Robust NFP generation for Nesting problems*

Pedro Rocha †

INESCTEC - Instituto de Engenharia de Sistemas e Computadores Tecnologia e Ciência, ISEP - Instituto Superior de Engenharia do Porto, Porto, Portugal

March 28, 2019

Abstract

Cutting and packing problems arise in a large variety of industrial applications, where there is a need to cut pieces from a large object, or placing them inside a containers, without overlap. When the pieces or the containers have irregular outline, the problem is classified as a Nesting problem. The geometrical challenges of the Nesting problem are addressed by focusing on the geometric aspect of the 2D pieces and containers involved. The challenges of the geometrical component are mainly derived from the complexity of the pieces, due to high number of vertices, which is common when dealing with real world scenarios. This complexity is challenging for current algorithms to process efficiently and effectively, leading to high computational cost and less satisfactory results, particularly when dealing with overlap verification operations. Usually, when tackling Nesting problems, the overlap verification process between two objects is done through the use of a structure known as No-Fit-Polygon (NFP).

In this work, the generation of the NFP is achieved through a simple algorithm which produces a simplified shape while reducing numerical precision errors and fully representing the region that forms the NFP including positions with perfect fits.

1 Introduction

Many problems that arise in several real world scenarios require the placement of pieces inside a container (or cutting pieces from a large object) without overlap and while aiming to minimize wasted space by achieving a compact configuration. If one of these problems deals with irregular pieces or containers it is classified as Nesting problem, which is also known as an Irregular Piece Packing problem. This problem requires an adequate geometric representation of its objects with great accuracy, which usually generates a significant amount of geometric information (vertices, pixels, etc), creating difficulties for current algorithms. One of the main challenges in the geometrical component of the Nesting problem is ensuring the non-overlap and correct containment of the pieces, while maintaining an acceptable level of computational efficiency.

The overlap verification in Nesting problems is usually achieved by using a geometrical structure known as No-Fit-Polygon (NFP). This structure is a set of feasible placement locations of one polygon relative to another, that allows specifying the regions and placement locations where pieces do not overlap. In this paper, an algorithm is presented that generates the NFP with a reduced number of geometrical components that define it, while using a simplified process that deals with numerical precision problems. This improves the quality of the NFP while also reducing the overhead cost of overlap verification.

This paper is organized in several sections, being this one the introduction. The remainder of this section contains the motivation for this work and main contributions. The following section, the second, present a literature review, making reference to some approaches used generate the NFP focusing on Nesting requirements. The third section contains a description of our approach, while the fourth presents alternative applications for the proposed algorithm followed by fifth section with the conclusions and future work.

 $^{^*}$ This research was partially supported by Project TEC4Growth - RL iMAN - Intelligence for Advanced Manufacturing Systems (NORTE-01-0145-FEDER-000020)

[†]Electronic address: pedro.f.rocha@inesctec.pt; pfr@isep.ipp.pt

1.1 Motivation

The proposed approach addresses the challenges of NFP generation, dealing with numerical precision problems and generating a NFP with a reduced amount of geometric components while maintaining its representation quality. Since the NFP is extensively used in approaches addressing Nesting problems, improving it leads to computational efficiency gains and possible improvements in solution quality in cases where a complete NFP representation was not previously used.

Improving the efficiency of the geometrical component leads to more resources being devoted to addressing the combinatorial component of the Nesting problem, which can enable the use of new approaches, and reaching higher quality solutions.

The improvement of current geometric overlap verification methods has a relevant impact on many scientific fields where similar problems arise, considering both economic and environmental aspects.

1.2 Contribution

In many real world scenarios, such as industrial applications that deal with leather, furniture, metallurgy, shipbuilding, sheet metal cutting, plastics, and others, arise problems that require placing a set of pieces into a container, in the most efficient non-overlapping configuration that minimizes wasted space. These problems require dealing with geometrical aspects derived from the structure of the pieces and their relative position inside the container, which must dealt with effectiveness and efficiently, while managing the trade off between solution quality and computational cost.

This work has a relevant impact in several industries and scientific fields, at various levels, wherever similar problems arise, producing benefits of economic (such as cost reduction) and environmental (less waste, energy and raw material consumption) importance. It provides an algorithm that is simple to implement, and generates overlap verification structures based on No-Fit-Polygons that are very robust. This has a significant impact by improving current approaches and opening new research paths in previously underdeveloped areas such as 3D irregular packing.

2 Literature Review

The placement of irregular items in the most efficient placement positions inside a container is heavily dependent on the geometrical structures that are used. The items cannot overlap and must completely fit inside the container. In the literature one can find approaches that are based on distinct geometric representations such as grids, circles, direct trigonometry, phi-functions and No-Fit-Polygons (NFP). In [Bennell and Oliveira, 2008] can be found an introduction to these different representations focusing specifically on Nesting problems.

The NFP has been a commonly used approach due to its increased efficiency compared to direct trigonometry, having the benefit of being built directly from the edges of the polygons that represent the real objects ([Bennell and Oliveira, 2008]).

2.1 The No-Fit-Polygon

The NFP is a collection of all feasible placement locations of a polygon relative to another. It transforms the overlap verification process between two polygons into the overlap verification between a polygon and a vertex, which is more computationally efficient. Another benefit is that the NFP can be generated at a pre-processing phase, but only for a reduced set of possible piece orientations. In situations where continuous rotations are advantageous, this approach is not adequate due to its significant computational cost generating the NFP for every new orientation of the pieces. One drawback of this approach is also the difficulty in developing a robust approach to generating the NFP from pairs of generic irregular shapes, due to emergence of degenerate cases. When the NFP is generated between a piece and the container, or hole, it is known as the Inner-Fit-Polygon (IFP). An example of a NFP and IFP generated by a sliding algorithm can be seen in Fig. 1 and an example of its degenerate cases can be seen in Fig. 2.

2.2 No-Fit-Polygon Generation

In the literature, there are three main approaches dedicated to generate the NFP. They are based on a Sliding Algorithm, Slope Diagrams or Minkowsky Sum and Polygonal Decomposition. There is also another approach that specifies the inverse region of the NFP, defined as Collision Free Region (CFR).

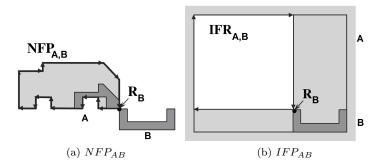


Figure 1: NFP between polygon A and polygon B (a) and IFP between board A and polygon B (b) (adapted from ([Gomes and Oliveira, 2006]))

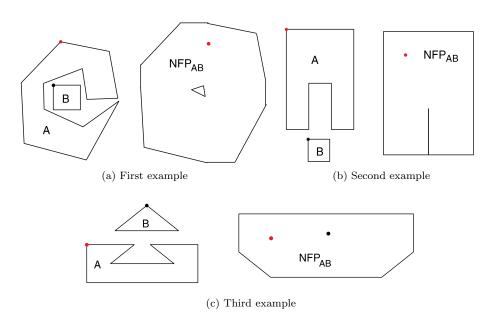


Figure 2: Examples of combinations of polygons that generate NFPs and their degenerated cases (adapted from [Bennell and Oliveira, 2009])

The sliding algorithm was initially proposed by [Mahadevan, 1984] and computes the NFP using an approach that simulates the sliding of an orbiting piece around a stationary piece, without ever loosing contact. The NFP is defined by the path created by the reference point of the orbital piece while it slides around the stationary piece until it reaches the initial position. This approach has difficulties dealing with pieces with concavities that have very narrow entries or internal holes. It also has several challenges when the scale of the pieces very large, since the orbital piece may lose contact with the edges of the stationary due to numerical precision errors. Another challenge is the identification of possible placement positions inside holes of the stationary piece, including perfect fit situations. This approach was improved by [Whitwell, 2004] which tries to address perfect fit sliding situations and an extension by [Burke et al., 2006] addresses the identification of holes that have potential for feasible placement positions. This is also explored further in [Burke et al., 2007]. The process for generating the NFP using the orbital algorithm can be seen in Fig. 3.

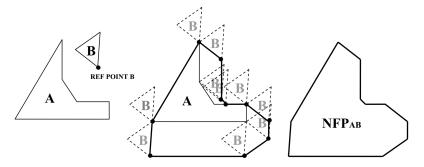


Figure 3: Sliding example to create the NFP between piece A and B.([Burke et al., 2007]).

The approach based on slope diagrams was proposed by [Cuninghame-Green, 1989] which computes the NFP using pairs of convex polygons, by using an ordered list of the edges of both polygons ordered by slope. The edges are added sequentially building the NFP until all edges are used. This approach has limited application due to only supporting convex polygons, which was addressed by [Ghosh, 1991] which extended its use for non-convex cases. When dealing with concavities, the approach does not use a sequence of ordered edges by slope, but it repeats the edges of the convex polygon every time it interacts with a concavity of the other polygon. The limitation of this method appears when the concavities interact with each other. The challenge of two concave polygons was addressed by [Bennell et al., 2001] by replacing the concave edges for artificial edges of the orbital polygon, which simulates a convex polygon. The artificial edges are then replaced by the original concave edges including other transversed edges by the stationary polygon. Other improvements can be seen in [Dean et al., 2006], [Bennell and Song, 2008]. This approach is also used to generate a concept similar to the NFP known as the Minkowsky sum, as seen in [Milenkovic et al., 1991] and [Bennell et al., 2001]. An example of the process for generating the NFP using Minkowsky sum can be seen in Fig. 4.

The approach based on polygonal decomposition computes the NFP by decomposing each irregular polygon into convex sub-polygons and then generating the NFP from each pair of convex sub-polygons from different pieces. The sets of convex NFPs for each pair of pieces can be used directly, or they may be merged together to obtain the full NFP from the original polygons. This merging process is the most challenging since its very complicated to merge the polygons while dealing with numerical precision errors and detecting perfect fit or perfect sliding placement positions. One of the challenges is how to produce the minimum number of convex sub-polygons for each polygon, which impacts the total number of NFPs between each pair of pieces, and also their complexity due to the number of vertices of each sub-polygon, depending on the approach used for decomposition. An example with several degrees of polygonal decomposition can be seen in Fig. 5. Some authors that use convex decomposition are [Watson and Tobias, 1999], [Babu and Babu, 2001] and [Agarwal et al., 2002]. Another similar approach is the decomposition into star-shaped polygons which can be seen in [Li and Milenkovic, 1995].

The approach based on the generation of a Collision Free Region ([Sato et al., 2013]) is created through boolean operations involving the NFP and IFP. This approach is still not widely used but it is very relevant. Its main advantage is the representation through a single polygonal structure that defines the CFR instead of using multiple simpler polygons (as when using convex decomposition) which simplifies overlap verification computations. The main downside arises when concavities and degenerated cases appear which make overlap verifications much more complex and computational expensive. An example of the CFR can be seen in Fig. 6.

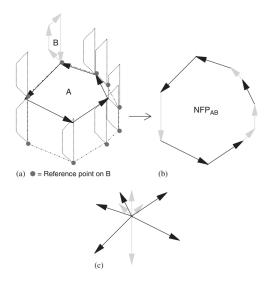


Figure 4: Slope example to create NFP between piece A and B.([Bennell and Song, 2008])

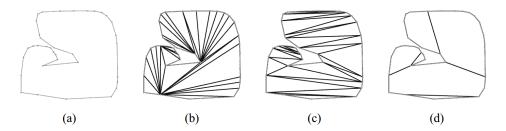


Figure 5: Decomposition example.([Agarwal et al., 2002])

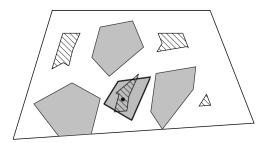


Figure 6: Colision Free Region example.([Sato et al., 2013])

The NFP is an efficient method to compute efficiently the relative position between pairs of pieces but the processes available for its generation are still limited. The correct identification of holes and perfect fit or perfect sliding placement positions need is still a computational expensive operation (due to being treated as degenerate cases, among other reasons) and there are still many problems derived from numerical precision which cause errors that need to be addressed.

The next section presents a new approach that aims to address some of these challenges.

3 Proposed NFP Algorithm

In this section is presented a simple algorithm that reduces the challenges generating NFPs that contain holes, perfect sliding placement positions and perfect fit placement positions. As a side not, this algorithm can also be used to merge normal polygons, however, the perfect fit and perfect sliding placement positions will be discarded because normal polygons do not have zero area regions.

3.1 Convex Decomposition and NFP Generation

The convex decomposition of the pieces is achieved through any algorithm using partition or covering decomposition approaches. As explained in the example of Polygonal Convex Decomposition, one of the main concerns is the total number of sub-polygons and vertices generated, which impact significantly and proportionally the computational cost. In this algorithm the number of sub-polygons and vertices is not considered a significant concern since the complexity of the final NFP generated will not have any dependence on it. The number of vertices and convex sub-polygons will only impact the computation times to produce the NFP in a pre-processing phase, which may be discarded in most practical applications (since its done only once).

The convex NFP generation algorithm uses Slope Diagrams based approach similar to the ones present in [Bennell and Oliveira, 2008] and in [Bennell and Song, 2008]. One example can be seen in Fig. 7.

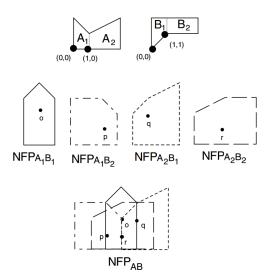


Figure 7: NFP generated by convex decomposition. ([Bennell and Oliveira, 2008])

3.2 NFP Algorithm Demonstration

The process to generate the full NFP between two different pieces A and B, which can be seen in Fig. 7, is demonstrated below. The pieces must be convex or be defined as a convex set of components. This is achieved using an algorithm for convex decomposition that will generate convex components of the original pieces. In the example of Fig. 8, piece B is already defined as a set of convex components.

The first step is to produce the NFP_{AB} components derived from all pairs of convex components between pieces A and B, as seen in Fig. 9(a), and (b). Then, intersections between pairs of NFP components can be computed, and merged into each respective NFP that generated them, as seen in Fig. 9(c).

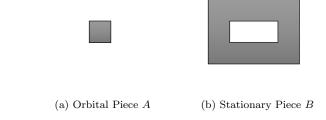


Figure 8: Pieces A and B used to generate NFP_{AB}

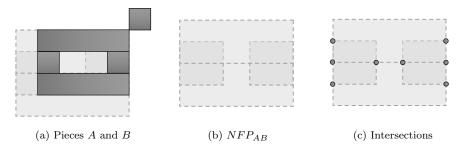


Figure 9: Generating convex NFP_{AB} intersections

Any vertices that are derived from intersections are integrated into the respective NFP outlines, producing the structure seen in Fig. 10(a), with all original vertices of the NFP components and all the intersection vertices. The following step is a division of all of the NFP component segments in half, by their midpoints, generating a new vertex that is integrated into the respective NFP components, as seen in Fig. 10(b).

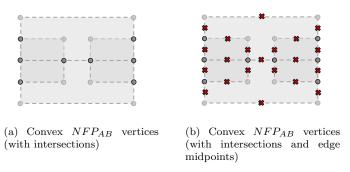


Figure 10: NFP_{AB} intersections

The convex NFP components are then converted into a directed graph, with all vertices (original, intersections and midpoints) being placed in their positions with the respective connections (edges) between each pair, in the same direction that was used to define the interior of each convex NFP component (i.e. counter-clockwise direction).

With this conversion into a graph, the similar vertices are aggregated together and their connections to other vertices are defined. In situations of multiple identical links between vertices (due to conversion into graph), only one link (or edge) is introduced into the graph.

Every vertex is then verified for containment inside each convex NFP component, and if containment is verified (and not intersecting the outline) the vertex is removed, and all edges that connect to it are deleted. This process removes all internal vertices, leaving only the vertices on the outline of the NFP components that form the full NFP, including holes, perfect sliding and perfect fit placement locations, as shown in Fig. 11(a) and (b). This step produces the full NFP, but it does not discriminate between the different components. In order to correctly identify outlines, holes and perfect sliding and fit locations,

another procedure must be done.

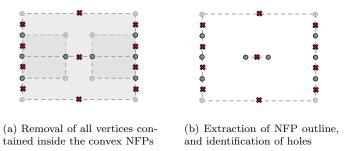


Figure 11: NFP_{AB} outlines and holes

To extract the different components, an algorithm that finds the most exterior outlines is used. This is explained below in section 3.2.1 and exemplified in section 3.2.2. It identifies each outline, hole and perfect sliding or perfect fit locations, as seen in Fig. 12, and returns them individually.

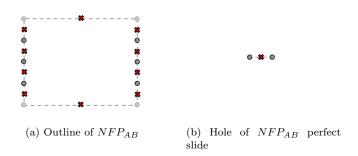


Figure 12: Generating NFP_{AB} outline and holes

The resulting graph represents the full NFP between original polygons. The last step is to convert the graph into the polygonal representation of the NFP with its holes, perfect fit and perfect sliding locations (identified by isolated vertices or sub-circuits without area). In order to identify holes or sub-circuits, one needs to use a specific method to detect exterior paths.

3.2.1 Outline Extraction Algorithm

The algorithm used to extract the outlines of the pieces starts by selecting, among the vertices of the graph G_{AB} , one or multiple vertices with the lowest coordinates in the X-axis, and among those, the single vertex with minimum y-axis coordinates. This vertex, V, is the starting point from which the outline will be extracted.

From V, all edges connected to it are compared considering their angle relative to a vertical segment connected to V. The edge(s) with the smallest angle relative to this vertical segment are selected and their direction is verified (Inward or Outward). As an example, in Fig. 13 there is only one edge connected to V with the smallest angle (S_A) given by α_A . Considering the same approach, the edge with the maximum angle relative to the vertical segment linked to V is S_1 with the angle α_1 .

The type of structure (outline, hole) can be deduced from the direction of the edge with minimum relative angle to the vertical segment. Four different cases can be presented, as shown in Fig. 14.

In the first case, in 14a, the edge (S_2) has an outward direction, which defines an outline, and not a hole. The outline starts at V and progresses through the edges until ending at V. In 14b, the edge (S_A) has an inward direction, which defines a hole that starts at V and progresses through the edges in reverse direction also ending at V.

Considering case 14c, V contains both an outline and a hole. The edge that is selected (S_2) defines an outline, which is also extracted and finishes at V from edge (S_1) . With the outline extracted, its edges are removed and the process starts over again. Vertex V is selected again, but this time, edge S_A is selected, defining a hole, which is then also extracted.

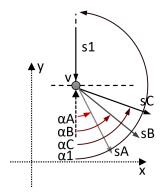


Figure 13: Selection of edge linked to V with smallest angle relative to vertical segment

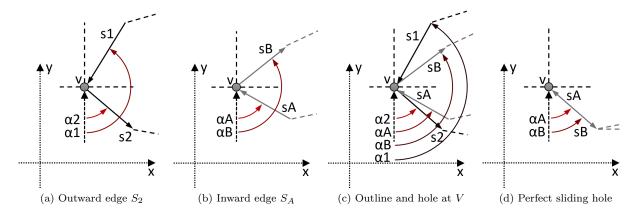


Figure 14: Edge selection for outline and hole extraction

The distinct behavior occurs in 14d, where there are 2 edges connected to V that end and start at the same vertex, due to having contrary directions, inward and outward. In this case, the structure is defined as a perfect sliding hole. If a vertex without connections is found (in the initial phase when searching for the vertex with minimum x-axis and y-axis coordinate) then this structure is defined as a perfect fit hole.

The resulting structure is the complete NFP with minimal geometric components (vertices and edges).

3.2.2 Outline Extraction Example

The outline and hole extraction process is presented in Fig 15. The NFP shown in Fig. 15a is defined by its directed edges (Fig. 15b). It starts by selecting the vertices with the lowest coordinates in the X-axis, and among those, the single vertex with minimum y-axis coordinates (shown as V in Fig. 15c). From V, the edge with smallest relative angle to a vertical segment is selected and its direction is verified (Inward or Outward). This distinction allows defining if the most exterior circuit defines a normal outline or a hole. In Fig. 15c the outward edge S_1 with minimum angle α_1 is selected and the outline is extracted (in red). The next iteration, in Fig. 15d, repeats this process by finding V and S_1 through the minimum angle α_1 , and the next circuit is extracted (in this case since S_1 is Inward, the circuit is a hole). Fig. 15e, repeats the process identifying an outline while in Fig. 15f a zero area hole is extracted.

3.3 Algorithm Description

Algorithm 1 NFP Generation Algorithm

Rebuild NFP_{AB} from G_{AB}^* using OExAlgo

*Using Outline Extraction Algorithm described in Algorithm 2

Return NFP_{AB}

20: end procedure

18: 19:

The proposed algorithm generates the full NFP using convex decomposition of the original pieces, generating the NFP from the combinations of the convex sub-polygons, and merging all of them together while correctly identifying holes, perfect slide locations and perfect fit locations in the layout.

The general steps of the NFP generation process is shown in the pseudo-code in Algorithm 1:

```
1: procedure GENNFP(A, B)
                                                                     \triangleright The NFP_{AB} of pieces A and B
       Convex decomposition for pieces A and B
                                                                     ▷ Covering or partition algorithm
2:
       Compute NFP_{AB} between convex sub-polygons
                                                                               ▶ Using slope diagrams
3:
4:
       Compute intersection points between NFP_{AB} components
                                                                         ▶ All edge pairs intersections
       Merge intersection points into respective NFP_{AB} component
                                                                         ▶ All edge pairs intersections
5:
6:
       for Each NFP_{AB} component edge do
          Compute and merge midpoint vertex
                                                                           ▷ Divide edges by midpoint
7:
8:
       end for
       Create graph G_{AB} from NFP_{AB} components
9:
                                                           ▶ Define graph by linking common vertices
      for Each NFP_{AB} component C do
10:
          for Each NFP_{AB} vertex V do
11:
             if V contained inside C then
12:
                 Eliminate V from G_{AB}
                                                                     ▶ Eliminate any contained vertex
13:
                 Eliminate edges linked to V from G_{AB}
                                                                              ▷ Eliminate linked edges
14:
              end if
15:
          end for
16:
17:
```

▷ Outlines, holes, perfect slide and fit positions

The NFP Generation Algorithm (GENNFP) starts by applying convex decomposition of both pieces A and B (in line 2) and proceeding to generate all NFP combinations between the pairs of convex components of the respective pieces A and B (line 3). These two steps are the pre-requisite for the correct execution of the algorithm. The following step (line 4) consists of computing the intersections between all generated NFPs, and integrating these vertices into the respective NFPs (line 5). The segment of each NFP is then divided by its midpoint, by integrating a dividing vertex in the middle of the segments of the NFP (line 7). In order to assist the next steps, all NFPs are converted into a graph structure, with directed edges, and shared common vertices between different NFPs (line 9). The vertices are then tested for containment inside each convex NFP, and the vertices found inside (and not on the outline) are then removed, together with all edges that connect to it ((line 13 and line 14)). The final step of

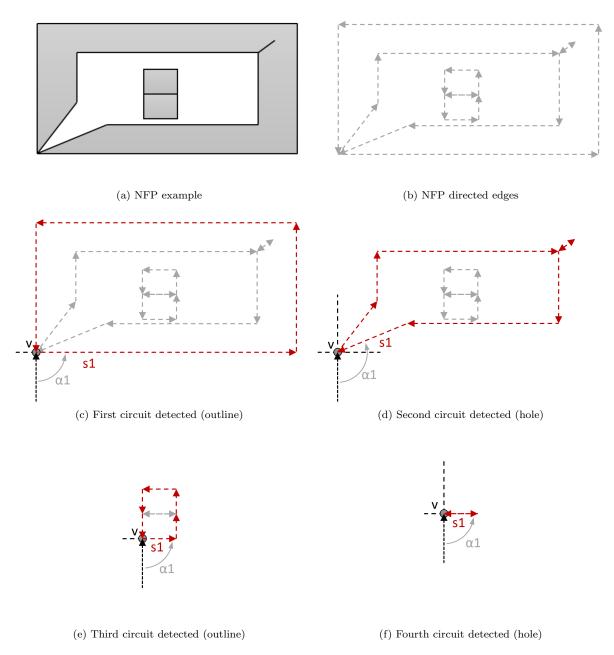


Figure 15: Outline and Hole Extraction Demonstration

the algorithm is the identification and extraction of the outlines and holes (including perfect sliding and perfect fit locations), which is then returned as the complete NFP (line 18). This is done using another algorithm that is presented in the pseudo-code in Algorithm 2.

Algorithm 2 Outline Extraction Algorithm

```
1: procedure OExALGO(G_{AB})
                                                                                    \triangleright The NFP_{AB} from G_{AB}
       Get set V_{minx} of vertices with the minimum X-axis coordinate from G_{AB}
2:
       Select vertex V with the minimum Y-axis coordinate from set V_{minx}
3:
       Select edges E linked to vertex V with minimum angle relative to a upward vertical segment
4:
       if E has one inward edge then
5:
           Extract hole (may be a perfect sliding)
                                                                               \triangleright Outward path starting at E
6:
7:
       else if E has only outward edge then
           Extract outline
                                                                               \triangleright Outward path starting at E
8:
       else if E has no connecting edges then
9:
                                                                                                      \triangleright Hole V
           Extract hole (V is a perfect fit)
10:
11:
       Return extracted objects (outlines, holes)
12:
13: end procedure
```

The outline extraction algorithm starts by selecting the vertex that has the minimum X-axis coordinate (line 2), and also the minimum y-axis coordinate (line 3), in this order of preference. This ensures that the vertex that is selected will not have any other vertex with a lower x-axis coordinate and, among similar vertices that have the same x-axis coordinate will select one (V) with the minimum y-axis coordinate. The step in (line 4) selects the edge or edges (connected to V) that have the smallest relative angle to a vertical segment linked to V. If V has two edges (with minimum angle) then it selects the edge with inward direction. If it only has one edge with inward direction then it is a normal hole. This is verified at line 5. If the edge selected has an outward direction (line 7), then an outline is extracted. If a vertex is found initially without any connections, then its defined as a perfect fit hole (line 9). During the extraction process, all vertices joining collinear edges are discarded. The objects are then returned in line 12, and the algorithm ends.

4 Validation using Literature Problem Cases

The NFPs shown in the following Fig. 16 were obtained using the current algorithm using the respective pieces and parameters as described in [Burke et al., 2007]. The piece coordinates were obtained using high detail rasterization of their figures. The NFPs were correctly generated and all degenerate cases correctly identified.

The first problem case shown in Fig. 16a deals with a pair of interlocking concavities that produce a NFP with a single hole. This case requires the detection of feasible placement positions derived from interactions of the concavities to generate holes in the NFP. The detection of holes is easier than other degenerate cases due to holes having empty regions of space with area. The larger the hole, the easier it is to detect it using different strategies due to larger range of motion possible for the placement of a piece. Detecting a perfect fit is much harder since it requires detecting a placement position without overlap between pieces. Holes may not be detected when using orbital approaches without adequate methods to detect feasible placement positions that define the holes.

In Fig. 16b is presented a problem case with multiple interlocking concavities. This case uses two similar pieces, where one is rotated 180°, and is a typical problem case for orbital approaches in the literature (as in [Mahadevan, 1984]). Due to its geometrical characteristics, when tackled by orbital approaches it requires the identification of multiple starting points created by the interactions of the concavities of both pieces. The holes have multiple different sizes, so some of the approaches may be able to detect the larger ones, but not the smallest. The current algorithm was able to successfully identify all five internal holes and the external outline.

Fig. 16c presents a problem case where perfect fit sliding positions are available. This problem is challenging due to difficulties determining feasible sliding translation movements. This requires the identification of exact placement positions, although with some range of movement across a certain direction. This is a challenge due to numerical precision and algorithm limitations.

The problem case of Fig. 16d presents an exact fit between jigsaw pieces (where the pieces are locked

together without one of them being contained inside the other and unable to move). The significant challenge of this problem case is the requirement of detecting the exact placement position without any gap or range of motion. It requires dealing adequately with any numerical precision problems that arise.

The last problem case, in Fig. 16e also deals with the challenges of detecting feasible starting positions involving holes. One of the holes of the stationary piece has multiple starting positions leading to a NFP where several distinct holes are present. The pieces used in this case are of significantly higher complexity than the ones used in previous problem cases. There is a mix of holes and interlocking concavities. If the approach used to address this case uses convex decomposition, the number of polygons generated may significantly increase its difficulty, due to the large number of NFPs produced (either by the high computational cost of the aggregate NFP set, or by the challenge of having an algorithm able to merge the convex NFPs to reconstruct the full NFP).

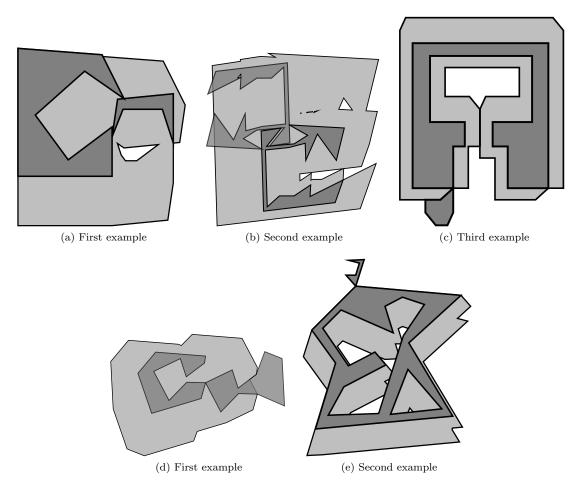


Figure 16: Examples of combinations of polygons that generate NFPs and their degenerated cases (adapted from [Burke et al., 2007])

The proposed algorithm was able to produce the full NFP for each problem case, detecting all holes and degenerate cases accurately.

5 Alternative Applications

The proposed NFP algorithm presented in section 3 was primarily developed to deal with the generation of full NFPs in two dimensions, but it can also be used to execute boolean operations between pairs of pieces, with some modifications (an example of this is shown in section 5.1). These modifications also enable the algorithm to address challenges in a third dimension (as shown in section 5.2).

5.1 Boolean Operations

The use of vertices in the middle of each edge allows the computation of boolean operations (such as the ones defined in Fig. 17) by selecting vertices to be removed, and edges that will have their direction inverted. Some of the supported boolean operations are intersections (AND), unions (OR), exclusive unions (XOR) and negation (NOT).

The boolean operation OR, seen in Fig. 17a, produces the union (or merging) between two pieces. This operation has the same process of the NFP merging algorithm described in this paper, but the zero area regions are ignored (only NFP has those structures). The union of pieces is achieved by removing any vertices that are contained (not on the outline) inside any of the pieces. An example of the vertices that have to be removed can be seen in Fig. 18a, where the mid-edge vertices (red) and the outline edges (light gray) are selected. The intersection vertices (dark gray) are not removed.

Regarding the boolean operation AND, seen in Fig. 17b, produces an intersection. This operation follows the same strategy of the boolean operation OR, but with a distinct difference in the selection of the vertices in Fig. 18a. In this situation, all vertices are removed except the intersection vertices (dark gray), and the vertices that are inside of at least one of the pieces (mid-edge as red, and outline as light gray).

Considering the operation XOR, seen in Fig. 17c, the result is achieved not by removing vertices but through inverting the edge directions between vertices. Considering the vertices presented in Fig. 18a, all edges connected to any vertex inside either one of the pieces are inverted. Any region that counts as an overlap between the two pieces is now considered a hole.

The boolean operation NOT, in Fig. 17c, defines a subtraction of one piece by another. In this case the relevant vertices to be removed are are all that are inside the piece that is substracting, the vertices that are maintained (besides the intersecting vertices) are the ones that are inside the piece being subtracted. The vertices that are to be removed are shown in Fig. 18a.

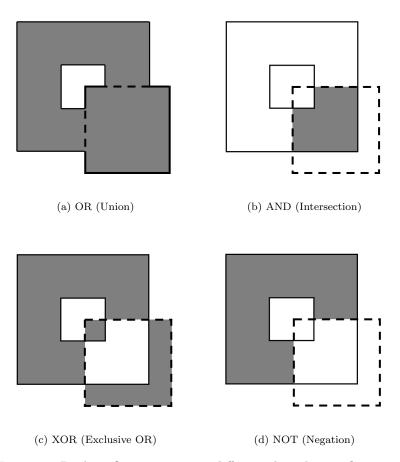


Figure 17: Boolean Operations using different algorithm configuration

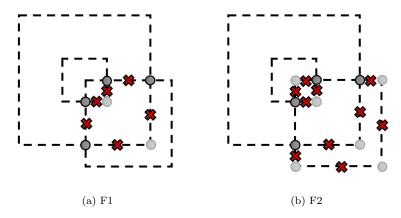


Figure 18: Relevant vertices for boolean operations

5.2 3D geometry

The proposed algorithm can also be applied to a third dimension as shown in the example in Fig. 19. In Fig. 19a the light gray region defines the overlap between both objects. Using one example of the operations for 2d structures, the application of the union (OR) strategy produces the result shown in Fig. 19b.

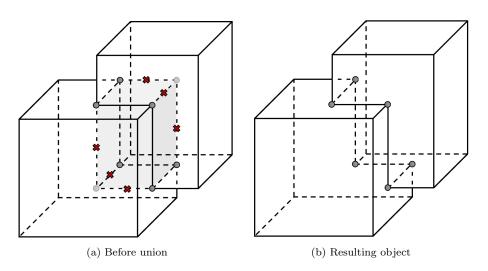


Figure 19: OR Boolean Operation (Union) in 3D

6 Conclusion

This work presents an algorithm for the generation of full NFPs that is simple to implement, and deals with any degenerate cases without effort, such as perfect fits or perfect sliding placement positions. This algorithm also reduces problems with numerical precision that may arise in other approaches (such as orbital approaches, and other iterative processes). This method also presents the benefit of being easily extended for boolean operations in both 2D and 3D, with the same advantages as when generating the NFP. The development of algorithms as the one presented in this work are an important contribution to the field, by improving the robustness and quality of current approaches, and enabling new research paths that have been mainly unexplored due to their complexity (such as 3D irregular packing).

Bibliography

- P.K. Agarwal, E. Flato, and D. Halperin. Polygon decomposition for efficient construction of minkowski sums. *Computational Geometry Theory and Applications*, 21:29–61, 2002.
- A.R. Babu and N.R. Babu. A generic approach for nesting of 2-d parts in 2-d sheets using genetic and heuristic algorithms. *Computer-Aided Design*, 33:879–891, 2001.
- J.A. Bennell and J.F. Oliveira. The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184(2):397–415, 2008.
- J.A. Bennell and J.F. Oliveira. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:93–105, 2009.
- J.A. Bennell and X. Song. A comprehensive and robust procedure for obtaining the no-fit-polygon using minkowski sums. *Computers & Operations Research*, 35(1):267–281, 2008.
- J.A. Bennell, K.A. Dowsland, and W.B. Dowsland. The irregular cutting-stock problem a new procedure for deriving the no-fit-polygon. *Computers and Operations Research*, 28:271–287, 2001.
- E. Burke, R. Hellier, G. Kendall, and G. Whitwell. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, 54:587–601, 2006.
- E.K. Burke, R.S.R. Hellier, G. Kendall, and G. Whitwell. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 179(1): 27 49, 2007. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2006.03.011. URL http://www.sciencedirect.com/science/article/pii/S0377221706001639.
- R. Cuninghame-Green. Geometry, shoemaking and the milk tray problem. New Scientist, 1677:50–53, 1989.
- H.T. Dean, Y. Tu, and J.F. Raffensperger. An improved method for calculating the no-fit-polygon. Computers & Operations Research, 33(6):1521–1539, 2006.
- P.K. Ghosh. An algebra of polygons through the notion of negative shapes. CVGIP: Image Understanding, 54(1):119–144, 1991.
- A.M. Gomes and J.F. Oliveira. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3):811–829, 2006.
- Z. Li and V. Milenkovic. Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operations Research*, 84:539–561, 1995.
- A. Mahadevan. Optimization in computer aided pattern packing. PhD thesis, North Carolina State University, 1984.
- V.J. Milenkovic, K.M. Daniels, and Z. Li. Automatic marker making. In *Proceedings of the Third Canadian Conference on Computational Geometry*, pages 243–246, 1991.
- A.K. Sato, T.C. Martins, and M.S.G. Tsuzuki. Collision free region determination by modified polygonal boolean operations. *Computer-Aided Design*, 45(7):1029–1041, 2013.
- P.D. Watson and A.M. Tobias. An efficient algorithm for the regular w1 packing of polygons in the infinite plane. *Journal of the Operational Research Society*, 50(10):1054–1062, 1999.
- G. Whitwell. Novel heuristic and metaheuristic approaches to cutting and packing. PhD thesis, University of Nottingham, 2004.