Article

An Improved Nesting Algorithm for Irregular Patterns

German Martinez-Martinez [1], Jose-Luis Sanchez-Romero [1,*], Antonio Jimeno-Morenilla[1] and Higinio Mora-Mora[1]

[1] Department of Computer Technology, University of Alicante; gmm59@alu.ua.es (G.M); jimeno@dtic.ua.es (A.J-M.); hmora@dtic.ua.es (H.M.-M)
* Correspondence: sanchez@dtic.ua.es (J.-L.S.-R).

Abstract: In industrial environments, nesting consists in cutting or extracting pieces from a material sheet, with the purpose of minimizing the surface of the sheet used. This problem is present in different types of industries, such as shipping, aeronautics, woodworking, footwear, and so on. In this work, the aim is to find an acceptable solution to solve complex nesting problems. The research developed is oriented to sacrifice accuracy for speed so as to obtain robust solutions in less computational time. To achieve this, a greedy method and a genetic algorithm have been implemented, being the latter responsible for generating a sequence for the placement of the pieces, where each piece is placed in its current optimal position with the help of a representation system for both the pieces and the material sheet.

Keywords: Nesting; cutting; irregular pattern; genetic algorithm; smart manufacturing

1. Introduction

Nesting has been a widely studied problem. In general, the problem starts with a material from which a set of parts or pieces (either 2D or 3D) must be extracted, with the purpose of minimizing the surface of the material sheet required for the extraction. Most studies focus on the problem oriented to irregular pieces, although there are also studies when the pieces are regular, this being a subtype of the problem. This problem arises in different types of industries, such as shipping, aeronautics, woodworking, footwear, etc.

Nesting process has been automated and a large number of tools has been developed so as to manage it; however, in many industries the process is still done manually, i.e. by an operator. In this case, the level of waste of the material used will depend on the experience of the operator and therefore, this human-mediated process can imply longer cutting times and possibly a higher cost due to under-utilization of material.

This problem is classified as NP-hard when the parts have arbitrary (irregular) shapes; therefore, it is impractical and computationally expensive to search for an optimal solution because there may be an enormous number of solutions. For this reason, numerous strategies, heuristic algorithms and approximations that offer acceptable solutions in relation to the result and computational time have been proposed.

In this work, the aim is to find an acceptable solution to solve complex nesting problems. The aim has been to sacrifice accuracy for speed, in order to obtain robust solutions in less computational time. To achieve this, it has been decided to use a greedy method and a genetic algorithm, being the latter responsible for generating a sequence for the placement of the pieces, where each piece is placed in its current optimal position with the help of an improved representation system for both the pieces and the material sheet.

2. Background

Nesting of 2D pieces has been categorized according to the shape of these pieces, i.e. regular or irregular. Nesting of regular pieces is usually focused on obtaining cuts of rectangular pieces, as it happens in the glass industry, while irregular cuts imply obtaining non-rectangular pieces, as it is often the case in the footwear industry. Many more strategies have been developed for solving the nesting problem in case of dealing with irregular pieces than for the case of dealing with regular ones, due to the fact that it is much more difficult to handle the edges/contours of the former. Moreover, due to the increasing relevance of irregular nesting in many industries, much work has been developed on this topic, mainly applied to textile, footwear and other industries.

The methods developed to solve this problem usually consist of three stages:

1. To develop a suitable geometric representation of the pieces, with the aim of simplifying the treatment of the irregularity of the pieces and of the sheet if necessary.
2. Develop a system for interference detection. This becomes quite tedious when the number of parts is very high and/or they have complicated shapes, making the calculations more onerous.
3. To develop some kind of heuristic that solves the problem of placing the pieces on the sheet, that is, obtaining the allocation of the pieces with their respective orientations so as to minimize the use of the sheet.

Next, some relevant research papers that develop some or all of the stages mentioned above are described.

In [1], Freeman and Shapira propose a representation for the pieces at the time of nesting (1st stage); to do this, an algorithm is developed in charge of finding the rectangle of minimum surface capable of wrapping an arbitrary closed curve, which is represented by a chain coding scheme. In this scheme, an arbitrary plane curve is composed of a sequence connected by straight line segments, which are superimposed on a finite-space square grid. Finally, the grid nodes that are closest to the intersections of the curve with the grid are connected in sequence. To find the rectangle enclosing the minimum area, the method is divided into two steps: find the convex polygon of minimum perimeter enclosing the given curve (i.e., its convex hull), and determine the rectangle of minimum surface enclosing this convex polygon. In the second stage, to find the rectangle of minimum area for a given chain-coded curve, it is only necessary to find the convex hull (formed by the points obtained above), and then try to test the set of rectangles that have a side collinear with the edges of the polygon, in order to find the rectangle with the smallest area that will contain the curve inside it.

In [2], Siasos and Vosniakos propose a method that covers all stages of the nesting problem. Regarding the process of modeling the parts, it uses a 4-way representation system. The initial part representation is an accurate 2D vector model and uses features such as straight lines, circular arcs and splines. The nesting process (called Bottom-Left-Fill-Left, BLFL) is divided into two stages: the first stage performs nesting with the parts in raster form in order to quickly locate a near optimal position on the sheet. Each piece is placed in the empty lower left part of the area large enough for the part. The second stage is responsible for making the positioning of each shape as accurate as possible. To generate the best sequence of parts, GA is used. Starting from an initial population consisting of n sets of parts (chromosomes), the GA uses the nesting procedure to sort each of the chromosomes according to the percentage of occupancy of the material sheet. The sorted chromosomes are crossed with each other to produce a new population and thus start a new generation.

In [3], Babu and Babu propose a method to tackle the whole problem by using a new representation method for both the material sheets and the parts by means of a new integer encoding. The parts are enclosed in an "imaginary" rectangle, where the surface of this rectangle is divided into a uniform grid of 1 mm x 1 mm size. Each element of the grid is called a pixel. These pixels are assigned a specific value depending on their position with respect to the part geometry. If the pixel lies within the part outline either partially or completely, the value is 0; otherwise, the pixel value is considered a positive integer value, starting from 1 for the rightmost pixel and increasing by one as you move towards the leftmost pixel of the rectangular enclosure. This process is done for all the orientations of each of the pieces to be nested. Regarding the nesting process, a method is used that places each piece in the first leftmost feasible part of the sheet. To arrange the pieces on the sheet, each piece is moved along the x-axis with an incremental movement in each and every pixel that makes up the grid on the sheet. This is done until the part is located in an acceptable position on the sheet, that is, until all pixels on the part with values of 0 exactly match the pixels on the sheet with values of 0. The authors propose to use a GA to generate the best sequence of parts and sheets with the best orientations. Each chain (chromosome) contains the parts, the sheets and the rotation angle of each part of the sequence. Each chromosome is subject to a crossover probability, which changes only the sequence of parts and sheets. The mutation process, on the other hand, alters the order of the sequence of slices and pieces and it also modifies the rotation angle and the position of the mirror image of a number of pieces of the sequence, in order to obtain the best orientation of the pieces in each case.

In [4], Prasad et al. develop a method which uses a 3D matrix so as to the information of the pieces. Each of the dimensions of the matrix represents: the number of parts, the number of features of the part, and the number of entities for each feature; entities are formed by elements such as lines, arcs, points, and so on, that are stored together with their coordinates. To perform the nesting process, a sliding algorithm called MPSR has been developed to nest pairs of parts. The algorithm can be explained as follows: given two pieces, the position and orientation of the piece with the larger number of vertices is made "fixed", while the other piece slides its edges counterclockwise, so that this piece is always in contact with the "fixed" piece, and no edge/vertex of the moving piece overlaps or crosses the edges of the "fixed" piece. The vertex with the largest y-coordinate on the "moving" part is made to coincide with the vertex with the smallest y-coordinate of the "fixed" part (this last vertex is called the sliding vertex). Next, the "guide" segment is selected, which is the segment that contains one of the vertices (either of the "fixed" or "moving" part), and along which the vertex of the moving part will be used to "slide". To generate the next sliding vertex, the "moving" part is moved along the guide segment as far as it can slide without interfering with the other part. The vertices and edges are labeled counterclockwise, and the "moving" part will also move around the "fixed" part counterclockwise. This process continues until the sliding vertex returns to the initial sliding vertex. When there are more than two pieces involved for nesting, initially the nesting will be performed for the two pieces as explained above. Then, the system will draw the boundary of the selected group of the two chosen parts and store it as a "composite" part. This "composite" part will be treated as a normal part and the above procedure will be repeated for a second part. Once the nesting of all the parts in the problem is complete, they are treated as a single part, which is reoriented from 0° to 180° to find the optimal position on the material sheet. Finally, the orientation with the best material utilization ratio is selected.

In [5], Jakobs uses the BL (Bottom Left) algorithm which is responsible for placing the piece as far to the left and down the area of the sheet, following a certain sequence formed by the identifiers of each of the pieces. For the generation of new sequences, the author uses a GA, where the fitness function measures the height occupied on the sheet. If two or more sequences have occupied the same height, the one with the smallest occupied area is chosen. The first sequence to nest at the beginning of the algorithm is the one composed by the pieces ordered according to their width in ascending order. Each sequence is submitted to a crossover operation; in contrast to the classical crossover, in this one the new sequence is obtained from two random numbers (one for each sequence) that indicate from which piece of the sequence the new one is obtained. In addition to the crossover operation, there is also the mutation, which is in charge of rotating 90º a piece of a given sequence at random. All of these procedures are repeated until a given upper limit is reached or no further improvement is achieved for a given period of time. For the case where the pieces are polygons, the algorithm would consist of three steps: embedding the polygons in rectangles, applying GA to the embedded rectangles, and bringing the polygons closer to each other (shrinking step). The first step is to determine the rectangles with minimum area for all polygons. To achieve this goal, the polygons are rotated around their center of gravity in a fixed number of equal angular increments. At each increment, the rectangle parallel to the x and y axes with minimal surface is calculated. Finally, the rectangles of minimum surface are chosen. In the second step, the genetic algorithm starts nesting the rectangles by distributing them on the material sheet. If there is no improvement over the current best sequence, it is necessary to "zoom in" the polygons further (if possible), because there may still be large gaps between them. For more precision, you can use the mirror effect on the polygons to be nested.

In [6], Han and Na use a polygon approximation and the decomposition of patterns into rectangles and circles so as to model the pieces, thus reducing the amount of processed data and execution time. Therefore, the parts can be basic components (rectangle and circle) or irregular parts, formed by a series of basic components. The decomposition is performed manually by using an interactive graphics device. The accuracy of the decomposed irregular pattern can be improved by increasing the number of basic components, but this is, however, limited by the storage capability of the computer and the complexity of the algorithm. To calculate the overlap between parts, in the case of basic components such as rectangle and circle, it is obtained by trivial comparisons of the maximum and minimum coordinates on the sheet material, which are also adopted for irregular parts. To perform the nesting of the parts in the sheet, the algorithm is divided into two stages: an initial design stage (SOAL) and an improvement stage (SA). The SOAL algorithm is based on the integration of Self-Organizing Map (SOM) and Fuzzy c-means (FCM). This algorithm produces the automatic control of the learning rate distribution. Initially, all parts are grouped around the center of the sheet by setting small random values in the position vectors of all parts. These position vectors are then modified during the self-organization process by input vectors that are uniformly distributed over the entire region of the allowed sheet. The SA algorithm offers a strategy very similar to the general iterative improvement algorithm, with one major difference: the annealing allows for perturbations. The annealing algorithm is based on the Monte Carlo technique and can be modeled mathematically using the concept of Markov chain. Rotation is allowed in 90° steps in the same way as in the initial design stage

As a summary to the research review, it can be concluded that some relevant research works that cover some (or all) stages that compose the 2D part cutting problem for different types of industries have been described. Table 1 shows the advantages and disadvantages of the methods described above.

3. Materials and methods

3.1 Problem definition

The problem to be addressed in this work could be described as follows: given a specified size of a quadrangular sheet and a set of pieces (or parts) with different shapes (regular and/or irregular), these pieces must be arranged on the sheet. The objective is to find an efficient arrangement of these pieces on the sheet, trying to optimize the sheet occupation. Only one sheet is available; therefore, if an arrangement of the pieces that fit inside the sheet is not found, nesting will not be possible. To determine the amount of material usage after nesting, the vertical line is formed at the limit of the rightmost part of the layout on the sheet. The remaining space to the right of this line represents the unused part of the material on the sheet.

3.2 Input

The parts used for nesting are contained in DXF files. The dxflib library is used so as to read these files. To draw the pieces, the LibreCAD software was chosen. In this type of applications for drawing figures, there are a series of components available such as points, lines, arcs, ellipses, etc. When reading the DXF file, this type of components are stored in a section called ENTITIES, and within this section is each of the entities used when making the figure.

The parameters of these entities are read and stored in classes; in this case, Point, Line, Circle and Ellipse. A figure is represented by a vector formed by these entities. After obtaining the vector that represents the part, the entities that form this part must be moved so as to place them in the second quadrant of the Cartesian coordinate axis. This must be done so that all the pieces are in the same "starting point", that is, the piece must be as far to the left and down as possible, but always all its points must be within the second quadrant. This allows to draw the part in any position in the coordinate system.

After moving the part to the second quadrant, it is drawn on an image file (PNG format). This image will be necessary to obtain the point on which the piece will rotate to get its orientations and to obtain its representation (subsection 3.3.1). To carry out this task, the artificial vision library OpenCV has been chosen, which provides functions to draw figures in a quick and easy way.

Table 1. Comparison of advantages and disadvantages of the different methods.

Method	Advantages	Disadvantages
Freeman and Shapira [1]	Relatively simple process to implement.	The workpiece envelope is a rectangle (lower precision). It does not take into account possible cavities in the parts.
Siasos and Vosniakos [2]	Fast focus for big problems. It takes into account the possible cavities present in the parts. Precise nesting system. Possibility to manually adjust the algorithm parameters.	Greater temporal complexity.
Babu and Babu [3]	Fast focus for big problems. It takes into account the possible cavities present in the pieces.	The nesting accuracy is lower.
Prasad, Somasundaram, and Rao [4]	Less spatial complexity by not using approximations to represent the pieces.	The designer's intervention is needed. The calculation time of the algorithm is proportional to the complexity of the geometry of the parts.
Jakobs [5]	Relatively simple process to implement.	The representation system may not be sufficiently accurate for parts with "complex" curves. It does not take into account possible cavities in the parts.
Han and Na [6]	The proposed approach saves a lot of CPU time through a two-stage focusing scheme. It allows to easily handle pieces with very varied geometric shapes. It takes into account the possible cavities present in the pieces.	More complex to implement.

Regarding the surface where the pieces will be placed, as mentioned in the definition of the problem, it is a square or rectangular shaped sheet. Therefore, its dimensions are introduced at the beginning of the process.

3.3 Spatial coding

To perform the nesting between the parts and the sheet, it is necessary that they are in the same coordinate system. In this work, a system based on a discrete representation has been chosen.

This type of representation consists of decomposing the elements into rectangles of the same size called strips. One main problem with this type of representation is that strips of a fixed size often result in a wasted free surface when one of them occupies only a small fraction of a part of the piece (right side). This situation is shown in Fig. 1. In this work, a semi-discrete representation is proposed, which is an improvement of the discrete representation. It consists of decomposing the parts and the sheet into a series of vertical strips of the same width, where the last strip can have a width smaller or equal to that of its predecessors, in order to avoid wasting free surface on the right side of the part.

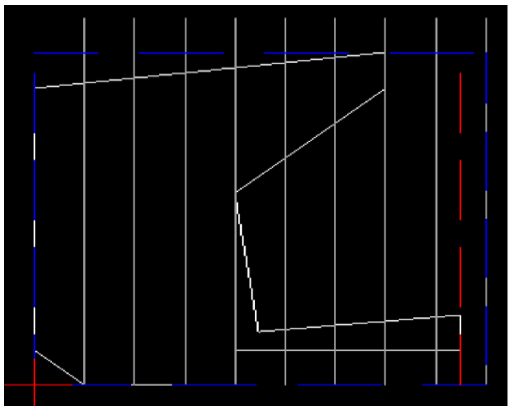


Figure 1. Visualization of the semi-discrete representation of a piece.

3.3.1 Spatial coding of pieces

It is obvious that the semi-discrete form results in a smaller enveloping rectangle for a part than the discrete form in most cases. Therefore, all parts for all their possible orientations use the semi-discrete representation, thus constructing an occupancy table for each orientation. In the mentioned table, the occupancy in each strip is designated by a range of fractional numbers as shown in Fig. 2.

1	2	3	4	5	6	7	8	9
0 ↓ 86	0 ↓ 88	0 ↓ 89	0 ↓ 91	10 ↓ 92	10 ↓ 17	10 ↓ 18	10 ↓ 19	10 ↓ 20
					66 ↓ 93	76 ↓ 95		

Figure 2. Semi-discrete representation in table format of the piece in the image above. Values are in mm.

This representation is obtained from the PNG image of the part read from the DXF file. This image is processed with OpenCV, obtaining the points that make up its outline, dividing it into strips according to the set width.

3.3.2 Spatial coding of the sheet

With respect to the sheet of material, the fractional number ranges represent the vacancy in each strip. At the start of the process, i.e. when there are no nested parts in the surface, each strip that makes up the sheet is initialized with the range between 0 and the sheet height. The last row of the vacancy table represents the width of each strip that divides the sheet.

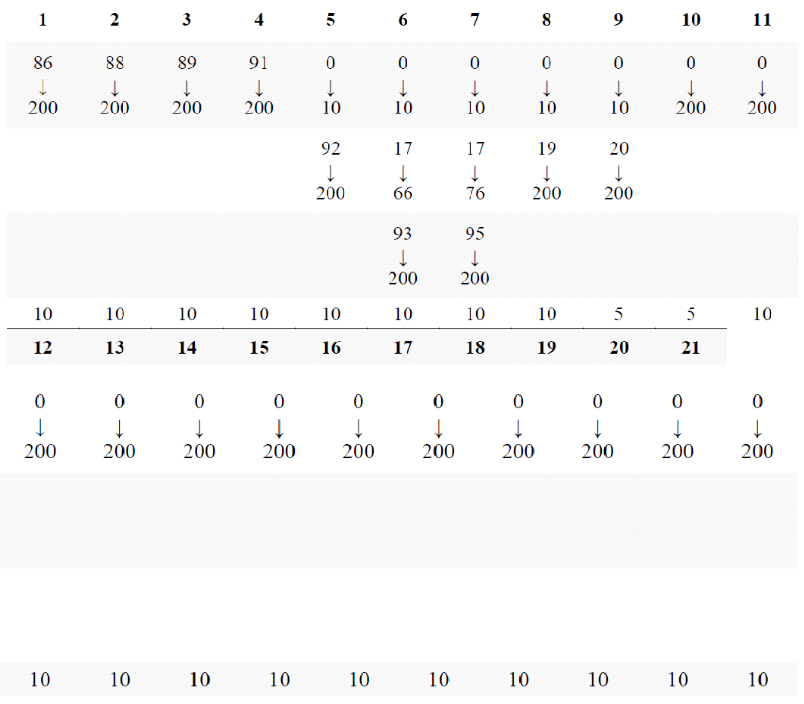
As the number of parts nested in the sheet increases, the vacancy table is updated accordingly as shown in Fig. 3. It can be observed that strips 9 and 10 of the vacancy table have a width of 5 mm; this is due to the fact that the last strip of the piece, by means of the semi discrete representation, has been simplified in this case to half the initial one which was 10 mm. This implies a greater number of strips forming the representation of the surface; therefore, the greater the number of strips, the more computation time is needed to find a feasible position for a part.

This means that a strip width that adapts properly to the parts, i.e. avoids as much as possible to perform this type of simplification to the last strip of each orientation of each part, significantly improves the computation time. Therefore, it is possible that a wider strip width (if it adapts well to the parts) obtains a better result in terms of accuracy and time than a narrower strip that adapts less well.

3.3.3 Obtaining the orientations of a part

To make a part rotate, a reference point is needed; in this work, the point on which the part will rotate is on its geometric center (centroid). This is obtained from the image generated after reading the part. Using OpenCV, the image that corresponds to the part is read and processed and, by means of the moments of the image, its centroid is obtained as a 2D point. This point is obtained for each rotation of the part, taking into account that it will always be composed of positive coordinates, since the part will always be in the second quadrant due to the translation operation mentioned in previous sections.

To obtain the centroid of each orientation, it is read and transformed to grayscale and to binary image. The next step is to obtain the contours of the binary image. The contour is filtered for accuracy. Finally, all moments are calculated up to the third order of the polygon obtained from the filtered contour. The centroid property is derived from the spatial moments of orders 0 and 1.



1	2	3	4	5	6	7	8	9	10	11
86 ↓ 200	88 ↓ 200	89 ↓ 200	91 ↓ 200	0 ↓ 10	0 ↓ 10	0 ↓ 10	0 ↓ 10	0 ↓ 10	0 ↓ 200	0 ↓ 200
				92 ↓ 200	17 ↓ 66	17 ↓ 76	19 ↓ 200	20 ↓ 200		
					93 ↓ 200	95 ↓ 200				
10	10	10	10	10	10	10	10	5	5	10

12	13	14	15	16	17	18	19	20	21
0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200	0 ↓ 200
10	10	10	10	10	10	10	10	10	10

Figure 3. Table of vacancies for the sheet surface area in the image above. Values are in mm.

The orientations of a part are formed after gradually rotating the part through a fixed angular interval between 0º and 360º indicated at the beginning of the processing. An interval of 0º or 360º makes the part have only one orientation (the initial one), while if an interval of 45º was indicated, the part would have 8 orientations. Each of the orientations will have its own vector with the same entities as the original part, but changing the value of its coordinates due to the rotation process, as well as having its own semi-discrete representation and its own centroid.

3.4 Search

Since the vacancy of the sheet is represented by one or more fractional number ranges in each of the strips, a feasible position can be determined by checking whether the vacancy segments present in the sheet material can accommodate all the occupancy segments of the part. First, it is checked if the leftmost strip of the workpiece finds a feasible position in the surface, and if so, proceed to check the rest of the strips of the workpiece. This process is accomplished by traversing the vacancy table of the sheet, starting from the bottom left corner up and to the right. Next, the cases that can occur in the search process are described:

When a vacancy segment is found to be large enough to accommodate the leftmost occupancy strip of the part, it is placed on the sheet by aligning the bottom edge of the occupancy segment to the counterpart of the vacancy segment. If there is interference on the next part strip with some of the already occupied parts of the surface, the part strips (placed so far) are slid up with the height of the interference.

If a position is unfeasible, the strips checked so far are moved to the next strip at the right (of the surface) until a feasible position is found for all of them, since it would be impossible to move the strips up in the sheet material (see Fig. 4) . If there are not enough surface strips left on the right, it means that the workpiece does not have any space to be nested in the current surface.

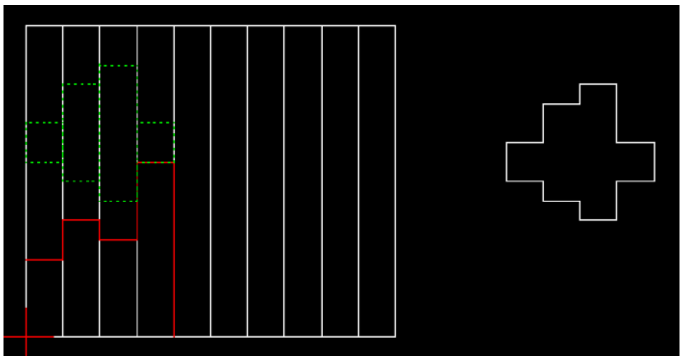


Figure 4. Repositioning of the current strips until a feasible position of the piece on the sheet.

3.5 Placement

As mentioned above, the nesting problem is NP-hard; therefore, finding an optimal solution considering all the pieces simultaneously would be computationally expensive, due to the huge number of possible solutions. In this paper, the proposal consists in using a Greedy Method that generates a sequence of parts, by means of a genetic algorithm (GA), for their subsequent placement. The objective of the proposed method is, therefore, to determine a suitable arrangement for all the pieces in an efficient manner. The method operates iteratively as shown in Fig. 5, and terminates when there is no improvement in n consecutive iterations (generations) in terms of the overall performance for the current best sequence of parts obtained so far. The different stages of the proposed algorithm are explained below.

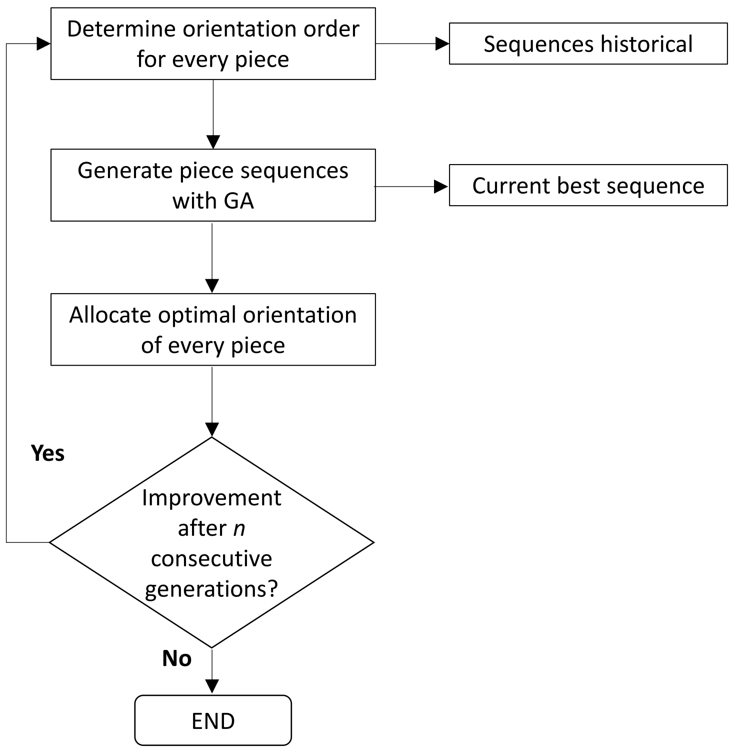

Determine orientation order for every piece
Sequences historical
Generate piece sequences with GA
Current best sequence
Allocate optimal orientation of every piece
Yes
Improvement after n consecutive generations?
No
END

Figure 5. Greedy Method flowchart.

3.5.1 Determining an orientation order for each part

The first step, performed only once at the beginning of the algorithm, is to determine an orientation order to place each part. This is achieved by constructing the enveloping rectangles formed by strips (semi-discrete representation) for all orientations of the part (explained in the subsection 3.3.1). The order of preference on the orientations is managed by the following rules:

1. Orientations with smaller width of the surrounding rectangle.

2. Orientations that have a shorter distance from their center of gravity to the bottom edge of the surrounding rectangle, i.e., the center of gravity that is closest to the base of the rectangle. This is because more free space is left at the top of the piece when it is nested in the surface and therefore makes other unplaced pieces more likely to fit better.

3. If two orientations are similar following the previous two rules, the last of these will be placed before the one already placed.

Regarding rule 2, it can happen when the use of the mirror effect on the pieces is allowed. This effect can be used when the surface of the sheet is made of the same material on both sides. The mirror effect makes the part and the counterpart of the piece have the same width on the rectangle that surrounds them. To implement this effect, a sign change has been made to all the coordinates that make up the vector of entities that represent each part, creating a new separate vector similar to the first one, but with the parts with their corresponding mirror effect. This means that, if the mirror effect option is activated, two PNG images are generated to obtain the rotation of each piece: the first one for the original piece and the second one for the piece with mirror effect. In this case, as an example, if there is a piece with a rotation interval of 45º, it will have 8 orientations, but if the mirror effect is considered, it will have 16 orientations (see Fig. 6).

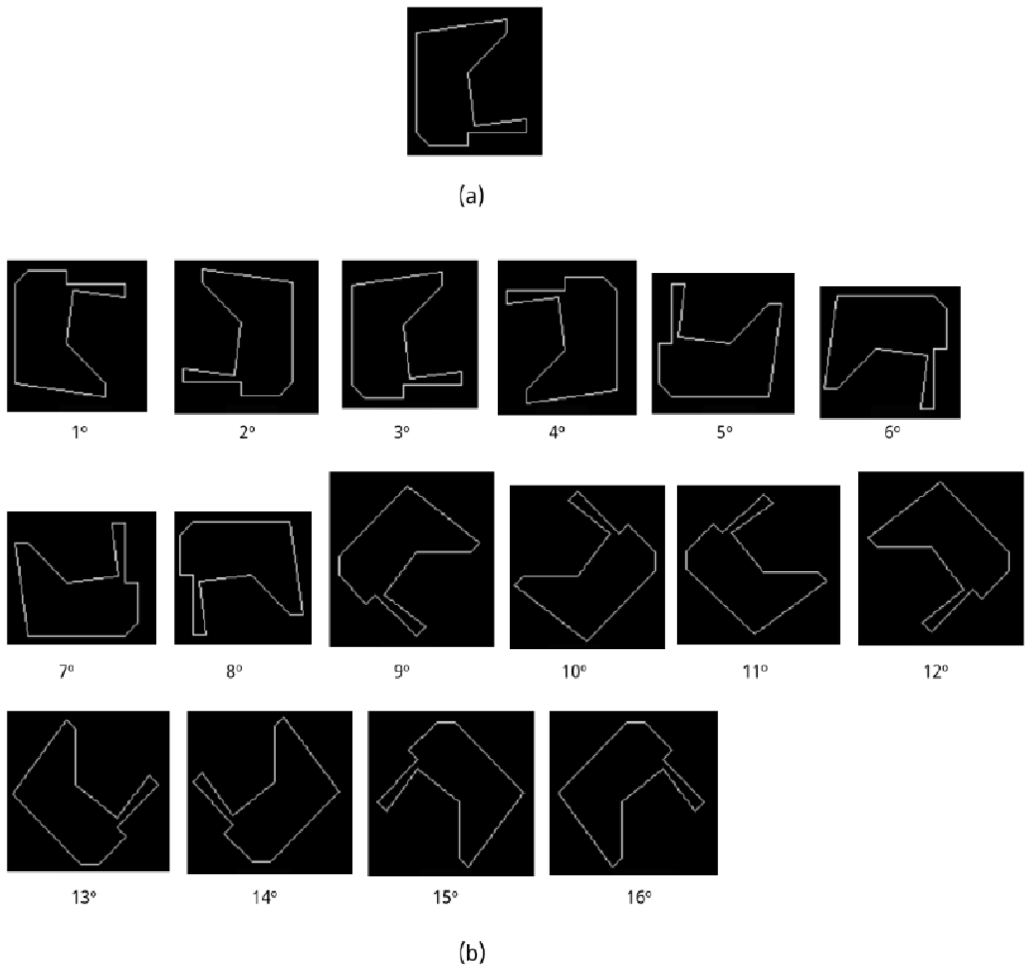


Figure 6. (a) Current part. (b) Orientation order for the rotations obtained from the current part, with a rotation interval of 45º and with mirror effect.

The purpose of constructing this orientation order is to reduce the search effort to place a part, since a feasible position of the part in the surface with a certain orientation in the proposed order automatically becomes the upper bound for the following orientations of the same part to search for a feasible position, as shown in Fig. 7. In the aforementioned figure, the first orientation becomes the upper bound for the following orientations, because no improvement in terms of performance can be achieved at or beyond that position of the surface. However, if the second orientation finds a better position than the first one, it becomes the new upper bound for the following orientations. The upper bound is the surface strip number in which the first strip of the part is framed, for the first orientation in Fig. 7, would be 5.

3.5.2 Generate sequences

For the proposed method, a mechanism that generates sequences is required in order to perform the placement process, which corresponds to the second stage of the algorithm. The GA is a suitable tool for this purpose due to its characteristics in evolutionary optimization. The GA generally encodes a possible candidate solution in several cells, which together form a linear matrix, also called chromosome or sequence, using a series of appropriate operators to achieve an evolutionary optimization. The operators involved in GA are: reproduce, crossover and mutation.

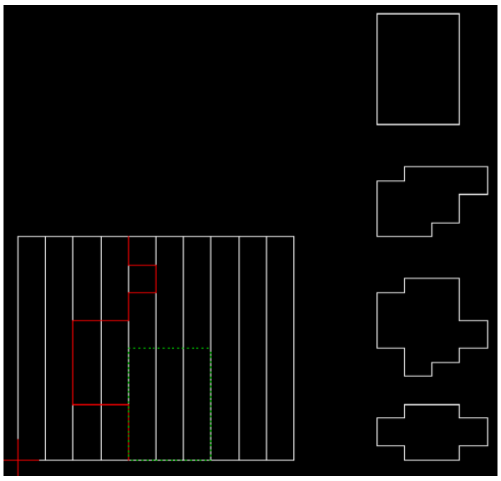


Figure 7. Setting an upper limit, using the established orientation order of a piece (right).

In this work, each cell represents a unique positive integer part number starting from 1, as shown in Fig. 8 (a). The sequence of parts is randomly generated at the beginning of the algorithm and the leftmost cell represents the first part to be placed in the successive process. Once the successive process is finished, several sequences with a worse performance score are discarded, i.e., those that have used more surface; in addition, invalid sequences are eliminated, i.e., sequences where one or several pieces that form them have not found feasible space in the surface. On the contrary, the sequences with a better performance are applied the crossover and mutation methods, together with the new randomly generated sequences that replace those previously eliminated for being worse or invalid.

Since the determination of the parts sequence is a permutation problem, the traditional crossover operation does not work for it, because the traditional one performs a cell crossover between two chromosomes; therefore, this process might give a new sequence that does not contain all the parts. Instead, the single chromosome crossover operation is adopted, where the left subsequence of the chromosome is exchanged with the opposite counterpart of the right one from a randomly chosen intermediate point, as shown in Fig. 8 (b). This operation is applied to all sequences.

The mutation operation in this implementation is responsible for swapping one of the cells of the randomly chosen sequence with one of its neighbors, as shown in Fig. 8 (c). This operation is performed on a random number of sequences, that is, at least on one of the current sequences and at most on all of them. Mutation guarantees that no point in the search space has a zero probability of being examined.

In addition to the modifications made by the crossover and mutation operators, two recordings are made in order to speed up the whole process and thus prevent the sequence with the best performance score from fluctuating during the process. The first recording is called "historical", in which all the sequences generated by the GA so far are recorded. This is done so that a sequence is not re-evaluated more than once, as it is not necessary and would add more computation time to the process. The second recording is responsible for storing the best sequence generated so far, that is, the one with the best performance. When the best sequence (new best) in the successive generations outperforms the current best, that is, occupies less surface, the current best is replaced by the new best.

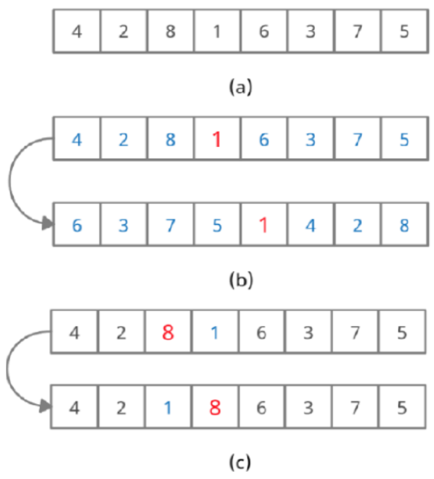


Figure 8. Generation of a sequence of parts by GA. (a) Chromosome encoding. (b) Crossover operation. (c) Mutation operation.

3.5.3 Searching for the optimum position and orientation of a piece in the sequence

After several sequences of parts are generated by the GA, the part placement process is started to determine the performance for each of the sequences. To reduce the search effort in selecting one orientation or another for a part to be nested, the upper bound, mentioned in the subsection 3.5.1, is used. As it can be seen in Fig. 6, the orientations are ordered according to the established criteria, and these mark the upper limit for the following orientations of the same part. When looking for feasible space for the first strip of one of the orientations of the part, if the upper limit imposed by an orientation with more preference over the one currently evaluated is reached or exceeded, it is no longer necessary to continue looking for a feasible position for the strip, since, even if a position were found, this orientation will have a worse performance with respect to the one that marked the upper limit.

This search method makes it necessary to check all orientations together with all their strips in the worst case for a part, while in the best case it is only necessary to check one orientation completely, and the rest only its leftmost strip.

3.6 Output Representation

Once the best sequence has been obtained after n generations, if one has been obtained, the nested sheet with all the selected pieces in a vacancy table format is achieved. The last step is to translate this representation to one where the user will appreciate the result in a more visual way.

The representation in a DXF type file has been chosen and the dxflib library has been used to write this type of files. The first element of the tuple represents the original part identifier, the second element is the original part rotation identifier, and the third element represents the starting point to draw the corresponding orientation in the output DXF file. This structure allows to access each entity vector of the respective orientation of each part, and start drawing the part orientation from the indicated point with the tools provided by the dxflib library.

4. Experimentation

In order to validate the proposed strategy, different regular and irregular patterns have been used, together with different values for the set of variables involved in the proposed method. These variables are:

- Number of generations.
- Number of sequences in each generation.
- Width of the strips forming the semi-discrete representation of the pieces and sheet.
- Rotation angle for the pieces.

All the case studies shown below have been performed in a test environment based on an Intel(R) Core(TM) i5-7200U at 2.50GHz. In addition, for the sequence generation process using the GA, 10 chromosomes (sequences) have been chosen. The search process was stopped when the performance had no improvement after 30 consecutive iterations (generations).

4.1 Case study 1

In the first case, the location of 28 irregular pieces, generated in the following way, is analyzed. The different patterns were arranged on a material of 80x50 cm. Fig. 9 shows the patterns selected for this experiment.

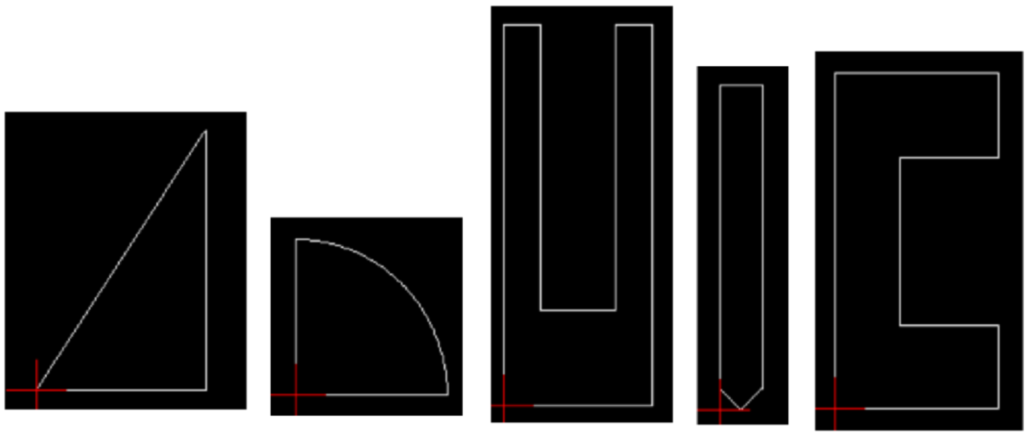


Figure 9. Shapes used for case study 1.

For this case, the orientation order for each part was constructed using an interval of 45º rotation, along with various values for the width of the strips forming the semi-discrete representation. Table 2 shows the comparison when running the nesting algorithm with different strip widths.

As expected, the execution time increases as the strips that form the semi-discrete representation become narrower, basically because when searching for a feasible position, a larger number of strips must be computed. On the other hand, a smaller strip width does not imply a higher nesting accuracy, because it is possible that wider strips adapt better to the pieces; besides having fewer strips to go through when searching for a feasible position in the surface of the sheet, this can be seen in the results with a strip width of 1 cm and 2 cm, obtaining results of 488 mm and 480 mm, respectively. The results are shown below in DXF and graphical format.

Table 2. Results obtained from Case study 1.

Strip width	**1**	**2**	**3**	**4**
Ocuppied surface (mm)	488	480	534	550
Computation time (s)	4.303	0.849	0.365	0.343

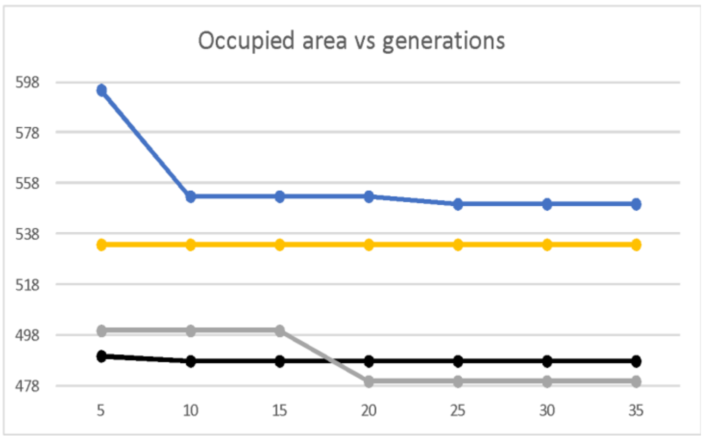


Figure 10. Occupied surface vs number of generations for different strip width (black = 1 cm; grey = 2 cm; yellow = 3 cm; blue = 4 cm).

As it can be seen in Fig. 10, in the first 10 to 20 generations an improvement in the performance is achieved throughout the process, with the exception of the third figure where performance remains stable during the 30 iterations. Looking at the results, it can be conclude that, with a smaller number of generations, similar results could be obtained in less computation time. Final result is shown in Fig. 11.

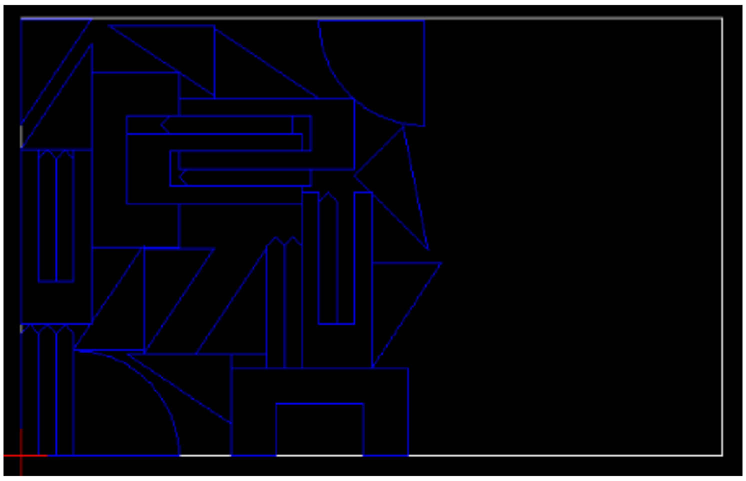


Figure 11. Final result after the nesting process of case study 1 with a strip width of 2 cm.

4.2 Case study 2

In this second case, another set of 31 irregular pieces have been used to be arranged in an 80x60 cm sheet. Fig. 12 shows the pieces used.

In this case, the rotation interval for each piece has been 30º, together with different values for the strip width. Table 3 shows the comparison of the algorithm executions. As in the previous case, there is a decrease in computation time with a greater strip width, without relation with a greater precision in the nesting of the pieces in the sheet. It can also be observed that the computation time is higher than the previous case, because there are a greater number of pieces to be nested and, above all, because there are a greater number of orientations per piece; in this case there are 12 orientations, while in the previous example there were 8. The results are shown below in DXF and graphical format.

Fig. 13 shows, as in the previous case, that fewer generations provide similar results, even in a shorter time, with the exception of the third graph, where the performance improvement occurs between generation 20 and 30; therefore, a reduction in generations could result in a worse performance than the current one. Results are shown in Fig. 14.

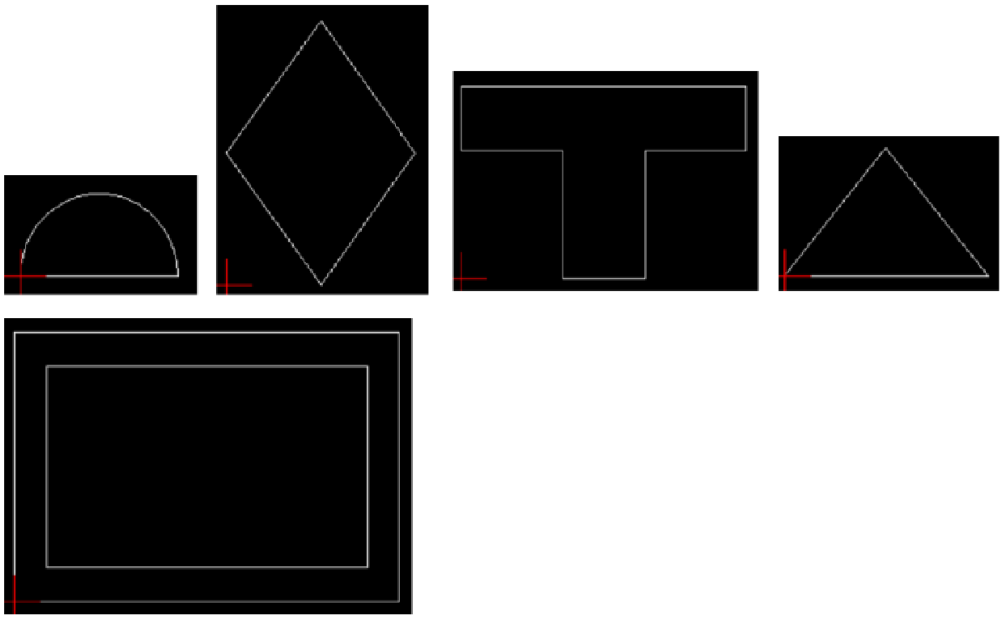


Figure 12. Shapes used for case study 2.

Table 3. Results obtained from case study 2.

Strip width (cm)	1	2	3	4
Ocuppied surface (mm)	570	620	600	720
Computation time (s)	10.8	1.343	0.73	0.334

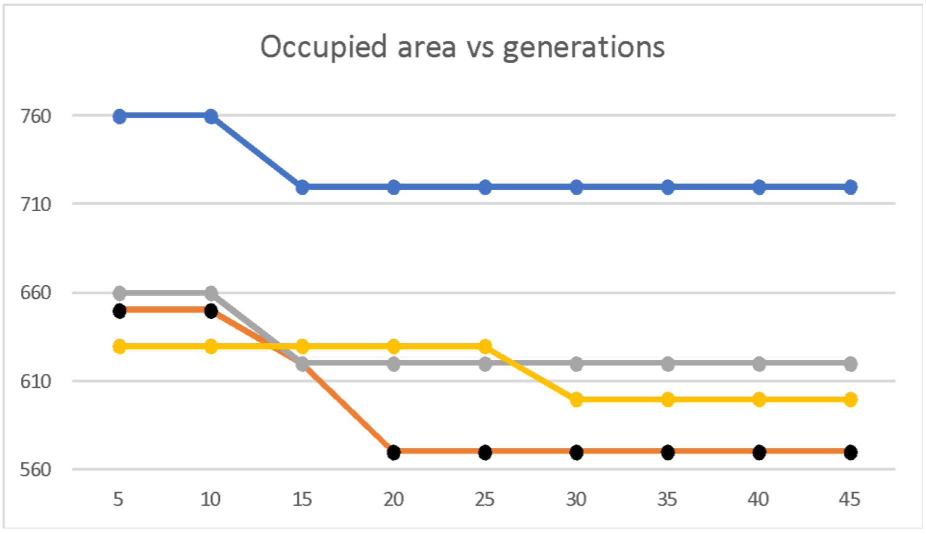


Figure 13. Occupied surface vs number of generations for different strip width (brown = 1 cm; grey = 2 cm; yellow = 3 cm; blue = 4 cm).

4.3 Case study 3

In this third case, 35 irregular pieces of 4 different patterns have been used, which have been arranged on an 80x60 cm sheet. Fig. 15 shows the pieces used.

In this case the rotation interval for each piece has been 1º, with the values used in the two previous cases for the width of the strips. Table 4 shows the results obtained.

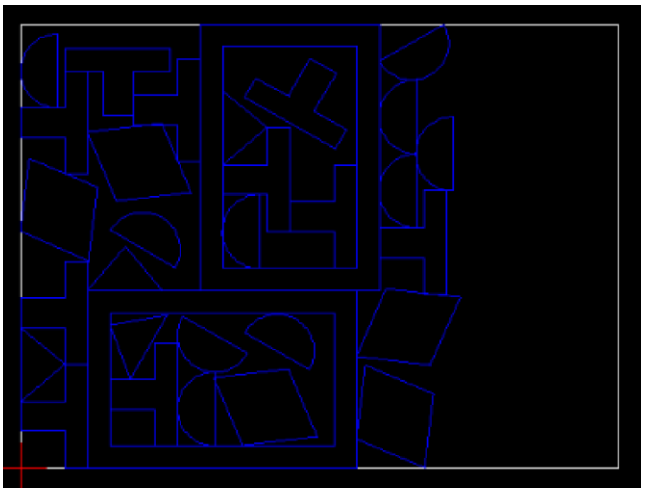


Figure 14. Final result after the nesting process of case study 2 with a strip width of 3 cm.

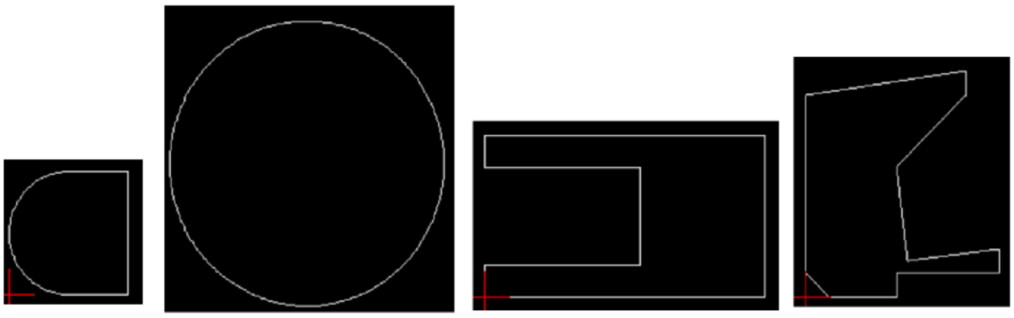


Figure 15. Shapes used for case study 3.

Table 4. Results obtained from case study 3.

Strip width (cm)	1	2	3	4
Ocuppied surface (mm)	440	440	540	480
Computation time (s)	277.69	16.04	7.525	2.658

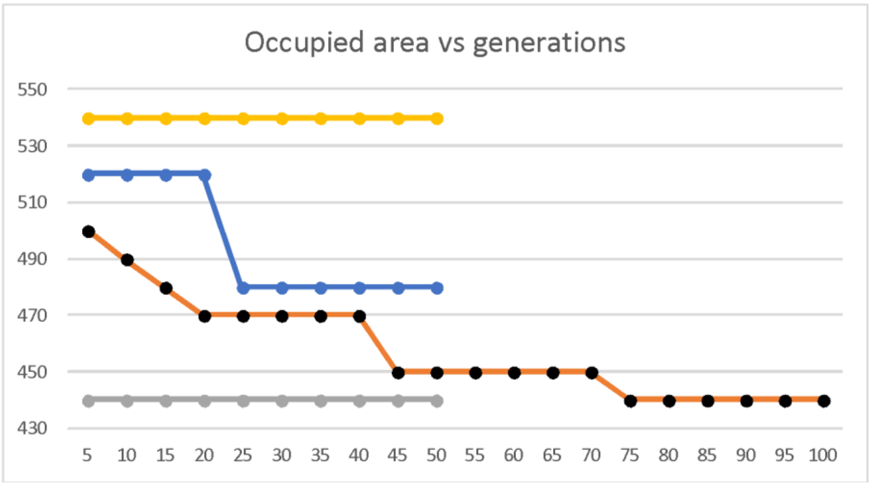


Figure 16. Occupied surface vs number of generations for different strip width (grey = 1 cm; brown = 2 cm; yellow = 3 cm; blue = 4 cm).

As it can be observed, the computation times are much longer than in the previous cases, mainly due to the fact that the different orientations per piece have changed from

12 to 360, since the rotation interval is 1°. Something to highlight is the quite wide difference in the execution time of the case with a strip width of 1 cm compared to the rest. This is mainly due to the fact that in the case of 1 cm, 100 generations have been executed, while in the others, at most 50 generations have been run. In this case, the options with a strip width of 2 cm and 4 cm offer better results in relation to precision and computation time. The results are shown below in DXF and graphical format.

Unlike the previous graphs, Fig. 16 shows a decrease in the number of generations would achieve a reduction in computation times, but with a high probability of achieving worse performance than the current ones. A better alternative would be to decrease the sequences per generation which could obtain a better time/performance ratio than the current one. Results are shown in Fig. 17.

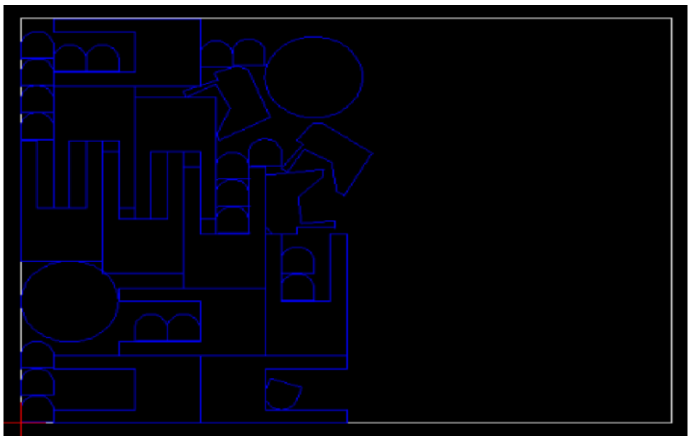


Figure 17. Final result after the nesting process of case study 3 with a strip width of 2 cm.

4.4 Case study 4

For this case, 50 units of a single piece have been used, together with different values for the rotation interval of the piece, which must be distributed in a 140x80 cm sheet. Since there is only one type of piece, it is not necessary the generation of sequences by the GA, because the order for nesting will always be the same. Fig. 18 shows the piece used.

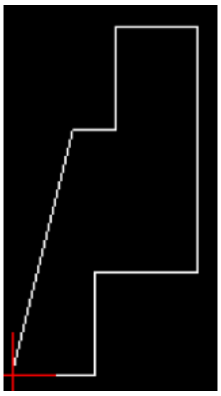


Figure 18. Shape used for case study 4.

In this case a strip width of 2 cm has been used, together with different values of rotation interval. Table 5 shows the results after running the nesting algorithm with different values for the rotation interval.

Table 5. Results obtained from case study 4.

Rotation step (deg)	1	5	15	35	45	55

Ocuppied surface (mm)	980	1040	1140	1100	980	1180
Computation time (s)	0.203	0.054	0.035	0.032	0.031	0.031

As it can be analyzed from the results, a smaller rotation step always requires a longer calculation time, but this does not necessarily lead to a better use of the material surface.

This can be observed in the rotation steps of 1º and 45º; in both the same use of the surface of the sheet is obtained, but with a lower time with the rotation interval of 45º with respect to 1º, giving a difference of 0.172 s between both.

As mentioned above, since there is only one type of part in the nesting process, it is not necessary to generate sequences, therefore, it is not possible to obtain a graph as in the previous cases. Results are shown in Fig. 19.

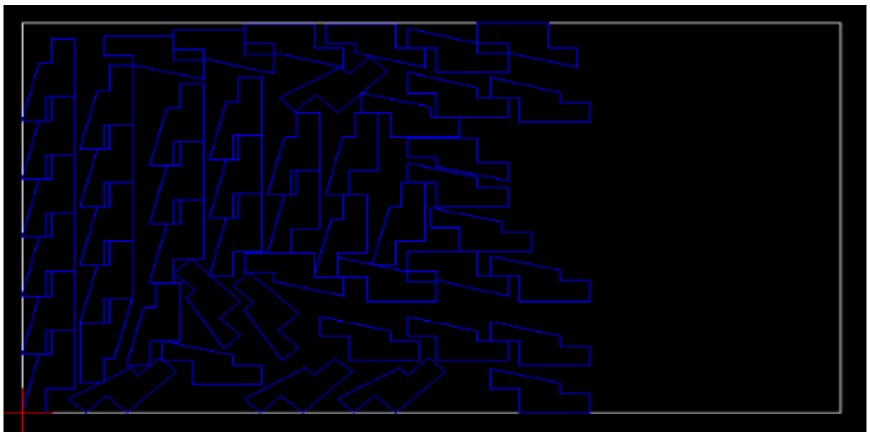


Figure 19. Final result when performing the nesting of case study 4 with a rotation step of 45º.

4.5 Performance comparison with a commercial tool

In this section, the performance shown by the proposed method is compared with one of the online cutting applications available at CNC-APPS [7], which uses a proprietary solution. To make the comparison, the cases previously shown are used.

The performance comparison is mainly focused on the used surface and the total processing time. The configuration of the other parameters is set with the same values as for each case seen above, except for the strip width, as this option is not available in the online tool. Table 6 shows the comparison about performance and computation time of the proposed method and the commercial tool. Something to highlight is that case study 2 could not be compared, because the online tool does not seem to nest correctly pieces with holes inside (case study 2 has a piece of this type).

As can be seen in the Table 6, in case studies 1 and 3 the proposed method obtains better results in relation to nesting accuracy and computation time than the ones provided by the benchmark, except for case 3 with strip width 1, whose computation time is much higher, mainly due to the intense use of iterations for this particular case, as mentioned above. In case 4, it can be seen in Table 7 that the proposed method obtains a worse use of the sheet surface, although not much higher; however, it achieves much shorter execution times, mainly due to the fact that, being a single type of piece, the generation of sequences by the GA is not necessary and, therefore, the algorithm becomes simpler and the computation time decreases.

Throughout this paper, it has been mentioned that the aim of the proposed method was to determine an acceptable arrangement of parts, i.e., to sacrifice accuracy for efficiency. After analyzing the results, it can be concluded that the goal of achieving a good relationship precision/efficiency is fulfilled, since good results are achieved with respect to the use of the surface and quite acceptable computation times even better than expected in many cases.

It can also be concluded that the proposed method is competitive with respect to the CNC-APPS tool, obtaining quite good results in computation time (generally better); in addition, the proposed method allows to use parts with holes inside, while the online tool apparently does not work properly with this option.

Table 6. Comparison of results of the proposed method with the benchmark, case studies 1 and 3.

	Case 1				**Case 3**			
Strip width (cm)	1	2	3	4	1	2	3	4
Surface occupied by benchmark (mm)	600				596			
Surface occupied by proposed method (mm)	480	480	534	550	440	440	540	480
Computation time of benchmark (s)	22				38			
Computation time of proposed method (s)	4.303	0.849	0.365	0.343	277.69	16.04	7.525	2.658

Table 7. Comparison of results of the proposed method with the benchmark, case study 4.

Rotation step (deg)	**1**	**5**	**15**	**35**	**45**	**55**
Surface occupied by benchmark (mm)	980	990	995	1100	965	1050
Surface occupied by proposed method (mm)	980	1040	1140	1100	980	1180
Computation time of benchmark (s)	42	45	42	41	35	36
Computation time of proposed method (s)	0.203	0.054	0.035	0.032	0.031	0.031

5. Discussion

After analyzing the different experimental cases, it can be concluded that the proposed method, together with adequate values for the different parameters, achieves quite competitive results in relation to performance and computation time compared to the CNC-APPS tool. It must be taken into account that, for most industrial practices, the nesting process is generally applied to cases where the number of parts is relatively large. When the sum of the surfaces of all the parts is fairly close to the surface of the sheet, it is necessary to verify all those parts can be nested on the sheet. If this is accomplished, it becomes important to locate the parts to minimize material waste, especially when the sheet material has a high cost. Therefore, it is insignificant to discuss cases where there are only a small number of parts to nest in a relatively large sheet, as this can easily be achieved by human intervention.

In short, the proposed method meets the expectations, obtaining comparable results to the CNC-APPS tool.

6. Conclusions

This paper presents an efficient and effective approach to the nesting or packaging problem that includes a greedy method, which was used to find the optimal solution for each part by following the order in a sequence, and a proposal for a new, better spatial coding scheme, called semi-discrete representation. Furthermore, a Genetic Algorithm is applied to generate the sequence of parts and an iterative process is proposed to increase the overall performance within a tolerable computation time.

The proposed approach only searches a small fraction of the huge solution space (NP-hard problem), where each piece of the sequence is always placed at its optimal position according to the upper bound rule. Experimental results show that the improvement in performance becomes negligible as more generations are involved.

The efficiency of the proposed approach is partially achieved by the upper bounding technique, which is based on the assumption that each part in the sequence is placed at its current optimal position; this also allows the iterative process to become faster, as there is no need to consider all the strips that make up the different orientations of each part, checking at best the first orientation of a part and the leftmost strip of the remaining orientations. Secondly, it is worthwhile mentioning the semi-discrete representation, which makes the updating of the state of the material sheet easier, contributing therefore to the efficiency of the proposed approach. In terms of effectiveness, the proposed method tries to optimize each part placement by following the order of a sequence rather than achieving rigorous optimization. Experimental results show that the proposed approach achieves good performance for all the adopted examples.

Next, some lines of future work that can be carried out on the method developed in this paper are exposed:

- Elimination of redundant orientations: in those parts with geometric symmetry, some of the rotation angles used to obtain the different orientations become redundant and can be omitted, thus reducing the number of orientations to be checked at the time of the part placement process. For example, it is sufficient to consider rotation angles from 0º to 180º for a rectangular part, decreasing the number of orientations from 8 to only 4.
- Application of an anti-aliasing filter: when obtaining the semi-discrete representation of each of the orientations of the parts, they are drawn using OpenCV in a PNG file. Therefore, if there is a staggered effect on one or more lines that make up the part, the occupancy table that represents this orientation may become less accurate than it should be. The solution would be to apply an anti-aliasing effect when drawing each PNG image.
- Apply a minimum gap between pieces: when cutting the pieces on the surface of the sheet material, the machine may have some error when making this cut. A solution can consist in applying a previous gap, introduced at the beginning of the process.

References

1. Freeman, H.; Shapira, R. Determining the minimum-area encasing rectangle for an arbitrary closed curve. Communications of the ACM, 1975, 18(7), 409-413.
2. Siasos, A.; Vosniakos, G. C. Optimal directional nesting of planar profiles on fabric bands for composites manufacturing. CIRP Journal of Manufacturing Science and Technology, 2014, 7(3), 283-297.
3. Babu, A. R.; Babu, N. R. A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. Computer-Aided Design, 2001, 33(12), 879-891.
4. Prasad, Y. K. D. V.; Somasundaram, S.; Rao, K. P. A sliding algorithm for optimal nesting of arbitrarily shaped sheet metal blanks. THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH, 1995, 33(6), 1505-1520.
5. Jakobs, S. On genetic algorithms for the packing of polygons. European journal of operational research, 1996, 88(1), 165-181.
6. Han, G. C.; Na, S. J. Two-stage approach for nesting in two-dimensional cutting problems using neural network and simulated annealing. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 1996, 210(6), 509-519.
7. CNC-APPS. Available online: https://cnc-apps.com/en/ (accessed on 18 October 2021).

Author Contributions: Conceptualization, G.M. and J.-L.S.-R..; methodology, G.M. and A.J-M.; software, G.M. and J.-L.S.R..; validation, G.M., A.J-M, and H.M.-M.; investigation, G.M. and J.-L.S.-R.; data curation, G.M. and H.M.-M.; writing—original draft preparation, G.M. and J.-L.S.-R..; writing—review and editing, G.M. and A.J.-M.; project administration, J.-L.S.-R., and A.J.-M.; funding acquisition, A.J.-M. and H.M.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Spanish Ministry of Science and Innovation, grant number PID2020-120213RB-I00.

Conflicts of Interest: The authors declare no conflict of interest.