orientation is not applicable to degenerated edges, so shared degenerated edges are classified as both coincident shared and opposite shared.

Table 1
Boundary collection rules for operation A *op* B. Conventional boundary and degenerated edges have the same rules, except that degenerated edges have no orientation.

op	Polygon	Label	Orientation
Union	A or B	External or coincident shared	No change
Difference	A	External	No change
	B	Internal	Invert
	A or B	Opposite shared	No change

Fig. 14 shows four degenerated edge collection cases for difference Boolean operation. In this case, A is a collision free region and B is a NFP. As previously explained, a degenerated edge can be classified as internal, external and boundary. The boundary classification can happen with conventional and degenerated boundaries (see Fig. 14(c) and (d)). Situations in which a degenerated edge from A is internal to B, and where a degenerated edge from B is external to A do not create a degenerated edge in the result (see Fig. 14(a) and (b)).

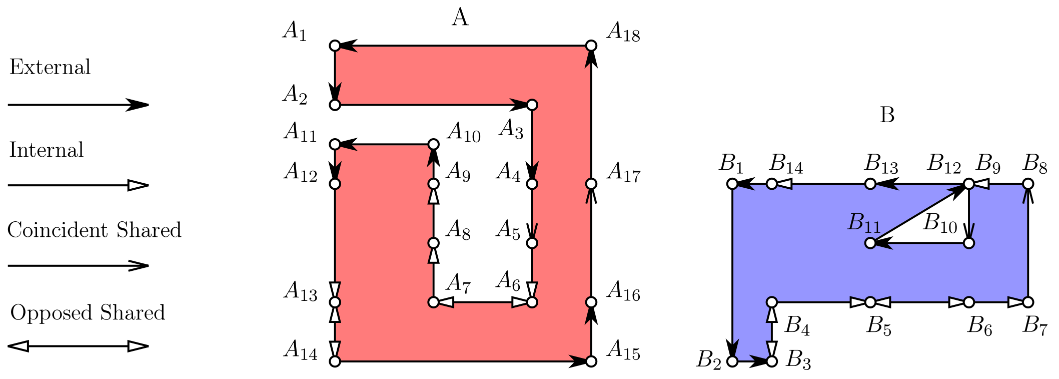
External
Internal
Coincident Shared
Opposed Shared
A
B
A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12} A_{13} A_{14} A_{15} A_{16} A_{17} A_{18}
B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9 B_{10} B_{11} B_{12} B_{13} B_{14}

Fig. 12. Polygons A and B with all intersecting vertexes determined and classified edges.

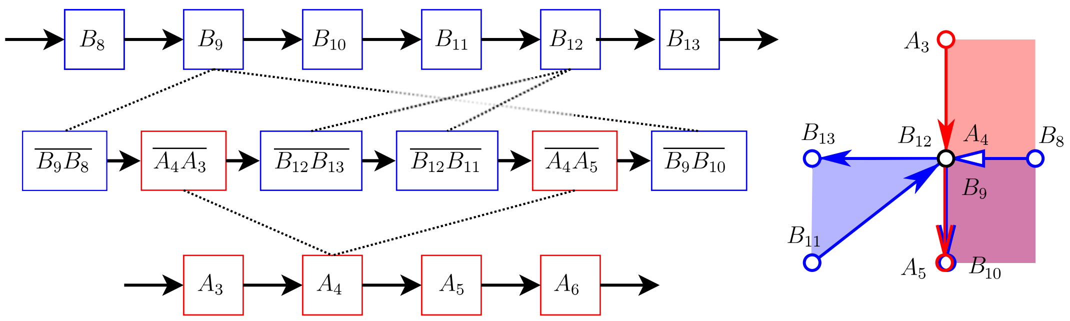
B_8 B_9 B_{10} B_{11} B_{12} B_{13}
$\overline{B_9B_8}$ $\overline{A_4A_3}$ $\overline{B_{12}B_{13}}$ $\overline{B_{12}B_{11}}$ $\overline{A_4A_5}$ $\overline{B_9B_{10}}$
A_3 A_4 A_5 A_6
A_3 B_{13} B_{12} A_4 B_8 B_9 B_{11} A_5 B_{10}

Fig. 13. Circular list D of edges emanated from coincident vertexes A_4, B_9 and B_{12}.
Source: Example taken from Fig. 12.

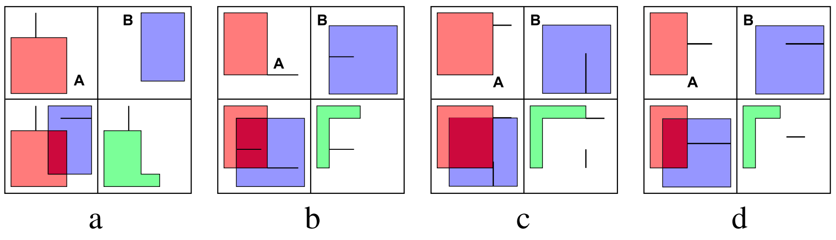

Fig. 14. Four degenerated edge collection cases for difference Boolean operation. A is a collision free region and B is a NFP. (a) Degenerated edge from A is external to B. (b) Degenerated edge from B is internal to A. (c) Degenerated edges from A and B are boundary (conventional). (d) Degenerated edges from A and B are boundary (degenerated).

Fig. 15 shows three degenerated edge collection cases for union Boolean operation. As A and B are NFPs, the vice-versa situation must be considered. The situation in which a degenerated edge from A is internal to B does not create a degenerated edge in the result. Fig. 16 shows five degenerated vertex collection cases for the difference Boolean operation, where A is a collision free region and B is a NFP. Situations in which a degenerated vertex from A is internal to B, and in which a degenerated vertex from B is external to A do not create a degenerated vertex in the result (see Fig. 16(a) and (b)). Fig. 17 shows four degenerated vertex collection cases for union Boolean operation, where A and B are NFPs. The vice-versa situation must be considered. The situation where a degenerated vertex from A is internal to B does not create a degenerated vertex in the result.

The need to adopt a specific collection order in the conventional boundary is justified by the need of obtaining valid oriented contours in the result. If edges have been collected randomly, a new step should be added, in which edges are sorted for obtaining a valid oriented contour. By appropriately choosing connected edges, the final result is a valid connected oriented contour. This suggests that the collection should follow the oriented contour of one of the inputs, in which all the edges are connected and correctly oriented. When an intersection point is encountered, a decision has to be made in order to proceed to the correct contour.

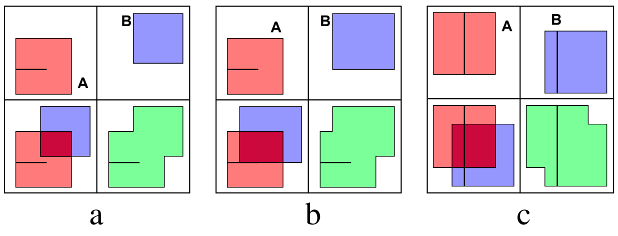

Fig. 15. Three degenerated edge collection cases for union Boolean operation. A and B are NFPs. (a) Degenerated edge from A is external to B. (b) Degenerated edge from A is boundary (conventional). (c) Degenerated edges from A and B are boundary (degenerated).

A new set of boundary collection rules is then defined, called jump rules [24]. According to these rules, when an intersection point is found, the next edge to be collected is the first in counter-clockwise order that follows the boundary collection rules. As mentioned, counter-clockwise adjacency can be determined by using circular list D (see Fig. 13). Degenerated edges are also contained in this list, but they should be ignored in the jump rules, as a conventional boundary is being collected. The collection of the degenerated boundary is very straightforward. As there is no connection between degenerated edges or vertexes, they can be directly

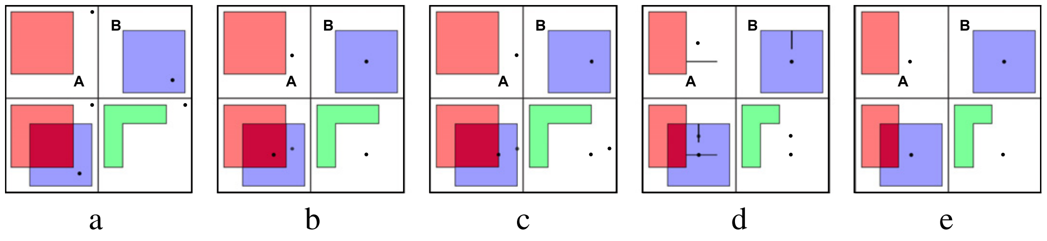

Fig. 16. Five degenerated vertex collection cases for difference Boolean operation. *A* is a collision free region and *B* is a NFP. (a) Degenerated vertex from *A* is external to *B*. (b) Degenerated vertex from *B* is internal to *A*. (c) Degenerated vertexes from *A* and *B* are boundary (conventional). (d) Degenerated vertexes from *A* and *B* are on boundary (degenerated edge–vertex coincidence). (e) Degenerated vertexes from *A* and *B* are boundary (degenerated vertexes coincidence).

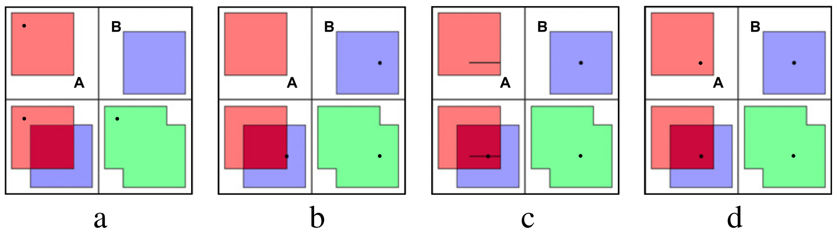

Fig. 17. Four degenerated vertex collection cases for union Boolean operation. *A* and *B* are NFPs. (a) Degenerated vertex from *A* is external to *B*. (b) Degenerated vertex from *B* is boundary (conventional). (c) Degenerated vertex from *B* is boundary (degenerated edge–vertex coincidence). (d) Degenerated vertexes from *A* and *B* are boundary (degenerated vertexes coincidence).

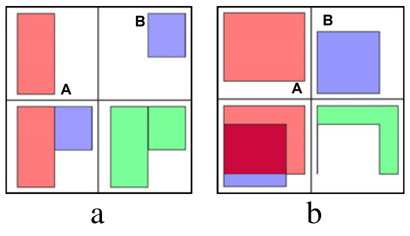

Fig. 18. Two degenerated edge creation cases. (a) Shows a union Boolean operation in which *A* and *B* are NFPs. It is an opposed shared edge case. (b) Shows a difference Boolean operation where *A* is a collision free region and *B* is a NFP. It is a coincident shared edge case.

included in the result if they obey the boundary collection rules (see Figs. 14–17).

4.4. Degenerated boundary creation

Possibly the most significant difference between regularized and non-regularized Boolean operations algorithms is that, in the latter, additional checking must be performed to determine if new elements should be included in the result boundary. Such elements are always degenerated edges or vertexes. In regularized Boolean operations algorithms, boundary edges from the result are always found in the input boundaries. If degenerated boundaries are to be processed separately, the result of a non-regularized Boolean operations algorithm can have degenerated edges which were not contained in the input degenerated boundaries. Furthermore, degenerated vertexes can also only exist in the final result. To determine whether a new degenerated edge will be created, a rule similar to a boundary collection rule is checked. The edge is then included in the degenerated boundary (see Fig. 18).

While the creation of degenerated edges follows the same procedure as the collection of the regular boundary, checking for new degenerated vertexes is a more complex task. Degenerated vertexes arise from the crossing of edges, more specifically, crossing where no edges follow boundary collection rules, including the one regarding the creation of degenerated edges. This is justified by the fact that all the intersecting vertexes are part of the final result. They are usually an endpoint of a collected edge or a degenerated edge and the only exception is when none of the edges that have the intersecting vertex as its endpoint are collected. Fig. 19(a) and (b) show two cases of vertex creation for the union Boolean operation and Fig. 19(c) and (d) show two cases for the difference Boolean operation. Other possible cases can involve more or less crossing edges; however, the generation rule is the same, i.e. no edges must follow the boundary collection rules. It is important to note that degenerated vertex creation should not be checked alongside jump rules, as the latter occurs on intersections in which one edge is already included. Table 2 shows the creation rules for the union and difference operations.

Degenerated boundaries classification and collection are processed in a different module. Degenerated edge creation is only

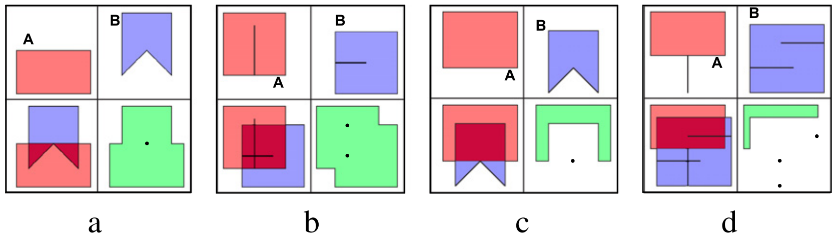

Fig. 19. Four degenerated vertex creation cases. (a)–(b) Show union Boolean operations where *A* and *B* are NFPs. (a) Degenerated vertex is created from conventional boundaries. (b) Degenerated vertex is created from degenerated and conventional boundaries intersection. (c)–(d) Show difference Boolean operations in which *A* is a collision free region and *B* is a NFP. (c) Degenerated vertex is created from conventional boundaries. (d) Degenerated vertex is created from degenerated and conventional boundaries intersection.

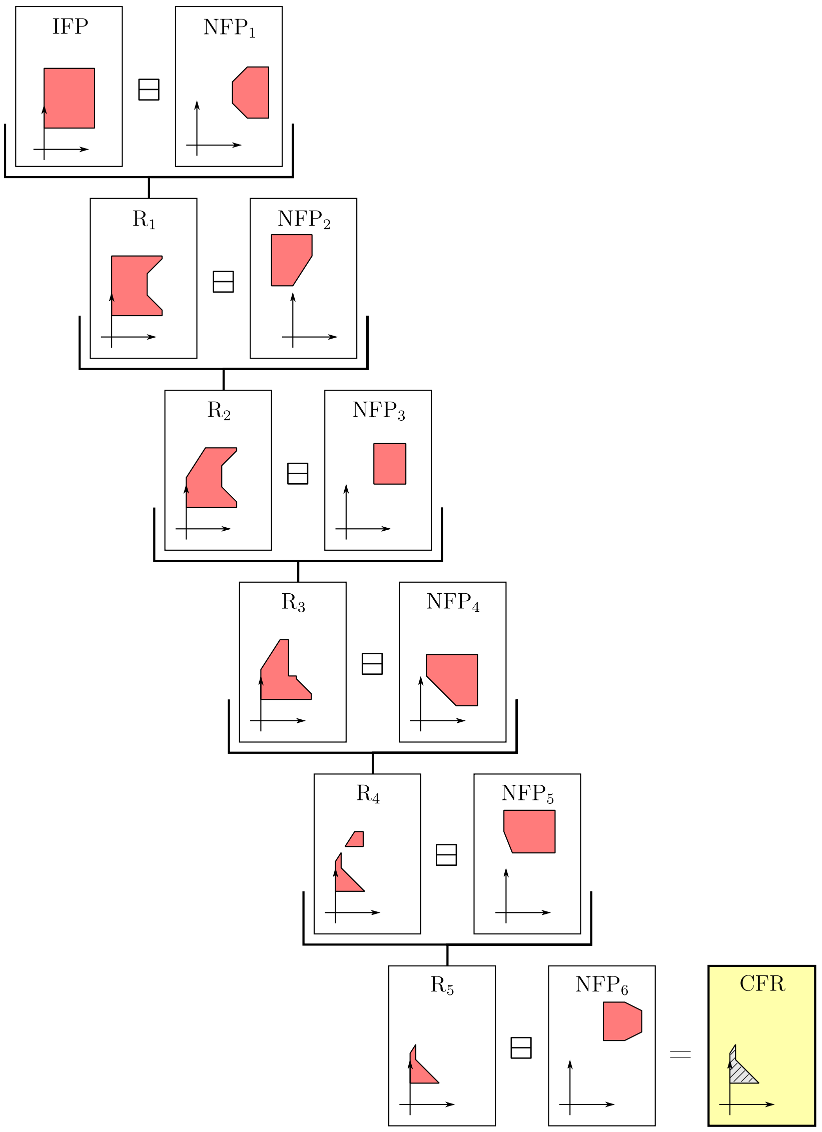

Fig. 20. Example of a serial determination of the collision free region using exclusively the difference Boolean operation. NFP: nofit polygon. IFP: inner fit polygon. CFR: collision free region. R_1–R_5 represents the results of the Boolean operations (**DIFF**).

Table 2
Generation rules for degenerated edges and vertexes.

op	Type	Rule
Union	Edge	Opposite shared labeled
Difference	Edge	Coincident shared labeled
Both	Vertex	Crossing of exclusively uncollected edges

possible for conventional boundary coincident edges, and it is thus processed in the conventional boundary module. On the other hand, degenerated vertexes can be originated from any crossing vertex; hence, all the intersections must be checked and thus it does not fit in a single module. The resulting degenerated boundary is composed of degenerated edges collected from conventional boundary edges and degenerated vertexes and edges collected from input degenerated boundaries.

4.5. Parallelization

Collision free region determination is performed using a sequence of NFPs as input. Eq. (5) shows the necessary operations to be performed. In this work; the collision free region is calculated according to three different implementations. Considering that $\mathcal{P}$ is a queue and n is the number of items in the queue. The first implementation is shown in Algorithm 1; it only consists of difference operators, as the collision free region is determined by serially subtracting every NFP from the inner fit polygon (see Fig. 20). The first implementation has no parallelization.

```
$\Pi = \Lambda(\mathcal{C}, P_{n+1})$;
**for** $i = 1$ **to** $n$ **do**
    $\Pi = \Pi \boxminus \{i(\text{pop}(\mathcal{P})) \ominus i(P_{n+1})\}$;
```

Algorithm 1: Collision free region determination using exclusively difference Boolean operations serially (**DIFF**).

The second implementation is an algorithm in which all NFPs are serially united and then subtracted from the inner fit polygon, shown in Algorithm 2. Union and difference operators are used in this implementation, and there is no parallelization (see Fig. 21). The union of all NFPs is defined as the obstructed region, as it represents all the forbidden translation for a given item, given a set of already placed items.

The third implementation is a parallel algorithm to calculate the obstructed region described in Algorithm 3. In this algorithm,

```
$\Pi$ =i(pop($\mathcal{P}$)) $\ominus i(P_{n+1})$;
for $i = 2$ to $n$ do
    $\Pi = \Pi \uplus \{i(\text{pop}(\mathcal{P})) \ominus i(P_{n+1})\}$;
$\Pi = \Lambda(\mathfrak{C}, P_{n+1}) \boxminus \Pi$;
```

Algorithm 2: Collision free region determination using unions and one difference Boolean operation (**UNI**).

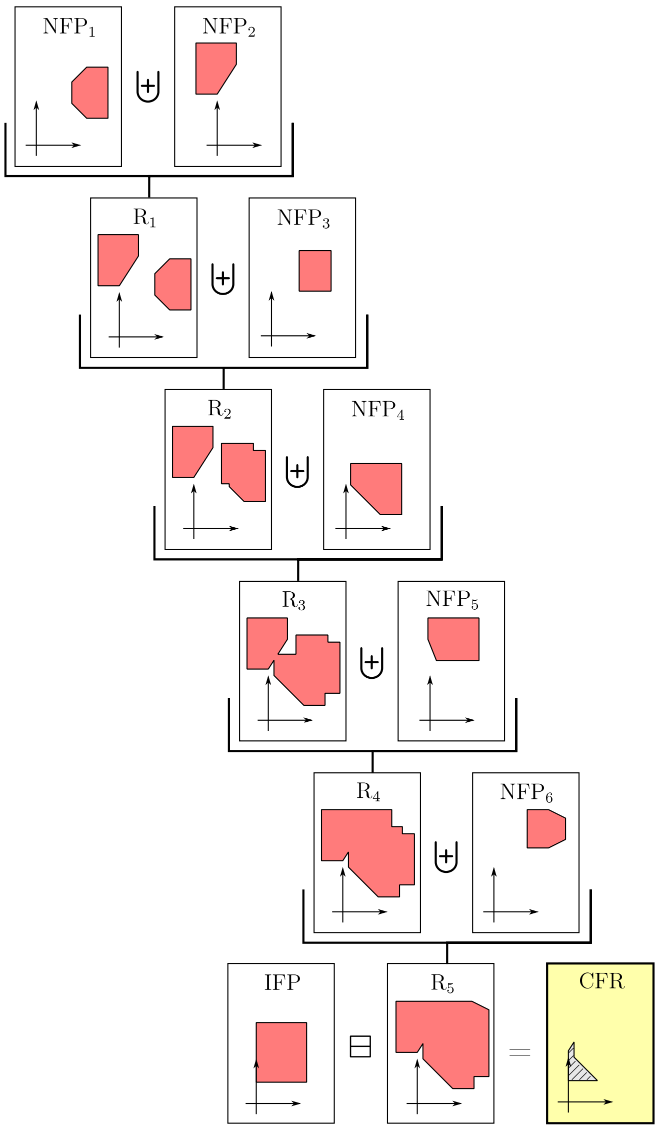

Fig. 21. Example of a serial determination of the collision free region by initially determining the obstructed region. NFP: nofit polygon. IFP: inner fit polygon. CFR: collision free region. R$_1$–R$_5$ represents the results of the Boolean operations (**UNI**).

union operations are processed in parallel. Initially, all NFPs are determined and pushed in queue $\mathcal{R}$. Each thread pops two NFPs, the NFPs are united to define an obstructed region, and the obstructed region is pushed back at the end of the queue. Just one thread accesses the queue at a time. If the queue does not have two obstructed regions, the threads that finished the union operation are kept in an idle state until the queue has enough obstructed regions. The procedure is repeated $n - 1$ times, where n represents the total number of NFPs, until the queue has just one obstructed region. At this moment, the threads are killed and the collision free region is determined by subtracting the obstructed region from the inner fit polygon. Fig. 22 shows a example of parallel collision free region determination using the proposed algorithm. All horizontally

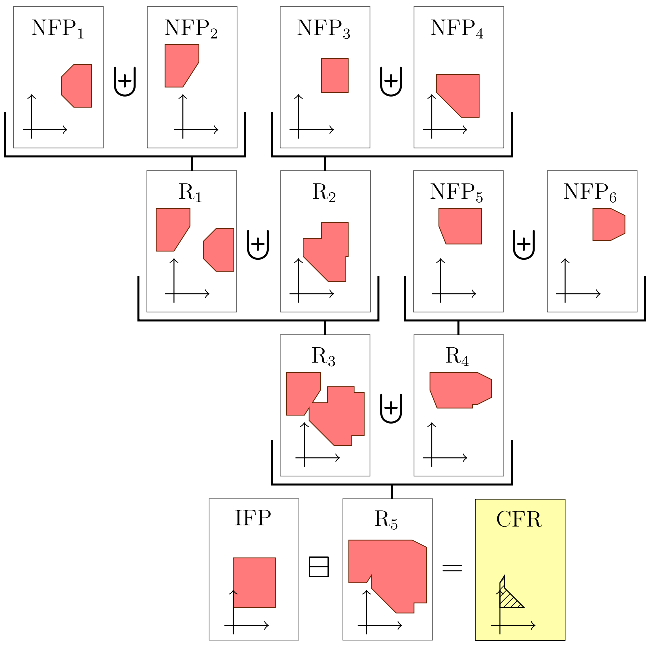

Fig. 22. Example of a parallel determination of the collision free region. NFP: nofit polygon. IFP: inner fit polygon. CFR: collision free region. R$_1$–R$_5$ represents the results of the Boolean operations. Considering that the original sequence of NFPs is {NFP$_1$, NFP$_2$, NFP$_3$, NFP$_4$, NFP$_5$, NFP$_6$}, the results are determined in the following order: {R$_1$, R$_2$, R$_3$, R$_4$, R$_5$ } (**UD**).

aligned operations are performed in parallel. Finally, to obtain the collision free region, the obstructed region is subtracted from the inner fit polygon.

```
for $i = 1$ to $n$ do
    $\Pi = \{i(\text{pop}(\mathcal{P})) \ominus i(P_{n+1})\}$;
    pushBack($\mathcal{R}, \Pi$);
for $i = 2$ to $n$ do
    $\Pi$ =pop($\mathcal{R}$)$\uplus$pop($\mathcal{R}$);
    pushBack($\mathcal{R}, \Pi$);
$\Pi = \Lambda(\mathfrak{C}, P_{n+1}) \boxminus$pop($\mathcal{R}$);
```

Algorithm 3: Collision free region determination using parallel unions and one difference Boolean operation (**UD**).

5. Results

In order to demonstrate the speed of the proposed approach, generation statistics for 15 benchmark problems gathered from the literature are shown. These data sets can be found on the EURO Special Interest Group on Cutting and Packing (ESICUP) website.[1] For each problem, the time to generate 1000 times the collision free region for all items is reported. Placement order and position of items were randomly defined. The last item in the sequence is the item to be placed. Table 3 displays the results obtained using exclusively the difference Boolean operation (see Fig. 20). All the experiments were conducted on an i7 860 multi-core processor with 4 cores and 4 GB RAM. Sato et al. [29] used this approach to determine the collision free region, and the Albano, Dagli, Jakobs and Marques solutions found by the proposed algorithm are the best results published in the literature.

Fig. 23 shows an example of the determination of the collision free region for a single item. The final layout, current item P$_7$ and

1 http://www.fe.up.pt/esicup.

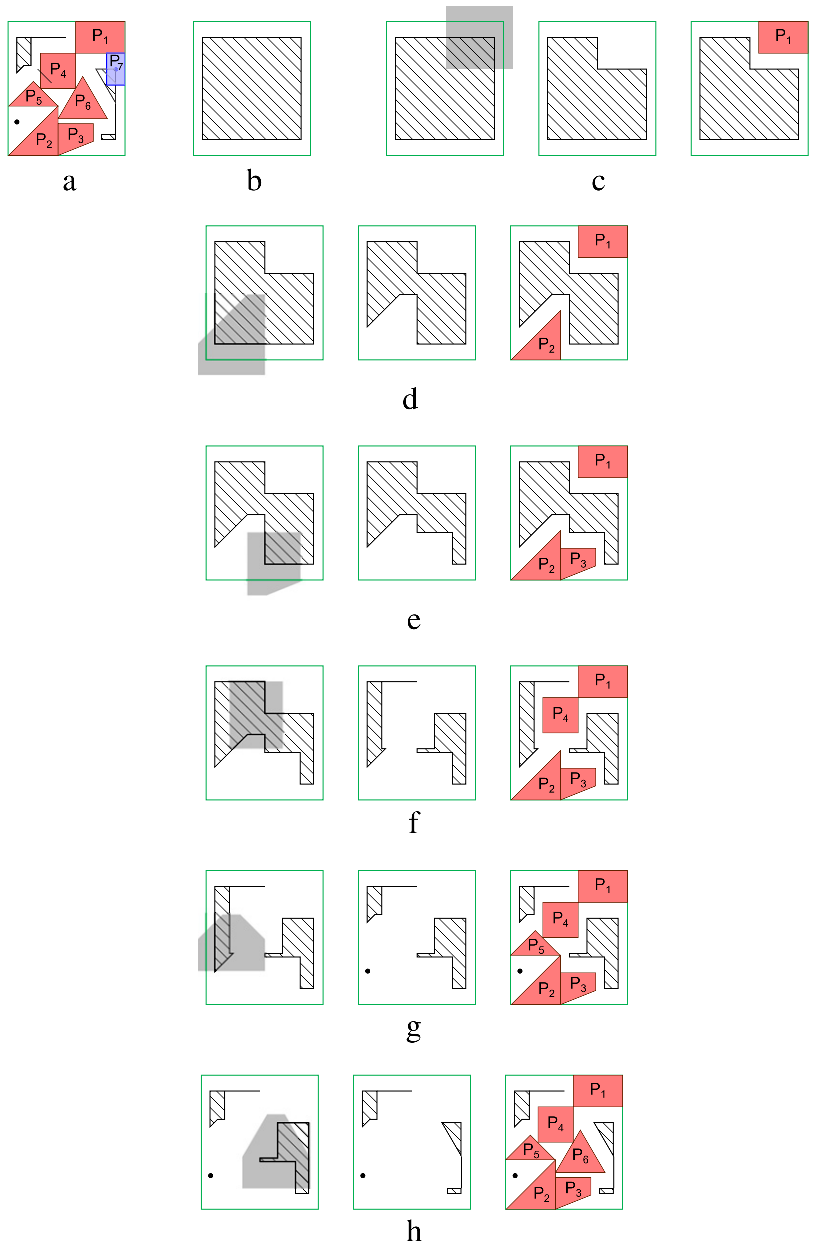

Fig. 23. Determination of the collision free region for item P_7. Items and container geometry obtained from the Fu problem instance. Items P_1, P_2, P_3, P_4, P_5 and P_6 are already placed. The final collision free region has one degenerated vertex and two degenerated edges where item P_7 can be placed.

Table 3
Generation of all collision free region in benchmark data sets repeated 1000 times. **TNP**: total number of polygons; **ANE**: the average number of edges (cited from [6]—the total number of vertexes is exactly the same as the total number of edges).

Case	TNP	ANE	Time (s)
Albano	24	7.25	2.33
Dagli	30	6.30	2.20
Dighe1	16	3.87	0.63
Dighe2	10	4.70	0.58
Fu	12	3.58	0.62
Jakobs1	25	5.60	1.54
Jakobs2	25	5.36	1.91
Mao	20	9.22	1.77
Marques	24	7.37	2.23
Shapes0	43	8.75	4.24
Shapes1	43	8.75	4.38
Shapes2	28	6.29	1.39
Shirts	99	6.63	12.53
Trousers	64	5.06	12.00

the corresponding collision free region, represented by the hatch pattern filled polygons with degenerated edges and a degenerated vertex, are displayed in Fig. 23(a). The inner fit polygon of item P_7 can be observed in Fig. 23(b). Fig. 23(c)–(h) represent the difference sequence, each relating to one Boolean operation. Both input polygons, the intermediate collision free region and the NFP, are shown on the left. The placed item used to generate the NFP is exhibited on the right and the result is displayed in the middle.

To compare processing times for the three implementations proposed, a new batch of tests was executed. A parallel version of the algorithm with the obstructed region is also included in the tests. Table 4 shows execution times obtained for these evaluations. Three serial implementations were developed: **DIFF**, **UNI** and **UD** (serial). The minimum between the three is selected to create the graph shown in Fig. 24. Considering the same implementation, the main difference in time among the cases is related to the total number of edges. One might observe that Shapes0, Shapes1 and

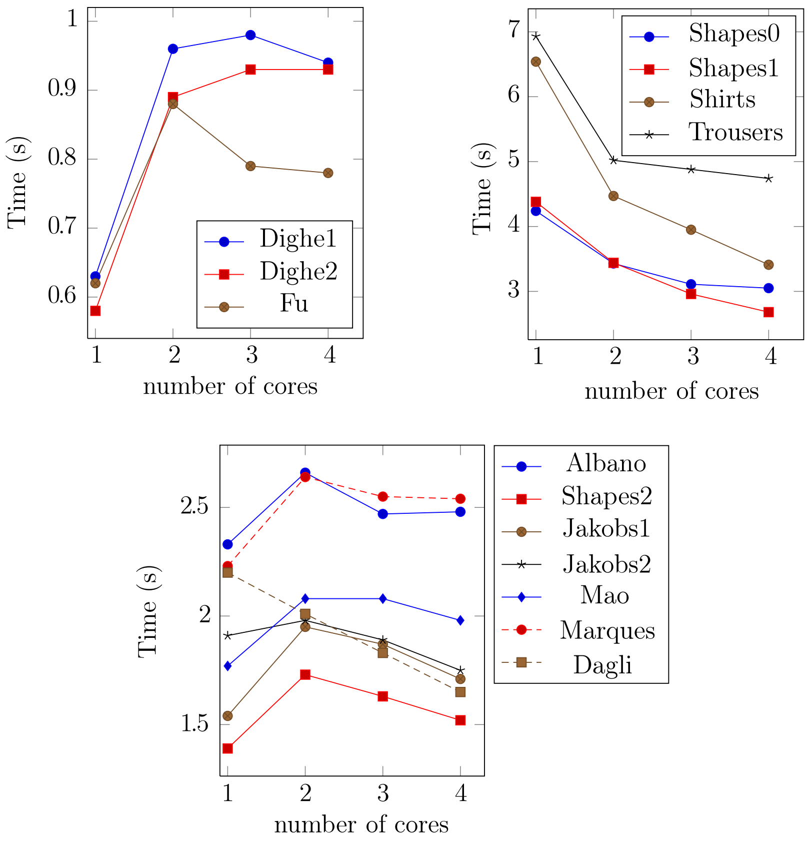

Fig. 24. Parallel collision free region generation times.

Trousers have a similar total number of edges (376), but the processing time for Trousers is almost three times the Shapes0 and Shapes1 time processing. As intermediary results of Shapes0 and Shapes1 are determined, the total number of edges is kept low. On the other hand, the intermediary results for Trousers have a high total number of edges. Considering that the same sequence of NFPs is used for all the implementations, for the same case there are differences in the time processing as NFPs are combined in different ways. The three serial implementations show different time processing as intermediary results have a different total number of edges. From Fig. 24, it is possible to observe that the adoption of the obstructed region is only beneficial in puzzles with a high number of items. It can also be noted that Dighe1, Dighe2, Fu, Jakobs1, Mao, Marques and Shapes2 problems ran slower using the parallel algorithm when compared to its faster serial counterpart.

Table 4
Single collision free region generation repeated 1000 times. **TNE**: total number of edges; **DIFF**: execution times (s) using only differences (serial); **UNI**: execution time (s) using unions and difference; **UD**: execution times (s) using unions and difference.

Case	TNE	DIFF	UNI	UD			
				Serial	2 cores	3 cores	4 cores
Albano	174	2.33	3.80	3.13	2.66	2.47	2.48
Dagli	189	2.20	3.49	2.53	2.01	1.83	1.65
Dighe1	62	0.63	0.97	1.03	0.96	0.98	0.94
Dighe2	47	0.58	0.85	0.93	0.89	0.93	0.93
Fu	43	0.62	0.67	0.85	0.88	0.79	0.78
Jakobs1	140	1.54	2.92	2.37	1.95	1.87	1.71
Jakobs2	134	1.91	2.90	2.34	1.98	1.89	1.75
Mao	184	1.77	2.98	2.31	2.08	2.08	1.98
Marques	177	2.23	4.29	3.24	2.64	2.55	2.54
Shapes0	376	4.24	7.16	4.57	3.43	3.11	3.05
Shapes1	376	4.38	7.31	4.66	3.44	2.96	2.68
Shapes2	176	1.39	2.58	2.06	1.73	1.63	1.52
Shirts	656	12.53	17.28	6.54	4.47	3.95	3.41
Trousers	324	12.00	17.21	6.93	5.02	4.88	4.74

6. Conclusions

The collision free region concept is defined and non regularized Boolean operations are shown to be necessary to correctly determine it. Three different algorithms can be defined to determine the collision free region: one based exclusively on difference operations, and the other based on union and difference operations. One algorithm that uses union and difference operations can be parallelized (**UD**).

The collision free region has two types of boundaries: conventional boundary and degenerated vertexes and edges. A robust non regularized Boolean operation algorithm using finite precision is proposed in order to compute the union between NPFs and the difference between a collision free region and a NFP. The proposed algorithm robustly determines conventional boundaries, degenerated vertexes and degenerated edges.

The parallelized algorithm was implemented and tested. Execution times for serial and parallelized algorithms were measured and the advantage of the parallelized algorithm was only observed in problems with a larger number of items.

Acknowledgments

André Kubagawa Sato was supported by CNPq and FAPESP (Grant 2010/19646-0). Thiago Castro Martins was supported by FAPESP (Grant 2009/14699–0). Marcos Sales Guerra Tsuzuki was partially supported by CNPq (Grants 304.258/2007–5 and 309.570/2010–7). Thiago de Castro Martins was partially supported by CNPq (Grant 06415/2012-7). This research was supported by FAPESP (Grants 2008/13127–2 and 2010/18913–4).

References

[1] Bennell JA, Oliveira JF. The geometry of nesting problems: a tutorial. European Journal of Operational Research 2008;397–415.
[2] Adamowicz M, Albano A. Nesting two dimensional shapes in rectangular modules. Computer Aided Design 1976;8(1):27–33.
[3] Babu AR, Babu NR. A generic approach for nesting of 2D parts in 2D sheets using generic and heuristic algorithms. Computer Aided Design 2001;33:879–91.
[4] Lee W-C, Ma H, Cheng B-W. A heuristic for nesting problems of irregular shapes. Computer Aided Design 2008;40:625–33.
[5] Dowsland KA, Vaid S, Dowsland BW. An algorithm for polygon placement using a bottom-left strategy. European Journal of Operational Research 2002; 141:371–81.
[6] Gomes AM, Oliveira JF. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. European Journal of Operational Research 2006;171:811–29.
[7] Egeblad J, Nielsen BK, Odgaard A. Fast neighborhood search for two- and three-dimensional nesting problems. European Journal of Operational Research 2007;183:1249–66.
[8] Imamichi T, Yagiura M, Nagamochi H. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. Discrete Optimization 2009;6:345–61.
[9] Martins TC, Tsuzuki MSG. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. Expert Systems with Applications 2010;37:1955–72.
[10] Ghosh PK. An algebra of polygons through the notion of negative shapes. CVGIP: Image Understanding 1991;54:119–44.
[11] Dean HT, Tu Y, Raffensperger JF. An improved method for calculating the no-fit polygon. Computers & Operations Research 2006;33:1521–39.
[12] Bennell JA, Song X. A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. Computers & Operations Research 2008; 267–81.
[13] Agarwal PK, Flato E, Halperin D. Polygon decomposition for efficient construction of minkowski sums. Computational Geometry 2002;21:39–61.
[14] Li Z, Milenkovic V. Compaction and separation algorithms for non-convex polygons and their applications. European Journal of Operational Research 1995;84:539–61.
[15] Burke EK, Hellier RSR, Kendall G, Whitwell G. Complete and robust no-fit polygon generation for the irregular stock cutting problem. European Journal of Operational Research 2007;179:27–49.
[16] Gomes AM, Oliveira JF. A 2-exchange heuristic for nesting problems. European Journal of Operational Research 2002;141:359–70.
[17] Berg M, Cheong O, Kreveld M, Overmars M. Computational geometry: algorithms and applications. 3rd ed. Springer-Verlag; 2008.
[18] Martins TC, Tsuzuki MSG. Simulated annealing applied to the rotational polygon packing. In: Proceedings of the IFAC symposium information control problems in manufacturing. 2006. p. 475–80.
[19] Hoffmann CM. The problems of accuracy and robustness in geometric computation. Computer 1989;22:31–41.
[20] Bentley JL, Ottmann TA. Algorithms for reporting and counting geometric intersections. IEEE Transactions on Computers 1979;C-28:643–7.
[21] Hu C-Y, Patrikalakis NM, Ye X. Robust interval solid modelling part I: representations. Computer Aided Design 1996;10:807–17.
[22] Wallner J, Krasauskas R, Pottmann H. Error propagation in geometric constructions. Computer Aided Design 2000;32:631–41.
[23] Martínez F, Rueda AJ, Feito FR. A new algorithm for computing Boolean operations on polygons. Computers & Geosciences 2009;35:1177–85.
[24] Leonov MV, Nikitin AG. An efficient algorithm for a closed set of Boolean operations on polygonal regions in the plane. Tech rep. A.P. Ershov Institute of Informatics Systems; 1997.
[25] Requicha AAG. Representation of solid objects—theory, methods and systems. ACM Computing Surveys 1980;12(6):437–64.
[26] Greiner G, Hormann K. Efficient clipping of arbitrary polygons. ACM Transactions on Graphics 1998;17:71–83.
[27] Liu YK, Wang XQ, Bao SZ, Gombosi M, Zalik B. An algorithm for polygon clipping, and for determining polygon intersections and unions. Computers & Geosciences 2007;33:589–98.
[28] Vatti BR. A generic solution to polygon clipping. Communications of the ACM 1992;35:56–63.
[29] Sato AK, Martins TC, Tsuzuki MSG. An algorithm for the strip packing problem using collision free region and exact fitting placement. Computer Aided Design 2012;44:766–77.