Pesquisa Operacional (2023) 43: e275212 p.1-26
doi: 10.1590/0101-7438.2023.043.00275212
© 2023 Brazilian Operations Research Society
Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
ARTICLES

A BIOBJECTIVE MATHEURISTIC FOR THE INTEGRATED SOLUTION OF THE IRREGULAR STRIP PACKING AND THE CUTTING PATH DETERMINATION PROBLEMS

Larissa Tebaldi Oliveira[1,2*], Maria Antónia Carravilla[3], José Fernando Oliveira[4] and Franklina Maria Bragion Toledo[5]

Received May 29, 2023 / Accepted July 27, 2023

ABSTRACT. Irregular strip packing problems are present in a wide variety of industrial sectors, such as the garment, footwear, furniture and metal industry. The goal is to find a layout in which an object will be cut into small pieces with minimum raw-material waste. Once a layout is obtained, it is necessary to determine the path that the cutting tool has to follow to cut the pieces from the layout. In the latter, the goal is to minimize the cutting distance (or time). Although industries frequently use this solution sequence, the trade-off between the packing and the cutting path problems can significantly impact the production cost and productivity. A layout with minimum raw-material waste, obtained through the packing problem resolution, can imply a longer cutting path compared to another layout with more material waste but a shorter cutting path, obtained through an integrated strategy. Layouts with shorter cutting path are worthy of consideration because they may improve the cutting process productivity. In this paper, both problems are solved together using a biobjective matheuristic based on the Biased Random-Key Genetic Algorithm. Our approach uses this algorithm to select a subset of the no-fit polygons edges to feed the mathematical model, which will compute the layout waste and cutting path length. Solving both strip packing and cutting path problems simultaneously allows the decision-maker to analyze the compromise between the material waste and the cutting path distance. As expected, the computational results showed the trade-off's relevance between these problems and presented a set of solutions for each instance solved.

Keywords: cutting and packing problem, nesting problem, cutting path determination problem, biobjective matheuristic, BRKGA.

*Corresponding author
[1]School of Sciences, São Paulo State University (UNESP), Campus Bauru, Av. Eng. Luís Edmundo Carrijo Coube, 14-01, 17033-360, Bauru, SP, Brazil – E-mail: larissa.tebaldi@unesp.br – https://orcid.org/0000-0002-4081-3765
[2]Institute of Mathematical and Computer Sciences, University of São Paulo (USP), Avenida Trabalhador São-carlense, 13560-970, São Carlos, SP, Brazil
[3]INESC TEC, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal – E-mail: mac@fe.up.pt – https://orcid.org/0000-0002-9245-2674
[4]INESC TEC, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal – E-mail: jfo@fe.up.pt – https://orcid.org/0000-0002-4061-1311
[5]Institute of Mathematical and Computer Sciences, University of São Paulo (USP), Avenida Trabalhador São-carlense, 13560-970, São Carlos, SP, Brazil – E-mail: fran@icmc.usp.br – https://orcid.org/0000-0003-2823-7600

1 INTRODUCTION

A wide range of products, such as shoes, clothing, and furniture, are manufactured by assembling several small pieces that are sometimes cut from the same material. Anand & Udhayakumar (2019) present an example of metal sheet parts used in ship manufacturing. To produce these small pieces, large objects are usually cut into small parts. Determining where each piece will be cut from the object, i.e., the cutting layout, is an important part of the production process. Finding the best layout, i.e., the layout with minimum material waste is an optimization problem known in the literature as the Cutting and Packing Problem. When the pieces have irregular shapes, as in the garment industry, the problem is known as the Nesting Problem. Several mathematical models and methods have been proposed to deal with nesting problems (for review see Bennell & Oliveira (2009), Álvarez-Valdés et al. (2018) and Leão et al. (2020)).

As soon as the layout is established, the problem of finding the best way to cut the pieces arises. Known as the Cutting Path Determination Problem (CPDP), the goal is to find the shortest cutting path that the cutting tool has to follow to cut the layout previously defined. The shortest path is critical when the cutting tool material is expensive. Besides this, the time consumed in the cutting process strongly influences the companies' productivity. In the literature, most of the CPDP is solved with heuristic methods (Manber & Israni (1984), Han & Na (1999), Lee & Kwon (2006), Moreira et al. (2007), Imahori et al. (2008) and Rodrigues & Ferreira (2011)), whereas few papers present exact approaches (Dewil et al. (2011), Dewil et al. (2014) and Silva et al. (2019)).

Manber & Israni (1984) and Imahori et al. (2008) affirm that the choices made during the layout phase strongly influence the minimum cutting path determination but, in practice, these two problems are solved hierarchically, i.e., first, a layout is defined, and then a cutting path is established (Pott & Glaab (2003) and Sherif et al. (2014)). Anand & Babu (2015) proposed a method to solve the packing problem and the cutting path by considering rectangular pieces. The cutting machine consumes material during the cutting process, therefore the objective is to minimize the material waste and the cutting path distance by forming common cutting edges between adjacent pieces. Oliveira et al. (2020) proposed the first mathematical model that integrates the nesting and the cutting path determination problems. The authors considered the discrete positioning of the pieces and the cut by pieces, which will be explained later in this paper. However, the model was able to optimally solve only small instances.

In this paper, both nesting and cutting path determination problems are solved simultaneously using a Biobjective Biased Random-Key Genetic Algorithm based matheuristic (BRKGA-BiMath). Our objective is to support the decision-makers with a pool of trade-off solutions, allowing them to choose the best layout and cutting path set for each possible situation faced during the production process. It is important to highlight that BRKGA has been successfully used for different variants of cutting and packing problems (Mundim et al. (2016), Mundim et al. (2017), and Amaro-Júnior et al. (2017)).

The BRKGA-BiMath is the first heuristic proposed to solve the nesting and cutting path determination problems together. It is a population method in which each solution is evaluated using two linear models. In this approach, the BRKGA is used to select variables which will reduce the complexity of the nesting model, converting it into a linear model that is easier to solve. After obtaining a layout, a second mathematical model minimizes its total cutting path distance. These two values (layout length and cutting path distance) become one fitness value by using the fast non-dominated sorting and the crowding distance algorithms proposed by Deb et al. (2002). During the BRKGA-BiMath evolution, the best solutions for both problems have more chances to be maintained within the population. The computational results showed the trade-off between the layout length and cutting path distance, i.e., when the layout length increases, the cutting path distance decreases, and vice-versa.

The paper is organized as follows. In Section 2, the studied problem is defined. The BRKGA-BiMath is detailed in Section 3, while the computational experiments are reported in Section 4. In Section 5, the conclusions and an outlook on future research directions are presented.

2 PROBLEM DEFINITION

The Nesting and Cutting Path Problem (NCPP) addressed in this article is defined from the nesting and the cutting path determination problems underlying it. Therefore, in this section, we define both problems and their relevant characteristics.

2.1 Nesting problem

The irregular strip packing problem, also known as the nesting problem, is a problem within the broader class of Cutting and Packing problems, in which a set of irregularly shaped pieces, which can be convex or non-convex, must be cut from an object of fixed width and unlimited length. Usually, the goal is to find a layout with minimum material waste. An example of a layout with convex and non-convex pieces is illustrated in Figure 1.

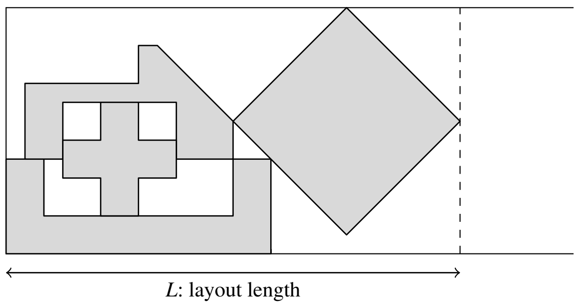
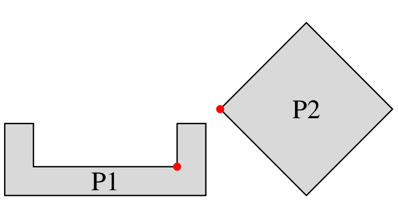


Figure 1 – Example of a layout with four pieces.

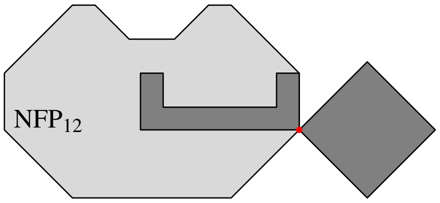
The main difficulties when dealing with nesting problems are ensuring the geometric feasibility, i.e., guarantee that (i) every two pieces do not overlap each other; and (ii) the pieces are entirely inside the object. The most common strategies used to deal with these geometric issues are the raster points, the direct trigonometry, the no-fit polygon (NFP), and the phi-function. For more details about these strategies, the reader is referred to Bennell & Oliveira (2008). Nesting problems are NP-complete (Fowler et al. (1981)) and, according to the typology proposed by Wäscher et al. (2007), they are classified as two-dimensional irregular open dimension problems.

Throughout this paper, we consider each piece represented by a reference point and vertices coordinates relative to this reference point, as illustrated by Figure 2a. Therefore, positioning a piece within the object is equivalent to placing the reference point in any coordinate within the object. To ensure the geometric feasibility in the layouts, the no-fit polygon (NFP) structure is used. Figure 2b presents the NFP between pieces $P1$ and $P2$, denominated by NFP_{12}. The NFP is a polygon obtained from the two pieces and is used to determine whether those pieces overlap (reference point of $P2$ inside NFP_{12}), touch (reference point of

$P2$ on NFP$_{12}$), or are separated from each other (reference point of $P2$ strictly outside NFP$_{12}$). A benefit of using NFP is that it also handles the shapes of the pieces, them being either convex or non-convex.



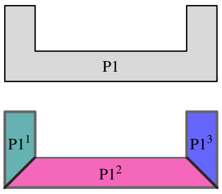
(a) Pieces and their reference point.



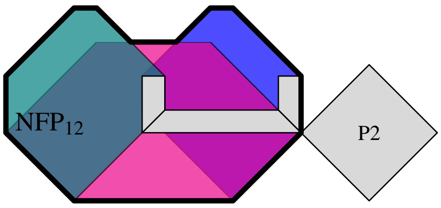
(b) NFP.

Figure 2 – Pieces and NFP.

The matheuristic proposed uses the NFP Covering Model (NFP-CM) (Cherri et al. (2016)) that is one of the most efficient models to deal with nesting problems. According to the authors, the main advantage of using NFP-CM is the fact that it uses simpler geometric tools, making it easier to be implemented and numerically more stable. In this model, each piece is represented by a set of convex polygons, the parts of the piece, allowing to represent both convex and non-convex pieces. Due to this representation, the NFP between piece i and piece j is defined as the set of all NFP of parts of the piece i and parts of the piece j (NFP$_{ij}^{p}$). Figure 3 shows pieces $P1$, $P2$ and the NFP$_{12}$ when represented by a set of convex polygons. Figure 3a presents the original piece $P1$ (a non-convex polygon) and its representation using a set of convex polygons, while Figure 3b presents the NFP$_{12}$ as the union of all NFP parts, i.e., it combines NFP$_{12}^{1}$ (green), which is the NFP between $P2$ and the first part of $P1$, NFP$_{12}^{2}$ (pink), which is the NFP between $P2$ and the second part of $P1$ and NFP$_{12}^{3}$ (blue), which is the NFP between $P2$ and the third part of $P1$.



(a) P1 and P1 represented by a set of 3 convex parts (P1^p, $p = 1,2,3$



(b) NFP$_{12}$ as the union of all NFP parts.

Figure 3 – Set of convex polygons representing pieces and NFP, in NFP-CM.

The indexes, parameters, and variables of the NFP-CM are defined below.

Sets, indexes and parameters

- $N = \sum_{t=1}^{T} d_t$: total number of pieces to be cut, where T is the number of piece types and d_t is the demand of each type, $t \in \{1,\dots,T\}$;
- $i, j \in \{1,\dots,N\}$: indices of pieces to be cut;
- NFP$_{ij}$: no-fit polygon between pieces i and j;
- $\mathbb{Q}_{ij} = \{1,\dots,Q_{ij}\}$: set of parts of NFP$_{ij}$;
- NFP$_{ij}^{p}$: part p of no-fit polygon NFP$_{ij}$ (NFP$_{ij} = \bigcup_{p\in\mathbb{Q}_{ij}}$ NFP$_{ij}^{p}$);

- $\mathbb{K}_{ij}^p = \{1,\ldots,K_{ij}^p\}$: set of directed edges of the NFP$_{ij}^p$;
- $(c_{ij,x}^p, c_{ij,y}^p)$ and $(d_{ij,x}^p, d_{ij,y}^p)$: two consecutive vertices of the NFP$_{ij}^p$;
- W: object width;
- l_i^{min} and l_i^{max}: distances between the reference point of the piece i and its leftmost and rightmost vertex, respectively;
- w_i^{min} and w_i^{max}: distances between the reference point of the piece i and its highest and lowest vertex, respectively.

Variables

- L: layout length;
- x_i: x-coordinate of the reference point of the piece i;
- y_i: y-coordinate of the reference point of the piece i;
- v_{ij}^{pk}: binary variable that equals 1 if the reference point of the piece j is on the right of edge k of NFP$_{ij}^p$ and 0 otherwise.

The NFP-CM, proposed by Cherri et al. (2016), is given by:

$$\min \quad L \tag{1}$$

s.a:

$$l_i^{min} \le x_i \le L - l_i^{max}, \qquad i \in \{1,\ldots,N\}, \tag{2}$$

$$w_i^{min} \le y_i \le W - w_i^{max}, \qquad i \in \{1,\ldots,N\}, \tag{3}$$

$$\begin{aligned}&(c_{ij,y}^p - d_{ij,y}^p)(x_j - x_i) - (c_{ij,x}^p - d_{ij,x}^p)(y_j - y_i) + (c_{ij,x}^p - d_{ij,x}^p)c_{ij,y}^p \\ &- (c_{ij,y}^p - d_{ij,y}^p)c_{ij,x}^p \le (1 - v_{ij}^{pk})M_{ij}^{pk}, \qquad i \in \{1,\ldots,N-1\}, \\ &\qquad\qquad j \in \{i+1,\ldots,N\}, k \in \mathbb{K}_{ij}^p, p \in \mathbb{Q}_{ij},\end{aligned} \tag{4}$$

$$\sum_{k \in \mathbb{K}_{ij}^p} v_{ij}^{pk} = 1, \qquad i \in \{1,\ldots,N-1\}, j \in \{i+1,\ldots,N\}, p \in \mathbb{Q}_{ij}, \tag{5}$$

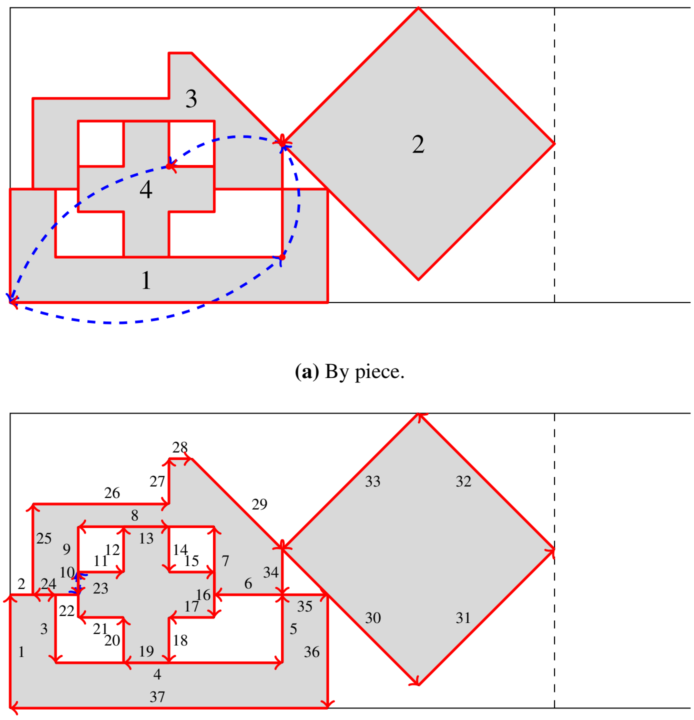
$$x_i, y_i \in \mathbb{R}_+, \qquad i,j \in \{1,\ldots,N\}, \tag{6}$$

$$v_{ij}^{pk} \in \{0,1\}, \qquad i,j \in \{1,\ldots,N\}, k \in \mathbb{K}_{ij}^p, p \in \mathbb{Q}_{ij}. \tag{7}$$

The NFP-CM aims to minimize the layout length. Constraints (2) and (3) are responsible for ensuring that all pieces are inside the object. Constraints (4) ensure that there is no overlap between the pieces and they are written based on the NFP between pieces i and j. To ensure no overlap between the pieces, the reference point of the piece j must be in the outer area of the NFP$_{ij}^p$. These constraints are satisfied when we guarantee that the reference point of piece j is on the right side of exactly one oriented edge k, $k \in \mathbb{K}_{ij}^p$, considering the counterclockwise direction of the edges. Constraints (5) ensure that exactly one edge k, $k \in \mathbb{K}_{ij}^p$, separates pieces i and j and, finally, constraints (6) and (7) define the domains of the variables.

2.2 The cutting path determination problem

The problem of finding the best way to cut the pieces arises after a layout is established. This problem is known as the cutting path determination problem (CPDP) and consists of finding the shortest path that the cutting tool has to follow to cut the previously defined layout. Figure 4 illustrates two types of cutting paths. In Figure 4a, each piece is cut entirely before the next one is cut and the numbers indicate the sequence in which the pieces are cut, while in Figure 4b, the cuts are done edge by edge and the numbers indicate the sequence of edges to be cut. In both figures, the red lines represent the cuts made by the cutting tool and the blue dotted ones represent the idle movements.



(a) By piece.

(b) By edge.

Figure 4 – Examples of cutting paths.

The approaches considering the cuts by edges are usually based on arc routing problems, in which the pieces' edges belong to the subset of required edges while the other edges of the complete graph represent the idle movements. The objective is to find the minimum path that contains all required edges. When considering the cut by piece, the approaches are based on the node routing problem, in which each piece is represented by a vertex and visiting a vertex corresponds to cut the piece entirely. Manber & Israni (1984), Moreira et al. (2007), Rodrigues & Ferreira (2011), Dewil et al. (2011), Dewil et al. (2014) and Silva et al. (2019) consider that the cuts are done by edges (Figure 4b), while Han & Na (1999) and Lee & Kwon (2006) consider the cut by pieces (Figure 4a). Oliveira et al. (2020), who proposed the first mathematical model that integrates the nesting and the cutting path determination problems, also consider the cut by pieces.

Several characteristics define a CPDP and each combination of these characteristics results in a different CPDP variation. In this paper, we consider the cut by edges version of the problem, done by a laser cutting tool on a thin metal plate (object), so the drilling time can be ignored; the cutting tool does not wear out during the cutting process, therefore, there is no change in the cutting capacity; the cut is made by only one cutting tool; the diameter of the tool is small, so there is no significant space between the pieces after the cut; and the cutting tool always starts at a predefined spot, called a depot, and returns to it at the end of the process. Regarding the object and the characteristics of the pieces, the edges of the pieces are considered independent, i.e. they do not need to be cut one right after the other at once; there is no mandatory direction for cutting an edge; the object is not affected by heat during the cutting process.

With similar characteristics as considered in this paper, Silva et al. (2019) considered two classical mathematical models to solve the CPDP. The first one is based on the rural postman problem (RPP), and the second one is based on the traveling salesman problem (TSP). The RPP model was a simpler and more efficient formulation that obtained better results when compared to the TSP model; therefore, it is used in the matheuristic proposed in this paper. In the RPP model, the vertices of the pieces are represented by nodes, while the sides of the pieces correspond to the required edges. The remaining edges of the complete graph represent the possible movements of the cutting tool between the vertices of the pieces while it is turned off, i.e., the idle movements.

The sets, parameters, and variables of the RPP model (Silva et al. (2019)) are defined below:

Sets and parameters

- $\mathbb{V}$: set of vertices;
- $\mathbb{A}$: set of edges;
- $\mathbb{A}_\mathbb{R}$: set of required edges ($\mathbb{A}_\mathbb{R} \subseteq \mathbb{A}$);
- $\mathbb{P}$: set of connected components in the subgraph defined by $\mathbb{A}_\mathbb{R}$;
- $\Delta(k)$: set of edges incident on vertex $k \in \mathbb{P}$;
- $\Delta_R(i)$: set of required edges incident on vertex $i \in \mathbb{V}$ ($\mathbb{A}_\mathbb{R} \cap \Delta(i)$);
- c_e: cost of traversing edge $e \in \mathbb{A}$.

Variables

- x_e: integer variable indicating the number of times edge e is traversed;
- w_i: auxiliary integer variable used to maintain the cardinality of vertex i even.

The RPP model, used by Silva et al. (2019) to solve the CPDP, is given by:

$$\min \sum_{e \in \mathbb{A}} c_e x_e \tag{8}$$

$$s.a. |\Delta_R(i)| + \sum_{e \in \Delta(i)} x_e = 2w_i, \qquad \forall i \in \mathbb{V}, \tag{9}$$

$$\sum_{e \in \Delta(k)} x_e \geq 2, \qquad \forall k \in \mathbb{P}, \tag{10}$$

$$x_e \in \mathbb{Z}_+, \qquad \forall e \in \mathbb{A}, \tag{11}$$

$$w_i \in \mathbb{Z}_+, \qquad \forall i \in \mathbb{V}. \tag{12}$$

The model (8) - (12) aims to minimize the cost (distance) of the cutting path. Constraints (9) ensure that each vertex has even cardinality. Constraints (10) guarantee that there is no illegal subtour. Finally, constraints (11) and (12) define the variables domain.

3 BRKGA-BASED MATHEURISTIC

To solve the NCPP, we developed a Biobjective BRKGA-based matheuristic (BRKGA-BiMath). Here, the BRKGA is used to select variables which will reduce the complexity of a mathematical model. The

BRKGA, proposed by Gonçalves & Resende (2011), is an evolutionary metaheuristic for discrete and global optimization problems, based on the random-key algorithm proposed by Bean (1994). According to the authors, the BRKGA is a framework to solve any optimization problem and is divided into two: the problem-independent part and the problem-dependent one. The problem-independent part is responsible for the evolution and requires no knowledge about the problem that is being solved, while the problem-dependent one is responsible for decoding the random-key vectors and returning a representative value (fitness), such as the cost, associated with each vector.

The BRKGA has been used to solve a large variety of problems, as the berth allocation and quay crane assignment problem (Correcher & Alvarez-Valdes (2017)), the flowshop scheduling problem (Andrade et al. (2019)) and the two-stage capacitated facility location problem (Biajoli et al. (2019)). Considering the cutting and packing problems, Mundim et al. (2017) used the BRKGA to determine the sequence of the pieces to be placed within an object with one or two open dimensions and Amaro-Júnior et al. (2017) proposed a parallel BRKGA with multiple populations, considering rotations of the pieces.

A brief review of the problem-independent structure is made in Section 3.1. For more details, the reader is referred to Gonçalves & Resende (2011). One of the contributions of this research lays in the problem-dependent structure, which is detailed in Section 3.2.

3.1 The BRKGA problem-independent structure

In BRKGA, each solution is represented by a vector of n random keys in which each key is a real number within the continuous interval [0,1). In general, the BRKGA algorithm starts with a population of p random-key vectors, also called chromosomes. In each generation, the chromosomes are divided into two sets: the elite set, formed by the pe chromosomes with better fitness, and the non-elite set, with the remaining population ($p - pe$). Figure 5 illustrates the BRKGA evolutionary dynamic, with the current generation k on the left and the next-generation $k+1$ on the right. All chromosomes of the elite set in generation k are copied to the next generation $k+1$, and a small number of pm random-key vectors, called mutants, is added to the next generation $k+1$. Finally, the $p - pe - pm$ chromosomes are generated by a crossover between two chromosomes (parent) from generation k, one chosen from the elite set and the other one from the non-elite set. Parent selection is made randomly and with replacement (sampling with replacement), i.e., a parent can be selected more than once in generation k.

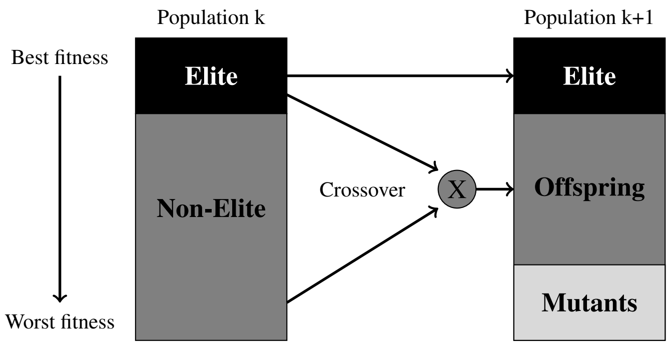


Figure 5 – BRKGA evolutionary dynamics.

Adapted from Gonçalves & Resende (2011).

The crossover process between two chromosomes is illustrated in Figure 6. Chromosome A represents an elite chromosome and Chromosome B a non-elite one. In this example, we consider an 80% probability that a descendant will inherit an allele from the elite chromosome. That means that if a number, randomly generated within [0,1), is less than 0.8, then the new chromosome inherits the allele from the elite chromosome, otherwise it inherits the allele from the non-elite chromosome. In the example, the descending chromosome inherits alleles from the elite chromosome in the first, fourth and fifth alleles, and inherits alleles from the non-elite chromosome in the second and third alleles.

Chromosome A	**0.12**	**0.46**	**0.24**	**0.62**	**0.99**
Chromosome B	0.47	0.83	0.73	0.19	0.45
Random number	0.32	0.81	0.95	0.25	0.51
Crossover probability of 0.8	$<$	$>$	$>$	$<$	$<$
Offspring chromosome	**0.12**	0.83	0.73	**0.19**	**0.45**

Figure 6 – Crossover between two chromosomes.

With the population completed, the decoder calculates the cost (fitness) of all new chromosomes, the population is divided into new elite and non-elite sets and the algorithm starts a new generation. The algorithm is interrupted when a stopping criterion is reached and the solution of the first elite chromosome is provided as the best solution.

3.2 The BRKGA problem-dependent structure

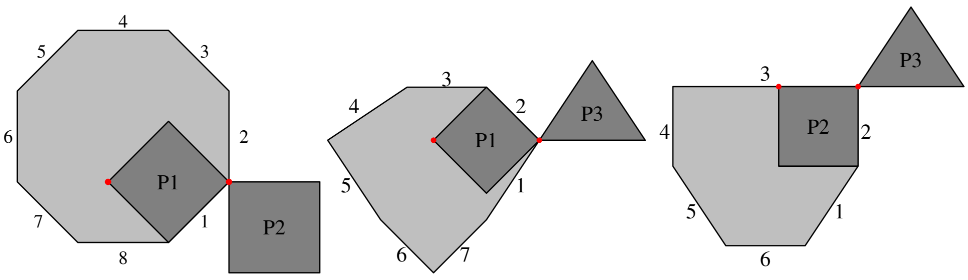
To adapt the BRKGA to any optimization problem, it is necessary to design only the problem-dependent part of the algorithm, i.e.: i) define what is represented by the chromosome, and therefore the chromosome size; ii) develop a decoder; and iii) return a representative value for each chromosome (fitness). In the following subsections, we detail each element of the BRKGA-BiMath problem-dependent structure for solving the NCPP.

3.2.1 Chromosome

The nesting model considered to provide a layout was proposed by Cherri et al. (2016) (NFP-CM). The NFPs are used to determine whether two pieces overlap, touch, or are separated from each other. Note that, in NFP-CM, there is a binary variable v_{ij}^{pk} for each edge k of NFP$_{ij}^{p}$, in which p represents the part p of NFP$_{ij}$, as explained in Section 2.1, and one constraint is added to the model for each v_{ij}^{pk}. However, only one constraint for each NFP$_{ij}^{p}$ needs to be activated.

In our matheuristic, we define that each allele of the chromosome represents an edge of the NFP$_{ij}^{p}$. Therefore, the size of the chromosome is equal to the sum of all parts of the NFP. More details are presented in the next section. The nesting model is reduced to a continuous linear model because the BRKGA, through the chromosome, defines which edges should be used to guarantee the non-overlap between pieces, i.e., which variables v_{ij}^{pk} are equal to 1. Consequently, model (1) - (7) becomes easier and faster to solve.

Once the size of the chromosome is defined (n), the BRKGA provides a population of p vectors (chromosomes) with n random keys each. Here, each random key encodes which edge of the NFP is used to ensure the pieces do not overlap each other. In other words, the chromosomes are vectors whose size equals the sum of all parts of the NFPs. Each element of the vector (i.e. each allele) is a real number in the range [0, 1], which will be decoded into one of the edges of a part of the NFP and used to ensure the pieces do not overlap each other. To make it easier to understand, we will illustrate it using an instance with only convex pieces, i.e., an instance in which $p = 1$ for all NFP_{ij}^{p}. The procedure is analogous for instances with NFP_{ij}^{p}, $p > 1$. This instance has three convex pieces, therefore 3 NFPs (NFP_{12}, NFP_{13} and NFP_{23}). Figure 7a presents the NFPs and their edges enumerated counterclockwise. The NFPs have a total of 8, 7 and 6 edges, respectively. Figure 7b shows an example of a chromosome while 7c shows its decodification. For the first allele, the interval [0,1) is equally divided into 8 parts (8 edges of the NFP_{12}). Therefore, random keys within the range [0; 0.125) are decoded into edge 1 of NFP_{12}, random keys within the range [0.125; 0.25) are decoded into edge 2 of NFP_{12} and so on. The same happens to the other alleles, according to the total edges of the respective NFP. In this example, the first random key is decoded into the third edge of NFP_{12}, the second random key is decoded into the first edge of NFP_{13} and the third random key is decoded into the fifth edge of NFP_{23}.



(a) The NFP's enumerated edges.

NFP_{12}	NFP_{13}	NFP_{23}
0.37	0.13	0.79

(b) Chromosome.

NFP_{12}	NFP_{13}	NFP_{23}
3	1	5

(c) Decoded chromosome.

Figure 7 – Proposed decoder.

3.2.2 Fitness

Following the decodification of each chromosome, a representative value (fitness) has to be returned to continue the BRKGA evolution. In the BRKGA-BiMath, two linear models are used, one for each problem. First, the NFP-CM, proposed by Cherri et al. (2016) and presented in Section 2.1, is used to obtain the nesting pattern. In the nesting model, the non-overlap constraints are written based on the NFP edges between each pair of pieces. Therefore, once a chromosome is decoded into a set of NFPs edges, this information is transferred to the NFP-CM model, which becomes a linear model as a consequence. After a layout is obtained, the RPP model, used by Silva et al. (2019) and presented in Section 2.2, is used to find the shortest cutting path for the nesting pattern.

After the process described above, each chromosome has two representative values: i) the layout length; and ii) the cutting path distance. However, only one fitness value should be informed back to BRKGA. Roque et al. (2017a), Roque et al. (2017b) and Tangpattanakul et al. (2013) solved the issue of transforming two representative values into one fitness by using the fast non-dominated sorting and the crowding distance algorithms proposed by Deb et al. (2002).

The fast non-dominated sorting approach sorts the population into distinct non-domination levels. The procedure starts by calculating, for each solution p, the number of solutions that dominate the solution p (n_p) and a set of solutions which solution p dominates (S_p). Every solution p, which $n_p = 0$, belongs to the first non-dominated front. For each solution in the first non-dominated front, we deduce the domination count n_q by one for every solution $q \in S_p$. If, for any solution q, n_q becomes zero, we put q in a separate list, meaning that q belongs to the second non-dominated front. This procedure continues until all solutions are sorted.

To preserve the solutions diversification, i.e., a uniform spread of solutions, we use the crowding distance procedure. This procedure is applied for each non-dominated front. First, we sort the population of the front according to each objective in ascending order. For each objective, the boundary solutions are assigned an infinite distance value, while all other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions. The total crowding distance of a solution is the sum of individual distance values concerning each objective.

Applying these two procedures, the representative values become one single fitness value. In other words, the layout length and the cutting path distance are replaced by the non-domination level through the fast non-dominated sorted strategy and the crowding distance. The BRKGA, which once returned only the best elite solution, now returns the first frontier.

3.2.3 Strategies to handle the infeasibility within a layout

Preliminary tests showed that some chromosomes, in the BRKGA-BiMath, led to infeasible solutions, i.e., pieces overlapping each other, within the BRKGA evolution. To analyze the BRKGA-BiMath behavior when considering infeasible layouts, a second strategy was considered, in which the nesting model was extended to allow the overlap. Artificial variables were added to the non-overlap constraints to allow the overlaps between each par of pieces and the objective function was modified to minimize the overlap throughout the evolution process.

These two strategies are detailed in the following. The one that does not allow layout infeasibility in the nesting model is denominated as Strategy 1 while the second one, which allows layout infeasibility, is denominated as Strategy 2.

Strategy 1: not allowing layout infeasibility in the nesting

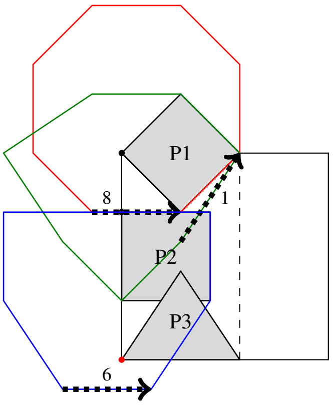
Here, we considered the NFP-CM reduced to a linear model, to provide the layout. For each feasible layout, the shortest path the cutting tool has to follow to cut the layout is determined through the RPP model used by Silva et al. (2019), presented in Section 2.2.

In case a chromosome leads to an infeasible layout, i.e., with pieces overlapping each other, the RPP model is not used and a big enough value is assigned to both layout length and cutting path distance.

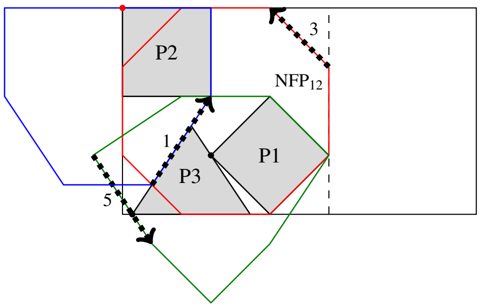
Strategy 2: allowing layout infeasibility in the nesting model

As already mentioned, some chromosomes may lead to infeasible layouts, especially in the BRKGA early generations. These infeasible layouts can be divided into two groups. Considering the same instance shown in Figure 7, Figure 8 shows an example for each group. Figure 8a shows the first infeasibility case, in which, for example, the chromosome [0.964; 0.212; 0.958] is decoded into edges 8, 1 and 6, respectively, which are highlighted. The decodification generates a layout with overlap between the pieces $P2$ and $P3$.

The second case is shown in Figure 8b. The chromosome [0.373; 0.660; 0.114] is decoded into edges 3, 5 and 1, respectively, which are also highlighted. Note that, in this second case, there is no overlap between the pieces. However, there is an infeasibility in the constraints of the model, since the reference point of the piece $P2$ lies in the right side area of edge 3 of the NFP$_{12}$.



(a) Solution with overlap.



(b) Solution without overlap, but infeasible.

Figure 8 – Example of solutions obtained by the decoder.

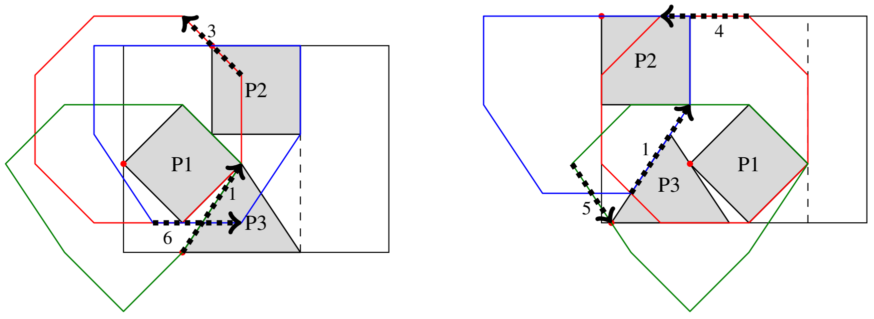
In Strategy 1, whenever one of these situations occur, the method considers the layouts as infeasible and a big enough value is assigned as the layout and the cutting path length. However, these chromosomes may have infeasibilities that could be beneficial in leading to better results. In both examples presented in Figure 8, changing only one edge is enough to obtain a feasible layout, as shown in Figure 9. Figure 9a presents a layout considering edges 3, 1 and 6 instead of 8, 1 and 6, respectively; while the layout presented in Figure 9b uses edges 4, 5 and 1 instead of 3, 5 and 1, respectively. We highlight that those are just manually created examples, built to illustrate situations that could happen within generations.

Therefore, to allow these situations, artificial variables a_{ij}^p are added to the non-overlap constraints to allow the overlaps between each pair of pieces i and j and the artificial variable b_i is added to the constraints that ensure that every piece i is entirely inside the object. In other words, constraints (3) and (4) are replaced by

$$w_i^{min} \leq y_i \leq W - w_i^{max} + b_i, \qquad i \in \{1,\dots,N\}, \tag{13}$$

$$\begin{aligned}&(c_{ij,y}^p - d_{ij,y}^p)(x_j - x_i) - (c_{ij,x}^p - d_{ij,x}^p)(y_j - y_i) + (c_{ij,x}^p - d_{ij,x}^p)c_{ij,y}^p \\ &- (c_{ij,y}^p - d_{ij,y}^p)c_{ij,x}^p - a_{ij}^p \leq 0, \qquad i \in \{1,\dots,N-1\}, \\ &\qquad\qquad\qquad\qquad j \in \{i+1,\dots,N\}, p \in \mathbb{Q}_{ij}\end{aligned} \tag{14}$$

respectively. In this strategy, the new objective is to find a feasible solution, i.e. a layout in which pieces do not overlap each other and are completely inside the object. Besides, among the solutions with no in-



(a) Solution without overlap (optimal). **(b)** Solution without overlap.

Figure 9 – Example of feasible solutions obtained by the decoder after changing only one allele value.

feasibility, we choose the one with the shorter length. Therefore, the objective function (1) is replaced by

$$\min \quad \sum_{i=1}^{N} b_i + \sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{p=1}^{Q_{ij}} a_{ij}^{p} + \alpha L, \tag{15}$$

in which α is a sufficiently small number. In our experiments, we considered $\alpha = 10^{-6}$.

For each feasible layout, i.e. layout in which $\sum_{i=1}^{N} b_i + \sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{p=1}^{Q_{ij}} a_{ij}^{p} = 0$, the shortest cutting path is determined by model (8) - (12). For infeasible layouts, a big enough value is assigned as the cutting path distance.

4 COMPUTATIONAL EXPERIMENTS

A set of 24 instances, from the nesting problem literature, was used in the computational tests. Details about the instances, such as names, object width, the total of piece types, and the total of pieces are shown in Table 1. Since the object has unlimited length, there is no initial length to be reported in Table 1. The instances can be found on the ESICUP website (https://www.euro-online.org/websites/esicup/). The BRKGA API proposed by Toso & Resende (2012) was also used.

The mathematical models were implemented in C/C++ and the experiments were performed on an Intel Core i7-7700 (3.60GHz x 8) computer, with 16GB RAM and Ubuntu 16.04 operating system, using the optimization software ILOG CPLEX 12.6 with the default parameters. The BRKGA parameters were experimentally tuned using the **irace** package proposed by López-Ibáñez et al. (2016). The parameters and their values are presented in Table 2, in which n is the size of the chromosome.

The computational experiments are conducted in three phases. In the first one, we evaluate BRKGA-BiMath Strategy 1 and Strategy 2. The best strategy is compared, in the second phase, with the hierarchical approach, in which the problems are solved individually. Finally, in the third phase, we test the BRKGA-BiMath to its limit, i.e., the objective is to analyze the BRKGA-BiMath capacity of solving instances with more pieces until it reaches the machine's memory limit. In all phases, the BRKGA-BiMath was restarted ten times, with different seeds. All ten runs' first rank frontier solutions were combined and then resorted using the fast non-dominated sorted strategy and the crowding distance algorithm.

Table 1 – Instances information.

Instances	Object	Piece	
	Width	Types	Total
three	7	3	3
threep2	7	3	6
threep2w9	9	3	6
threep3	7	3	9
threep3w9	9	3	9
shapes4	13	4	4
shapes4N	20	4	4
fu5	38	4	5
fu6	38	5	6
fu7	38	6	7
fu8	38	7	8
fu9	38	8	9
fu10	38	9	10
fu	38	10	12
rco1	15	7	7
rco2	15	7	14
rco3	15	7	21
rco4	15	7	28
rco5	15	7	35
blazewicz1	15	7	7
blazewicz2	15	7	14
blazewicz3	15	7	21
blazewicz4	15	7	28
blazewicz5	15	7	35

Table 2 – BRKGA parameters considered.

Parameters	Strategy 1	Strategy 2
Population size	$8n$	$10n$
Population to be considered as elite	21%	14%
Population to be replaced by mutants	15%	18%
Inheritance rate from an elite parent	59%	60%
Time limit	720s	720s

4.1 Phase I - Evaluating BRKGA-BiMath Strategy 1 and Strategy 2

In this phase, the main objective is to evaluate the relevance of allowing infeasible solutions in the BRKGA-BiMath evolution. As already mentioned, in Strategy 1, a big enough value is assigned to the layout length and the cutting path distance whenever the chromosome is decoded into an infeasible layout. Consequently, all these infeasible chromosomes have the same solution and will be discarded during the BRKGA evolution, eventually. In Strategy 2, the chromosomes are sorted by the amount of infeasibility in their layout. In other words, layouts with less infeasibility are higher ranked than the ones with more infeasibility. The cutting path distance is equal to a big enough value in the case of layout infeasibility.

The first rank frontier solutions, for each instance, are presented in Tables 3-5. The first column contains the name of the instances while the second and third columns present the layout length (Nest_1) and the cutting path distance (CP_1), respectively, considering Strategy 1. The fourth and fifth columns present the same information (Nest_2 and CP_2), considering Strategy 2. In bold, we highlight the layout length that equals the best-known values in the nesting literature and the minimum cutting path distance (CP = 0.00). The symbol "-" indicates that no feasible solution was found, while empty spaces mean that one strategy was able to find more solutions in the first rank frontier than the other.

Table 3 – First rank frontier for Strategies 1 and 2 - instances three and shapes.

Instances	Strategy 1		Strategy 2	
	Nest_1	CP_1	Nest_2	CP_2
three	**6.00**	2.82	6.20	2.26
	6.20	2.26	6.67	1.34
	7.00	**0.00**	7.00	**0.00**
threep2	**9.33**	10.72	9.67	4.00
	9.67	4.00	10.00	3.41
	9.78	3.19	10.67	2.00
	10.67	1.34	11.00	**0.00**
	11.00	**0.00**		
threep2w9	**8.00**	5.41	**8.00**	5.41
	8.20	4.26	8.20	4.26
	9.00	**0.00**	8.50	2.85
			9.00	**0.00**
threep3	14.67	6.04	14.25	10.26
	14.85	5.74	14.33	6.91
	16.00	2.38	15.67	4.16
	16.33	2.10	16.05	3.83
	17.00	2.00	17.00	**0.00**
	19.00	**0.00**		
threep3w9	12.00	7.83	12.00	2.83
	12.38	7.75	13.00	**0.00**
	12.40	4.58		
	13.00	**0.00**		
shapes4	-	-	**24.00**	2.00
			27.00	**0.00**
shapes4N	-	-	16.00	4.00
			17.00	**0.00**

Table 4 – First rank frontier for Strategies 1 and 2 - instances fu.

Instances	Strategy 1		Strategy 2	
	$Nest_1$	CP_1	$Nest_2$	CP_2
fu5	**17.89**	4.38	**17.89**	4.38
	24.00	**0.00**	24.00	**0.00**
fu6	24.00	**0.00**	24.00	**0.00**
fu7	**24.00**	**0.00**	**24.00**	**0.00**
fu8	**24.00**	**0.00**	**24.00**	2.00
			28.00	**0.00**
fu9	29.00	10.00	29.00	6.82
	34.00	2.00	35.00	**0.00**
	38.00	**0.00**		
fu10	38.00	**0.00**	32.81	34.60
			33.93	22.62
			37.13	11.10
			38.00	**0.00**
fu	-	-	47.00	16.90
			48.00	12.24
			52.00	12.00
			57.00	4.46
			59.00	2.00

Table 5 – First rank frontier for Strategies 1 and 2 - instances rco and blazewicz.

Instances	Strategy 1		Strategy 2	
	$Nest_1$	CP_1	$Nest_2$	CP_2
rco1	**8.00**	**0.00**	**8.00**	**0.00**
rco2	-	-	29.67	21.46
rco3	-	-	-	-
rco4	-	-	-	-
rco5	-	-	-	-
blazewicz1	-	-	9.45	11.22
			9.50	6.62
			9.67	6.16
			10.00	0.66
			11.00	**0.00**
blazewicz2	-	-	23.05	16.22
			25.00	14.57
blazewicz3	-	-	-	-
blazewicz4	-	-	-	-
blazewicz5	-	-	-	-

Analyzing Tables 3-5, we observe that the BRKGA-BiMath using Strategy 1 was unable to find feasible solutions for twelve instances: “shapes4”, “shapes4N”, “fu”, “rco2” - “rco5” and “blazewicz1” - “blazewicz5”. The literature best layout lengths were obtained for seven instances (“three”, “threep2”, “threep2w9”, “fu5”, “fu7”, “fu8” and “rco1”), while the shortest cutting path (CP = 0.00) were presented in all twelve instances solved by Strategy 1. Meanwhile, Strategy 2 was unable to find feasible solutions for only six instances: “rco3” - “rco5” and “blazewicz3” - “blazewicz5”. The literature best layout lengths were obtained for six instances (“threep2w9”, “shapes4”, “fu5”, “fu7”, “fu8” and “rco1”), while the shortest cutting path (CP = 0.00) were presented in fifteen instances solved by Strategy 2 (“three”, “threep2”, “threep2w9”, “threep3”, “threep3w9”, “shapes4”, “shapes4N”, “fu5”, “fu6”, “fu7”, “fu8”, “fu9”, “fu10”, “rco1” and “blazewicz1”).

It is also worth noting that not all instances presented a trade-off behavior between the layout length and the cutting path distance, see instances “fu7”, “fu8” and “rco1”. These instances presented a layout with both the minimum layout length and the shortest cutting path (CP = 0.00), considering the characteristics presented in Table 1. If others characteristics were assumed, this nonconflicting behavior could change to a conflicting one.

To evaluate which strategy performed better, three metrics are used to compare them. The first one, metric A, represents the amount of solutions within the first rank frontier obtained by Strategy 1 and Strategy 2, σ_1^A and σ_2^A, respectively; the second one, metric B, is the hypervolume, computed using the algorithm provided by Fonseca et al. (2006), that can be found on the website https://lopez-ibanez.eu/hypervolume; the third one, metric C, denotes the fraction of solutions obtained by Strategy 1 that are dominated by solutions obtained by Strategy 2, denoted by $\sigma_{1,2}^C$. The closer this metric is to 1 the more Strategy 2 dominates Strategy 1. Similarly, $\sigma_{2,1}^C$ denotes the fraction of solutions obtained by Strategy 2 that are dominated by solutions obtained by Strategy 1. The metrics’ values are presented in Table 6, in which the first column contains the instance names, while the second and third columns present the metric A for Strategy 1 and Strategy 2, respectively, the fourth and fifth columns present the metric B for Strategy 1 and Strategy 2, respectively, and the sixth and seventh columns present the metric C for Strategy 1 and Strategy 2, respectively.

According to metric A, we observe that Strategy 2 had a better performance than Strategy 1, since Strategy 2 was able to find more solutions in the first rank frontier than Strategy 1, for more instances. Strategy 1 obtained a total number of solutions in the first rank frontier equal to or greater than Strategy 2 for nine instances (“three”, “threep2”, “threep3”, “threep3w9”, “fu5”, “fu6”, “fu7”, “fu9” and “rco1”) while Strategy 2 obtained a total of solutions equal to or greater than Strategy 1 for fourteen instances (“three”, “threep2w9”, “shapes4”, “shapes4N”, “fu5”, “fu6”, “fu7”, “fu8”, “fu10”, “fu”, “rco1”, “rco2”, “blazewicz1” and “blazewicz2”).

Considering metric B, we observe that Strategy 2 performed better than Strategy 1, since Strategy 2 was able to find the first frontier solution with equal or bigger hypervolume for fouteen instances (“threep2w9”, “threep3”, “shapes4”, “shapes4N”, “fu5”, “fu6”, “fu7”, “fu8”, “fu10”, “fu”, “rco1”, “rco2”, “blazewicz1” and “blazewicz2”), compared to only eight instances (“three”, “threep2”, “threep3w9”, “fu5”, “fu6”, “fu7”, “fu9” and “rco1”) that Strategy 1 performed better.

Finally, observing metric C, while considering only the instances that were solved by both strategies, we observe that three instances solved by Strategy 1 had some solutions, within the first frontier, dominated by solutions obtained by Strategy 2 (“threep3”, “threep3w9” and “fu9”) while only two instances solved by Strategy 2 had some solutions dominated by solutions obtained by Strategy 1 (“threep2”, and “fu8”).

Table 6 – Comparison between Strategy 1 and 2, using three metrics.

Instances	σ_1^A	σ_2^A	σ_1^B	σ_2^B	σ_{12}^C	σ_{21}^C
three	**3**	**3**	**0.82**	0.57	**0.00**	**0.00**
threep2	**5**	4	**11.75**	1.60	**0.00**	0.50
threep2w9	3	**4**	1.56	**2.27**	**0.00**	**0.00**
threep3	**6**	5	13.14	**14.06**	0.33	**0.20**
threep3w9	**4**	2	**2.81**	0.36	0.75	**0.00**
shapes4	0	**2**	-	**0.51**	-	-
shapes4N	0	**2**	-	**0.51**	-	-
fu5	**2**	**2**	**1.18**	**1.18**	**0.00**	**0.00**
fu6	**1**	**1**	**0.01**	**0.01**	**0.00**	**0.00**
fu7	**1**	**1**	**0.01**	**0.01**	**0.00**	**0.00**
fu8	1	**2**	0.01	**0.61**	**0.00**	1.00
fu9	**3**	2	**33.91**	1.17	0.67	**0.00**
fu10	1	**4**	0.01	**62.77**	**0.00**	**0.00**
fu	0	**5**	-	**70.72**	-	-
rco1	**1**	**1**	**0.01**	**0.01**	**0.00**	**0.00**
rco2	0	**1**		**0.00**	-	-
rco3	0	0	-	-	-	-
rco4	0	0	-	-	-	-
rco5	0	0	-	-	-	-
blazewicz1	0	**5**	-	**14.27**	-	-
blazewicz2	0	**2**	-	**0.33**	-	-
blazewicz3	0	0	-	-	-	-
blazewicz4	0	0	-	-	-	-
blazewicz5	0	0	-	-	-	-

Combining these three metrics, we came to the conclusion that Strategy 2, which considers the layout infeasibility, outperforms Strategy 1. In other words, sorting the solutions considering the amount of infeasibility within the layout proved to be beneficial to the BRKGA evolution.

4.2 Phase II - Comparing BRKGA-BiMath with the hierarchical approach

In this section, we compare the BRKGA-BiMath using Strategy 2 and the hierarchical approach. The hierarchical approach consists of determining the minimum layout length first, using Cherri et al. (2016)'s NFP-CM, and then the minimum cutting path distance, solving the Silva et al. (2019)'s model (8) - (12). Both models were solved using CPLEX limited to 3,600 seconds each. However CPLEX solves the CPDP model in less than 10 seconds. Again, the BRKGA-BiMath were restarted ten times with different seeds, each run done in 720 seconds. All ten runs' first rank frontier solutions were combined and then resorted to using the fast non-dominated sorted strategy and the crowding distance.

The results of the hierarchical approach and the BRKGA-BiMath are presented in Tables 7-9. The first column contains the name of the instances, while columns 2 - 6 present the hierarchical approach results: the layout length (Nest.), the optimality gap related to the layout (GAP$_N$), the cutting path distance (CP), the optimality gap related to the CP (GAP_{CP}) and the total time to solve both models. Columns 7 - 8 present the BRKGA-BiMath results: the layout length (Nest.) and the cutting path distance (CP). The symbol "-" indicates that no feasible solution was found.

Analyzing Tables 7-9, the hierarchical approach was able to find feasible layouts for almost all instances, except "blazewicz5", and the optimal layout for twelve instances ("three", "threep2", "threep2w9", "shapes4", "shapes4N", "fu5", "fu6", "fu7", "fu8", "fu9", "fu10" and "rco1"), within less than 90 seconds. In all instances, except one, the cutting path distance obtained was the optimum considering the layouts and the runtime was less than 180 seconds. The exception happened in instance "rco5" in which CPLEX was not able to find a feasible solution for the cutting path model within the time limit. The reason could be the layout having many separate pieces, generating a non-connected graph, and therefore an NP-hard problem, see Silva et al. (2019).

Considering only the layout with the shortest length, in each instance, we notice a shorter cutting path distance in the BRKGA-BiMath results, except for "fu10", "rco2" - "rco5" and "blazewicz1" - "blazewicz5". It is worth mentioning that there may exist another layout with the same length but a smaller cutting path. For example, with the hierarchical strategy, "threep2w9" presented a layout with 8.00 and a cutting path of 7.41 length, while the BRKGA-BiMath was able to find another layout, with the same length, but a cutting path 2.00 shorter. In Figure 10, we illustrate the layouts and the cutting paths for this instance for both approaches.

Table 7 – The BRKGA-BiMath with Strategy 2 compared with the hierarchical approach - instances three and shapes4.

Instances	**Hierarchical approach**					**BRKGA-BiMath**	
	Nest.	GAP_N (%)	CP	GAP_{CP} (%)	T. time (s)	Nest.	CP
three	**6.00**	0.0	2.82	0.0	< 1	6.20	2.26
						6.67	1.34
						7.00	**0.00**
threep2	**9.33**	0.0	10.72	0.0	1	9.67	4.00
						10.00	3.41
						10.67	2.00
						11.00	**0.00**
threep2w9	**8.00**	0.0	7.41	0.0	4	**8.00**	5.41
						8.20	4.26
						8.50	2.85
						9.00	**0.00**
threep3	13.53	11.3	11.92	0.0	TL	14.25	10.26
						14.33	6.91
						15.67	4.16
						16.05	3.83
						17.00	**0.00**
threep3w9	11.00	21.4	3.73	0.0	TL	12.00	2.83
						13.00	**0.00**
shapes4	**24.00**	0.0	18.81	0.0	1	**24.00**	2.00
						27.00	**0.00**
shapes4N	**14.00**	0.0	8.82	0.0	1	16.00	4.00
						17.00	**0.00**

Table 8 – The BRKGA-BiMath with Strategy 2 compared with the hierarchical approach - instances fu.

Instances	Hierarchical approach					BRKGA-BiMath	
	Nest.	GAP$_N$ (%)	CP	GAP$_{CP}$ (%)	T. time (s)	Nest.	CP
fu5	**17.89**	0.0	8.55	0.0	< 1	**17.89**	4.38
						24.00	**0.00**
fu6	**23.00**	0.0	4.27	0.0	< 1	24.00	**0.00**
fu7	**24.00**	0.0	8.00	0.0	< 1	**24.00**	**0.00**
fu8	**24.00**	0.0	8.53	0.0	< 1	**24.00**	2.00
						28.00	**0.00**
fu9	**25.00**	0.0	9.97	0.0	5	29.00	6.82
						35.00	**0.00**
fu10	**28.69**	0.0	8.14	0.0	89	32.81	34.60
						33.93	22.62
						37.13	11.10
						38.00	**0.00**
fu	33.14	13.0	189.27	0.0	TL	47.00	16.90
						48.00	12.24
						52.00	12.00
						57.00	4.46
						59.00	2.00

Table 9 – The BRKGA-BiMath with Strategy 2 compared with the hierarchical approach - instances rco e blazewicz.

Instances	Hierarchical approach					BRKGA-BiMath	
	Nest.	GAP$_N$ (%)	CP	GAP$_{CP}$ (%)	T. time (s)	Nest.	CP
rco1	**8.00**	0.0	8.00	0.0	2	**8.00**	**0.00**
rco2	15.00	46.7	15.15	0.0	TL	29.67	21.46
rco3	22.57	77.6	23.09	0.0	TL	-	-
rco4	30.39	83.5	28.69	0.0	TL	-	-
rco5	38.46	87.0	-	-	TL	-	-
blazewicz1	7.40	29.1	6.68	0.0	TL	9.45	11.22
						9.50	6.62
						9.67	6.16
						10.00	0.66
						11.00	**0.00**
blazewicz2	14.67	65.9	14.60	0.0	TL	23.05	16.22
						25.00	14.57
blazewicz3	21.66	76.9	13.00	0.0	TL	-	-
blazewicz4	29.96	83.3	14.77	0.0	TL	-	-
blazewicz5	-	-	-	-	TL	-	-

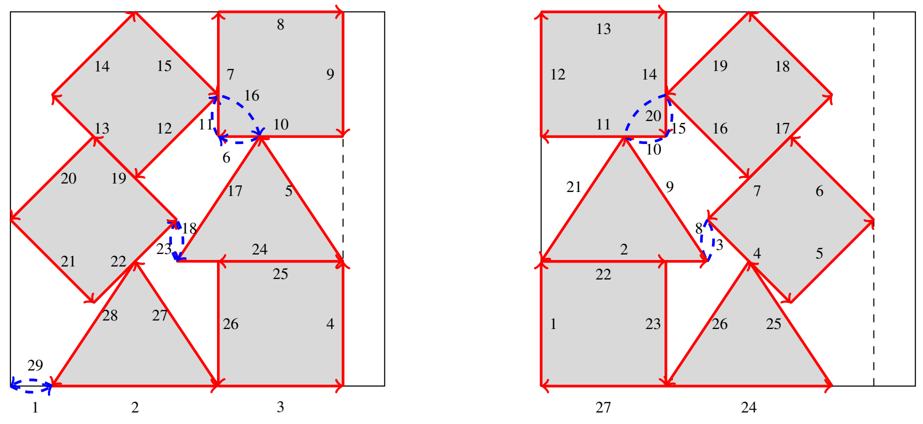

(**a**) Hierarchical approach. (**b**) BRKGA-BiMath.

Figure 10 – Layout and cutting path for instance threep2w9.

The influence of the layout in the determination of the cutting path is evident, i.e., when the layout length increases, the optimal solution for the cutting path distance decreases. For example, the cutting path distance of instance "threep2" decreased from 4.00, when the layout length was 9.67, to 0.00, when the layout length was 11.00. A layout with a shorter cutting path distance may improve the productivity of a workday since the time saved during the cutting process may be used to cut more layouts. On the other hand, this leads to more material waste, hence higher production costs. With these solutions at hand, the decision-maker can choose the most suitable solution according to the production situation. In other words, the decision-maker can choose a layout with a shorter cutting path distance but a longer layout length when the production is delayed, for example, or a layout with a shorter layout length but longer cutting path distance when the material costs are high.

However, as already mentioned, not all instances presented this trade-off behavior between the layout length and the cutting path distance, considering the characteristics presented in Table 1. For example, instance "fu7" presented a unique solution with both minimum layout length and minimum cutting path distance, i.e., a layout length of 24.00 (same as the hierarchical approach) with a total cutting path of 0.00. In Figure 11, we show the layouts and the cutting paths for this instance for both approaches. It is important to highlight that even in cases like this, it is worth using the BRKGA-BiMath, compared to the hierarchical approach, because it provides a layout with a shorter cutting path.

Finally, BRKGA-BiMath had solved instances with up to 14 pieces. However, to better evaluate BRKGA-BiMath capacity, we did the third phase of experiments.

4.3 Phase III - Analyzing BRKGA-BiMath capacity of solving instances with more pieces

In this phase, the objective is to test the BRKGA-BiMath capacity of solving instances with more pieces, considering Strategy 2, until it reaches the machine's memory limit. Therefore, a set of 7 instances was used in the computational tests. These instances, named as "three$\boldsymbol{p}$", have the same type of pieces as instance "three" and $\boldsymbol{p}$ represents the number of pieces of each type. We consider $\boldsymbol{p}$ equal to $4, 5, \ldots, 10$. The parameter values considered are the same presented in Table 2.

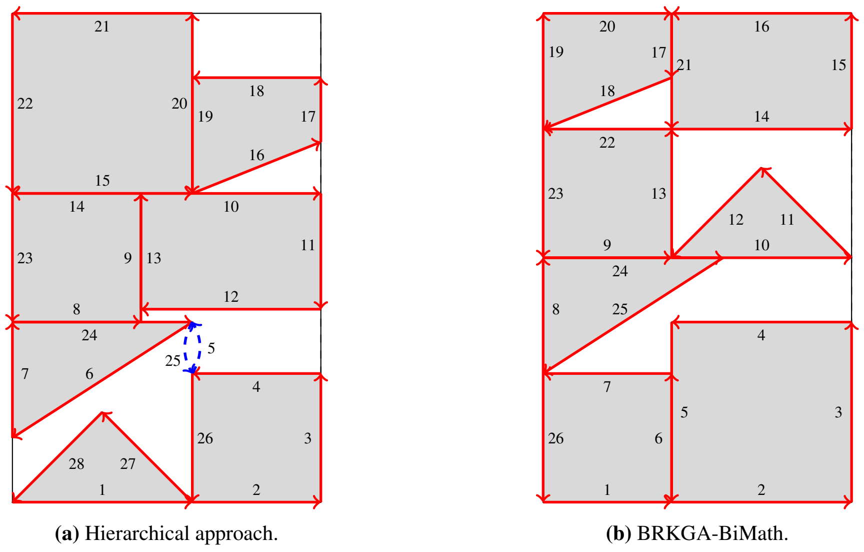


Figure 11 – Layout and cutting path for instance fu7.

The results are presented in Table 10. The first column contains the instances name, the second presents the layout length (Nest.) and the third presents the cutting path distance (CP). The symbol "-" indicates that no feasible solution was found.

Analyzing Table 10, the BRKGA-BiMath obtained solutions to instances with up to 15 pieces. For instances with more than 18 pieces, the BRKGA-BiMath was not able to evolve into a feasible layout within the time limit. Together with Phase II results, we conclude that BRKGA-BiMath can solve instances with up to 15 pieces and cannot find a solution for instances with more than 18 pieces.

5 CONCLUSIONS AND FURTHER RESEARCH

The cutting and packing problem appears in several production processes in which larger objects (boards) must be cut into smaller objects (pieces). The irregular strip packing problem studied in this paper consists of finding a layout of irregular pieces into a regular board of fixed width and unlimited length, to minimize the layout length.

In this paper, we developed a biobjective matheuristic based on BRKGA, the BRKGA-BiMath, for solving the integrated irregular strip packing and the cutting path determination problems. The BRKGA-BiMath is the first heuristic proposed to solve this version of the problem. In BRKGA, each solution is represented by a vector of n random keys. The transformation of this vector into a solution to the problem is done by a decoder. In this approach, the BRKGA is used to select variables which will reduce the complexity of the nesting model, converting it into a linear model that is easier to solve. In this model, the non-overlap constraints are written based on the nofit polygons (NFP) edges between a pair of pieces. After obtaining a layout, a second mathematical model minimizes the cutting path distance for feasible layouts.

The computational results showed the influence of the layout in the determination of the cutting path, i.e., in general, when the layout length increases the optimal solution for the cutting path distance decreases. BRKGA-BiMath was able to obtain a first frontier solution. In contrast, the hierarchical approach, i.e.,

Table 10 – First rank frontier for BRKGA-BiMath with Strategy 2.

Instances	**Nest.**	**CP**
three	6.20	2.26
	6.67	1.34
	7.00	**0.00**
threep2	9.67	4.00
	10.00	3.41
	10.67	2.00
	11.00	**0.00**
threep3	14.25	10.26
	14.33	6.91
	15.67	4.16
	16.05	3.83
	17.00	**0.00**
threep4	20.67	7.87
	22.00	6.00
	22.67	4.00
	22.83	3.92
	23.47	2.92
	24.67	1.34
threep5	32.67	14.14
	38.67	11.08
	39.67	11.06
	43.33	9.81
threep6	-	-
threep7	-	-
threep7	-	-
threep9	-	-
threep10	-	-

solving one problem at each time and in sequence, obtained just one solution, always focused on a low-length layout. A layout with a shorter cutting path distance may improve the productivity of a workday since the time saved during the cutting process may be used to cut more layouts. On the other hand, this leads to more material waste. With these solutions at hand, the decision-maker can choose the most suitable solution according to the production situation at the moment.

The results also showed that even in cases where instances do not present a trade-off behavior between the layout length and the cutting path distance, it is worth using the BRKGA-BiMath because it provides a layout with a shorter cutting path when compared to the hierarchical approach. In other words, BRKGA-BiMath obtained a layout with the same length as the hierarchical approach but with a shorter cutting path distance. Therefore, we conclude that the BRKGA-BiMath can solve both the irregular strip packing and the cutting path determination problems.

Due to our choice of a chromosome, i.e., each allele representing an edge from NFP_{ij}^{p}, infeasible layouts are obtained by the nesting model decoder throughout the evolution process. This choice negatively affected the BRKGA-BiMath performance. Meanwhile, the method solves instances with up to 15 pieces. A different chromosome choice, which does not lead to infeasible layouts, may provide even better results. In future works, we intend to investigate different chromosomes and different ways of decoding them into layouts, in order to be able to find a solution for instances with more than 15 pieces.

Acknowledgements

This research was funded by São Paulo Research Foundation (FAPESP) grants #2013/25743-6, #2016/09476-6, #2010/10133-0 and #2013/07375-0, National Council for Scientific and Technological Development (CNPq), grant #309161/2022-3 and CAPES(A026/2013) from Brazil. This work is a result of Agenda "PRODUTECH R3 - Agenda Mobilizadora da Fileira das Tecnologias de Produção para a Reindustrialização", nr. C645808870-00000067, investment project nr. 60, financed by the Recovery and Resilience Plan (PRR) and by European Union - NextGeneration EU.

References

ÁLVAREZ-VALDÉS R, CARRAVILLA MA & OLIVEIRA JF. 2018. *Cutting and Packing*. pp. 931–977. In: Martí R., Pardalos P., Resende M. (eds) Handbook of Heuristics. Springer, Cham.

AMARO-JÚNIOR B, PINHEIRO PR & COELHO PV. 2017. A Parallel Biased Random-Key Genetic Algorithm with Multiple Populations Applied to Irregular Strip Packing Problems. *Mathematical Problems in Engineering*, **2017**: 1–11.

ANAND KV & BABU AR. 2015. Heuristic and genetic approach for nesting of two-dimensional rectangular shaped parts with common cutting edge concept for laser cutting and profile blanking processes. *Computers & Industrial Engineering*, **80**: 111–124.

ANAND KV & UDHAYAKUMAR S. 2019. Design of adaptable pin configuration machine bed optimized with genetic approach for sheet metal cutting process. *Journal of Intelligent Manufacturing*, **30**(3): 1319–1333.

ANDRADE CE, SILVA T & PESSOA LS. 2019. Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. *Expert Systems with Applications*, **128**: 67–80.

BEAN JC. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, **6**(2): 154–160.

BENNELL JA & OLIVEIRA JF. 2008. The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, **184**(2): 397–415.

BENNELL JA & OLIVEIRA JF. 2009. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, **60**(1): 93–105.

BIAJOLI FL, CHAVES AA & LORENA LAN. 2019. A biased random-key genetic algorithm for the two-stage capacitated facility location problem. *Expert Systems with Applications*, **115**: 418–426.

CHERRI LH, MUNDIM LR, ANDRETTA M, TOLEDO FM, OLIVEIRA JF & CARRAVILLA MA. 2016. Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, **253**(3): 570–583.

CORRECHER JF & ALVAREZ-VALDES R. 2017. A biased random-key genetic algorithm for the time-invariant berth allocation and quay crane assignment problem. *Expert Systems with Applications*, **89**: 112–128.

DEB K, PRATAP A, AGARWAL S & MEYARIVAN T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**(2): 182–197.

DEWIL R, VANSTEENWEGEN P & CATTRYSSE D. 2011. Cutting path optimization using tabu search. *Key Engineering Materials*, **473**: 739–748.

DEWIL R, VANSTEENWEGEN P & CATTRYSSE D. 2014. Construction heuristics for generating tool paths for laser cutters. *International Journal of Production Research*, **52**(20): 5965–5984.

FONSECA CM, PAQUETE L & LÓPEZ-IBÁÑEZ M. 2006. An improved dimension - sweep algorithm for the hypervolume indicator. *In Proceedings of the 2006 Congress on Evolutionary Computation (CEC'06)*, .

FOWLER RJ, PATERSON MS & TANIMOTO SL. 1981. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, **12**(3): 133–137.

GONÇALVES JF & RESENDE MGC. 2011. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, **17**(5): 487 – 525.

HAN GC & NA SJ. 1999. A study on torch path planning in laser cutting processes part 2: Cutting path optimization using simulated annealing. *Journal of Manufacturing Systems*, **18**(2, Supplement 1): 62–70.

IMAHORI S, KUSHIYA M, NAKASHIMA T & SUGIHARA K. 2008. Generation of cutter paths for hard material in wire EDM. *Journal of Materials Processing Technology*, **206**(1-3): 453–461.

LEE MK & KWON KB. 2006. Cutting path optimization in CNC cutting processes using a two-step genetic algorithm. *International Journal of Production Research*, **44**(24): 5307 –5326.

LEÃO AA, TOLEDO FM, OLIVEIRA JF, CARRAVILLA MA & ALVAREZ-VALDÉS R. 2020. Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, **282**(3): 803–822.

LÓPEZ-IBÁÑEZ M, DUBOIS-LACOSTE J, CÁCERES LP, BIRATTARI M & STÜTZLE T. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, **3**: 43–58.

MANBER U & ISRANI S. 1984. Pierce point minimization and optimal torch path determination in flame cutting. *Journal of Manufacturing Systems*, **3**(1): 81–89.

MOREIRA LM, OLIVEIRA JF, GOMES AM & FERREIRA JS. 2007. Heuristics for a dynamic rural postman problem. *Computers & Operations Research*, **34**(11): 3281–3294.

MUNDIM LR, ANDRETTA M & DE QUEIROZ TA. 2017. A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. *Expert Systems with Applications*, **81**: 358–371.

MUNDIM LR, QUEIROZ T & ANDRETTA M. 2016. O BRKGA aplicado em problemas de corte de itens irregulares em um único recipiente. *Matemática aplicada à indústria: problemas e métodos de solução. São Paulo: Blucher*, **0**: 111–136.

OLIVEIRA LT, SILVA EF, OLIVEIRA JF & TOLEDO FMB. 2020. Integrating irregular strip packing and cutting path determination problems: A discrete exact approach. *Computers & Industrial Engineering*, **149**: 106757.

POTT A & GLAAB H. 2003. Optimization Problems in a Semi-Automatic Device for Cutting Leather. In: *Mathematics - Key Technology for the Future*. pp. 609–622. Springer Berlin Heidelberg.

RODRIGUES AM & FERREIRA JS. 2011. Cutting path as a Rural Postman Problem: solutions by Memetic Algorithms. *International Journal of Combinatorial Optimization Problems and Informatics*, **3**(1): 31–46.

ROQUE L, FONTES D & FONTES F. 2017a. A multi-objective unit commitment problem combining economic and environmental criteria in a metaheuristic approach. *Energy Procedia*, **136**: 362–368.

ROQUE LAC, FONTES DBMM & FONTES FACC. 2017b. A Metaheuristic Approach to the Multi-Objective Unit Commitment Problem Combining Economic and Environmental Criteria. *Energies*, **10**(12): 1–25.

SHERIF SU, JAWAHAR N & BALAMURALI M. 2014. Sequential optimization approach for nesting and cutting sequence in laser cutting. *Journal of Manufacturing Systems*, **33**(4): 624–638.

SILVA EF, OLIVEIRA LT, OLIVEIRA JF & TOLEDO FMB. 2019. Exact approaches for the cutting path determination problem. *Computers & Operations Research*, **112**: 104772.

TANGPATTANAKUL P, JOZEFOWIEZ N & LOPEZ P. 2013. Biased random key genetic algorithm with hybrid decoding for multi-objective optimization. In: *2013 Federated Conference on Computer Science and Information Systems*. pp. 393–400.

TOSO RF & RESENDE MG. 2012. A C++ application programming interface for biased random-key genetic algorithms. Technical report. AT&T Labs Research, Florham Park, New Jersey.

WÄSCHER G, HAUSSNER H & SCHUMANN H. 2007. An improved typology of cutting and packing problems. *European Journal of Operational Research*, **183**(3): 1109–1130.

How to cite
OLIVEIRA LT, CARRAVILLA MA, OLIVEIRA JF & TOLEDO FMB. 2023. A biobjective matheuristic for the integrated solution of the irregular strip packing and the cutting path determination problems. *Pesquisa Operacional*, **43**: e275212. doi: 10.1590/0101-7438.2023.043.00275212.