European Journal of Operational Research 279 (2019) 657–671

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

Raster penetration map applied to the irregular packing problem

André Kubagawa Sato[a], Thiago Castro Martins[a], Antonio Miguel Gomes[b], Marcos Sales Guerra Tsuzuki[a,*]

[a] Computational Geometry Laboratory, Department of Mechatronics and Mechanical Systems Engineering, Escola Politécnica da Universidade de São Paulo, Av. Prof. Mello Moraes, São Paulo 2231, Brazil
[b] INESC-TEC, Universidade do Porto, Porto, Portugal

ARTICLE INFO

Article history:
Received 10 April 2018
Accepted 3 June 2019
Available online 8 June 2019

Keywords:
Cutting
Irregular packing problem
Separation and compaction
Penetration depth
Raster representation

ABSTRACT

Among the most complex problems in the field of 2-dimensional cutting & packing are irregular packing problems, in which items may have a more complex geometry. These problems are prominent in several areas, including, but not limited to, the textile, shipbuilding and leather industries. They consist in placing a set of items, whose geometry is often represented by simple polygons, into one or more containers such that there is no overlap between items and the utility rate of the container is maximized. In this work, the irregular strip packing problem, an irregular packing variant with a variable length container, is investigated. The placement space is reduced by adopting a rectangular grid and a full search is performed using preprocessed raster penetration maps to efficiently determine the new position of an item. Tests were performed using simple dotted board model cases and irregular strip packing benchmark instances. The comparison of our results with the state of the art solutions showed that the proposed algorithm is very competitive, achieving better or equal compaction in 9 out of 15 instances and improving the average density in 13 instances. Besides the contribution of the new best results, the proposed approach showed the advantage of adopting discrete placement, which can be potentially applied to other irregular packing problems.

1. Introduction

Cutting and packing problems are important in different areas, such as the textile, wood, glass, and shipbuilding industries and are thus regarded as an important research field. Finding efficient solutions for these problems may generate significant economic and environmental benefits. The basic outline of a packing (or cutting) problem consists in assigning items to containers and arranging the layout such that the combined utility rate of the containers is maximized.

Irregular strip packing, also referred to as the irregular nesting problem, is studied herein. Proposals to solve this subgroup of cutting and packing problems must be capable of dealing with irregular shaped items, which is a complex task. The objective is to obtain the most compact configuration of items that can be placed into a rectangle container with a variable dimension. Due to the limitations of practical applications, each item can only be rotated by a finite set of angles, which may include 0°, 90°, 180° and 270°. According to the typology proposed by Wäscher, Haußner, and Schumann (2007), the variant considered herein is the 2-dimensional irregular open dimension problem.

In a textile application, using automatic computer-generated cutting patterns that minimize the volume of wasted material frequently provides a huge improvement over manually developing a layout. However, as the problem is NP-hard (Fowler, Paterson, & Tanimoto, 1981), it might not be possible to obtain the optimal layout in a limited time frame for large problems. Therefore, for solving a wider range of problems, a heuristic approach is adopted. To achieve a more efficient solution, the search space is reduced by adopting a grid placement for items and the layout is determined by an overlap minimization strategy. Moreover, the raster penetration map tool is proposed to reduce the online computational cost of computing the overlap value. So as to properly test the quality of the layouts, the proposed solution is executed by using dotted board cases and irregular strip packing instances from the literature. Even with the placement limitations, the results show that the algorithm is capable of achieving better solutions than the best approaches in the literature.

This text is structured as follows. Section 2 contains a brief literature review on irregular strip packing, which is described in detail in Section 3. Section 4 describes the proposed method to evaluate the overlap efficiently with the raster penetration map.

* Corresponding author.
E-mail address: mtsuzuki@usp.br (M.S.G. Tsuzuki).

https://doi.org/10.1016/j.ejor.2019.06.008
0377-2217/

This evaluation is employed in a separation and compaction strategy, described in Section 5, to solve the strip packing problem. Section 6 shows the results from the proposed algorithm ROMA and a comparison with results from state-of-the-art approaches in the literature is made. Finally, the conclusions are drawn in Section 7.

2. Literature review

The irregularity of shapes is a singular characteristic in irregular strip packing problems, which leads to a reduced number of publications in the field (Bennell & Oliveira, 2008). Due to its complexity, advanced geometric tools are employed to avoid overlap between items. In the literature, the main geometric tool is the nofit polygon, which was proposed by Art (1966). Bennell and Oliveira (2008) made a review of geometric tools for irregular packing problems.

Aside from the geometric restriction, the main challenge to solve the irregular strip packing problems is to develop an efficient search strategy, i.e., an algorithm that consistently generates layouts with high compaction in a small time frame. Mathematical models are capable of finding the optimal solution for small problems or instances with some special characteristics. Alvarez-Valdes, Martinez, and Tamarit (2013) developed a branch and bound algorithm based on a mixed integer formulation and were able to solve instances of up to 12 items. Cherri et al. (2016) adopted a robust mixed-integer linear programming model by convex decomposing items, which showed slightly improved results, which was later improved by Rodrigues, Cherri, and Mundim (2017) with new symmetry breaking constraints. Larger instances were optimally solved by Toledo, Carravilla, Ribeiro, Oliveira, and Gomes (2013), but with placement and item assortment limitations. Rotations are usually not considered in most exact solutions. Two exceptions include the works by Jones (2014) and Wang, Hanselman, and Gounaris (2018), which use a circle covering representation of the shapes and find optimal layout with up to five items.

Heuristic solutions are often employed to obtain solutions for medium and larger instances with discrete rotations. There are two main strategies employed in the literature: the search over the sequence and the search over the layout. The main difference resides in the layout representation: in the former case, the layout is represented by a sequence of items, whereas in the latter, the position of each item is directly represented by a pair of coordinates (Bennell & Oliveira, 2009). This directly impacts the search strategy. Search over the sequence solutions insert items sequentially into the container, without overlap. In the case of the search over the layout, items move freely and a separation method is usually employed.

The most popular constructive heuristic for search over the sequence uses the bottom-left policy. A greedy implementation of the bottom-left heuristic consists in alternating vertical and horizontal translations until the item is unable to move. Then, the challenge is to determine the placement sequence. Pinheiro, Amaro Júnior, and Saraiva (2016) adopted a random-key genetic algorithm to control the sequence and item rotation and a parallel implementation of this strategy was developed by Amaro Júnior, Pinheiro, and Coelho (2017). A constrained placement rule was enforced by Mundim, Andretta, and de Queiroz (2017) to reduce the search space and to accelerate collision detection using the no-fit raster. Burke, Hellier, Kendall, and Whitwell (2006) modified the greedy bottom-left layout construction by discretizing the horizontal search and applied a hill-climbing tabu search. There are a few strategies in the literature that employ other constructive heuristics (Oliveira, Gomes, & Ferreira, 2000; Sato, Martins, & Tsuzuki, 2012; 2015; Xu, Wu, Liu, & Zhang, 2017).

Most efficient solutions in the literature adopt a separation and compaction technique, which utilizes the search over the layout strategy. Two algorithms are usually applied to obtain a compact valid layout: separation and compaction. Li and Milenkovic (1995) proposed a position-based optimization model for compacting and separating irregular packing layouts. The nofit polygon is used to generate artificial constraints to generate a convex feasible solution space. This model was adopted by Bennell and Dowsland (2001) to generate valid layouts using a tabu search heuristic. Gomes and Oliveira (2006) hybridized the compaction and separation algorithms with a simulated annealing algorithm. At each iteration, the separation and compaction algorithms are executed sequentially, generating a new layout. The meta-heuristic simulated annealing is applied to escape local minima.

2.1. Overlap minimization approach

One of the major difficulties when adopting a search over the layout strategy is to guarantee the feasibility of the final layout. Overlap minimization techniques circumvent the issue by fixing the length of the container, simplifying the separation and compaction procedures. The layout compaction is then achieved by sequentially reducing the area of the container. Then, an overlap function is defined to relax the geometric restrictions.

Bennell and A. Dowsland (2010) proposed an algorithm which eliminates overlap by applying a tabu thresholding heuristic, in which the overlap function was the minimum horizontal distance separation. Egeblad, Nielsen, and Odgaard (2007) proposed an overlap minimization algorithm based on a method proposed by Faroe, Pisinger, and Zachariasen (2003) to solve a bin packing problem. At each iteration, a one-dimensional evaluation of the overlap function is performed to determine the translation for each item. Umetani et al. (2009) proposed the use of the directional penetration depth to measure the overlap between items and applied it to a guided local search heuristic. Imamichi, Yagiura, and Nagamochi (2009) and Leung, Lin, and Zhang (2012) solved the overlap minimization problem by using a non-linear programming model, moving all items simultaneously. Elkeran (2013) proposed the use of the meta-heuristic cuckoo search to perform a two-dimensional search for the least overlapping placement for each item. Wang et al. (2018) applied an ant colony flexible labour division to assign up to three actions for each item. The popularity of overlap minimization solutions in the literature is due to the consistency and quality of their results.

2.2. Dotted board model

Allowing for the free movement of items inside the container creates a large continuous solution space for the irregular strip packing problem. This complexity is one of the reasons for the difficulty in developing mathematical solvers to obtain optimum solutions. In order to circumvent this limitation, Toledo et al. (2013) proposed the dotted board model, in which the placement space is restricted by a grid. A similar constraint was already considered by Bennell and A. Dowsland (2010), which limited the neighborhood of the metaheuristic to the movement of an item within the grid.

In this work, we solved the dotted board model by using a heuristic approach based on the overlap minimization strategy. A grid space is adopted to allow a more efficient search for a new candidate when inserting an item into the layout. The proposed approach updates the solution from Bennell and A. Dowsland (2010) by applying the guided local search metaheuristic, which has gained popularity among some best algorithms in the literature (Elkeran, 2013; Umetani et al., 2009). It also develops and

applies the raster overlap map, which transfers computational load to a preprocessing stage.

3. Problem description

The irregular strip packing problem is a two-dimensional packing problem in which a set of items must be placed into one rectangular container in a valid configuration. The shape of items may be irregular and is herein represented by simple polygons. The container has a fixed width and a variable length. The objective is to obtain a feasible solution which maximizes the density of the configuration, i.e., minimizes the length of the layout. Up to four orientations are allowed for each item: 0°, 90°, 180° and 270°. The geometric constraints are: (1) an item must not overlap other items; and (2) items must lie completely inside the container.

A polygon P, with m edges, can be represented by an ordered list of its vertices $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$. Consider a set of simple polygons $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, which represents the geometry of the items. The container is represented by a semi finite rectangle $\mathcal{C}$, with the fixed dimension parallel to the y axis. When necessary, an item with a specific orientation is expressed as $P(o)$, where o is the angle of rotation of P. Operator $\imath(P)$ represents only the interior of P and $P \oplus v$ corresponds to applying a translation vector v to P. $\mathcal{O}$ is the set of admissible orientations for the items, which are typically multiples of 90°. A solution to the problem is described by a set of translation vectors $\{v_1, v_2, \ldots, v_n\}$ and a set of orientations $\{o_1, o_2, \ldots, o_n\}$. L is the length of the smallest container which contains all the items. The irregular strip packing problem can be described as

$$\begin{aligned}
\text{minimize} \quad & L \\
\text{subject to} \quad & \imath(P_i(o_i) \oplus v_i) \cap (P_j(o_j) \oplus v_j) = \emptyset, && 1 \le i < j \le n \\
& (P_i(o_i) \oplus v_i) \subseteq \mathcal{C}, && 1 \le i \le n \\
& o_i \in \mathcal{O}, && 1 \le i \le n \\
& v_i \in \Re^2, && 1 \le i \le n \\
& L \in \Re_+. && (1)
\end{aligned}$$

3.1. Overlap minimization problem

In overlap minimization techniques, the container has fixed dimension and an iterated local search algorithm is usually adopted to solve the original problem. At each iteration, the length of the container is changed and a feasible layout is searched by relaxing the no-overlap restriction. This is achieved by defining an overlap function and can be described as

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} f(P_i(o_i) \oplus v_i, P_j(o_j) \oplus v_j) \\
\text{subject to} \quad & (P_i(o_i) \oplus v_i) \subseteq \mathcal{C}, && 1 \le i \le n \\
& o_i \in \mathcal{O}, && 1 \le i \le n \\
& v_i \in \Re^2. && 1 \le i \le n \\
& && (2)
\end{aligned}$$

where f is the overlap function for a pair of items. Its value must be zero when they are separated, and positive otherwise. Thus, when the cost function is zero, it implies that all the overlap functions are zero; therefore, the layout has no overlap.

3.2. The dotted board model

The dotted board model limits the placement of items by using a rectangular grid. Therefore, the coordinates of the translation vectors v_i are expressed in grid dimensions. If g is the square grid step, then

$$v_i \in \{(x, y) \mid x = k \cdot g,\ y = l \cdot g\} \tag{3}$$

where $k, l \in \mathbb{Z}$.

4. Overlap evaluation

A common strategy is to relax the overlap restriction by penalizing the objective function. In overlap minimization solutions, the complexity of the problem is reduced even more by fixing the dimension of the container. Thus, the objective function is reduced to the penalization function, which is defined by the overlap function. The selection of this function is an important factor when developing a solution.

As items are allowed to move freely, the search space is continuous and, consequently, more difficult to explore even by heuristic approaches. To reduce the solution space, a placement grid, which limits the choices for positioning items inside the container, is adopted here. Thus, the overlap function only evaluates the objective function at grid positions. This is efficiently achieved by the use of a novel geometric tool: the raster penetration map.

4.1. Basic geometric concepts

The basic geometric tools are presented herein, as these concepts facilitate the understanding of the employed adapted geometric tools. The most popular geometric tool to detect overlap between a pair of irregular items is the nofit polygon.

For two items, the nofit polygon describes all the overlapping configurations. In order to determine the nofit polygon, one of the items is classified as the fixed item and the other as the movable item. The nofit polygon represents all the translations that, when applied to the movable item, causes it to overlap the fixed item (see Fig. 1(a)).

The other important geometric tool is the inner-fit polygon, which is derived from the nofit polygon. It represents the translations that place the movable item completely inside the container. By defining a reference point, the inner-fit polygon can be mapped into a region in space, as shown in Fig. 1(b).

In most cases, overlap minimization techniques do not relax the container protrusion restriction, with some exceptions (Imamichi et al., 2009; Leung et al., 2012). Therefore, the inner-fit concept is employed directly to ensure that all items are completely inside the container. In overlap minimization techniques, the nofit polygon is often used to aid the determination of the overlap function.

4.2. Geometric tools for the dotted board model

In the dotted board model, the placements of items are limited to discrete points in space. The set of points are limited to a uniform distribution, creating a square grid. Moreover, to simplify mathematical formulations, the reference point of the items is always set at the origin and, therefore, coincides with a grid point.

Both the nofit polygon and the inner-fit polygon can be directly applied to solve the dotted board model problem. However, as the placement is discrete, only a finite subset of the translations described by the geometric tools is needed. Therefore, adapted versions of the geometric tools are employed here.

The discrete nofit polygon represents all the grid translations that, when applied to the movable item, causes it to overlap the fixed item. It is a finite subset of the nofit polygon, as can be observed in Fig. 2(a). The discrete inner-fit polygon describes all the possible grid placements for a given item (see Fig. 2(b)). Given the set of grid points $\mathcal{G}$, the discrete inner-fit polygon induced by container $\mathcal{C}$ to movable item P, denoted $\Lambda(\mathcal{C}, P, \mathcal{G})$, can be defined as

$$\Lambda(\mathcal{C}, P, \mathcal{G}) = \{v \mid \forall\, \mathbf{a} \in \imath(P), \mathbf{a} + v \in \mathcal{C},\ v \in \mathcal{G}\}. \tag{4}$$

Note that both the discrete nofit polygon and the discrete inner-fit polygon can be obtained by rasterizing the nofit polygon and

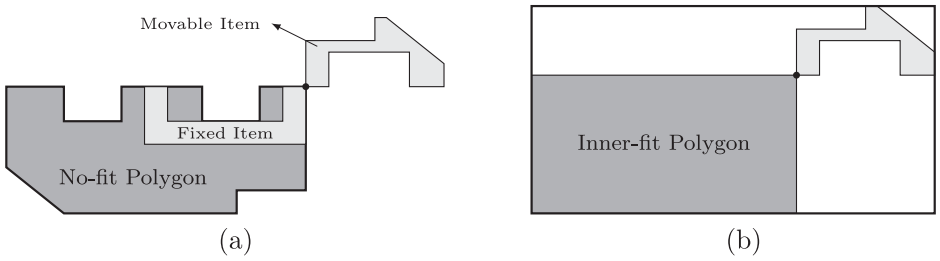

Fig. 1. Geometric tools examples: (a) nofit polygon and (b) inner-fit polygon.

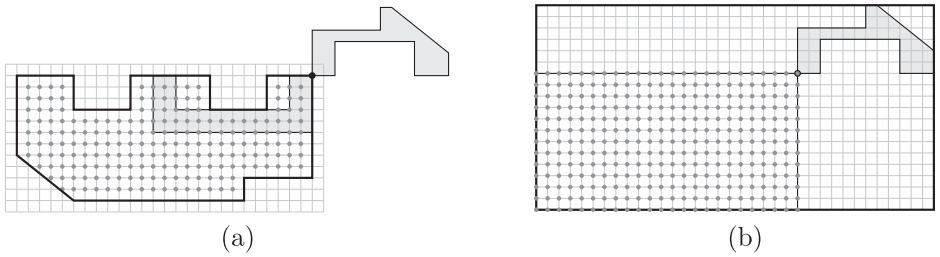

Fig. 2. Geometric tools for the dotted board: (a) discrete nofit polygon and (b) discrete inner-fit polygon.

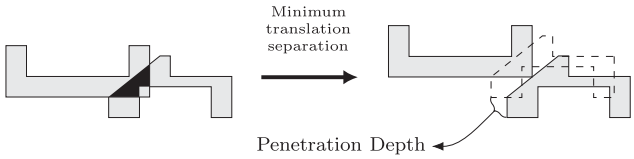

Fig. 3. Determination of the penetration depth of a pair of items.

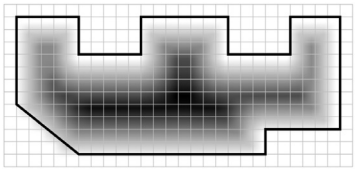

Fig. 4. Raster penetration map example.

the inner-fit polygon, respectively (see Fig. 2). This is achieved by treating each point in the grid as the center of a pixel or a binary matrix cell.

4.3. Raster penetration map

The proposed raster penetration map, which is the main tool adopted herein, builds upon the adapted geometric tools from the literature to efficiently determine the overlap. It retains the geometric collision precision and uses raster methods to accelerate the overlap determination while approximating the overlap value.

The overlap function is a collision penalty function defined for a pair of items. As such, it must be equal to zero when items are separated or touching and, when there is overlap, it must be higher than zero.

The penetration depth is frequently adopted as the overlap function in overlap minimization solutions. It corresponds to the norm of the minimum translation needed to separate a pair of items. Then, given two items P_i and P_j, the penetration depth $\delta(P_i, P_j)$ can be expressed as

$$\delta(P_i, P_j) = \min\left\{\|v\| \mid \imath(P_j \oplus v) \cap P_i = \emptyset\right\}. \quad (5)$$

When the items are not overlapping, the penetration depth is zero. Fig. 3 exemplifies the penetration depth calculation. The penetration can be determined using the nofit polygon to aid the overlap detection.

The raster penetration depth is an approximation of the penetration depth and defines the minimum translation, in grid units, to a grid point to separate a pair of items. The discrete nofit polygon can be employed to compute the raster penetration depth. It is equivalent to the number of grid steps that correspond to the distance from the current reference point position to the closest grid point not contained in the discrete nofit polygon. By multiplying the results by the grid size, it can be converted to an approximation of the penetration depth. The raster penetration map defines, in grid space, the value of the raster penetration depth of a pair of items for all the possible configurations. Fig. 4 shows an example of a raster penetration map.

The first step to obtain the raster penetration map is to determine the nofit polygon for the item pair. Then, it is rasterized according to the adopted grid, generating the discrete nofit polygon. Note that grid points on the contour of the nofit polygon should not be included in the rasterization results. Finally, to obtain the final map, the raster penetration depth must be determined for each marked grid point. It can be achieved by using the distance transform, which can be conducted in linear time in relation to the number of grid locations (Felzenszwalb & Huttenlocher, 2004).

The determination of the raster penetration map is a three-step process. The entire process is illustrated by the example in Fig. 5. Once determined, the raster penetration map can be reused by applying the translation corresponding to the fixed item, similar to the nofit polygon. Therefore, by calculating all the raster penetration maps in a preprocessing stage, the approximated penetration depth can be directly obtained from the raster penetration map.

Raster methods are usually employed in irregular packing problems to simplify the geometry of the items. The main drawback of this approach is the low precision achieved for the overlap detection. Furthermore, the geometric tools available are very efficient to deal with simple polygons. In the case of the raster penetration map and the discrete inner-fit polygon, the rasterization process does not impact the precision of the overlap evaluation. As these concepts are used as a foundation for the method introduced in this work, the collision precision of the proposed approach is equivalent to nofit polygon-based algorithms.

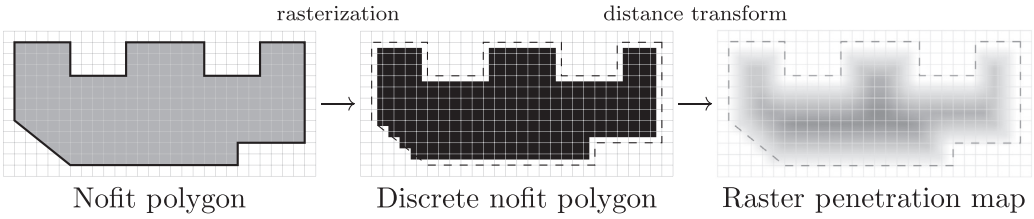

Fig. 5. Raster penetration map determination.

5. Separation and compaction algorithm

This section describes the proposed separation and compaction algorithm, which uses the search over the layout strategy to solve the irregular packing problem. The main challenge that search over the layout approaches face is the simultaneous management of two objectives: overlap elimination and layout compaction. The proposed approach adopts the overlap minimization strategy, which avoids this difficulty by fixing the length of the container, thus creating a pure separation sub-problem.

The objective when solving a strip packing problem is to obtain the layout with the highest density. In this work, the separation and compaction technique adopted is very straightforward: when a separation is successful, the container length is reduced; otherwise, the container is expanded. Parameters r_{dec} and r_{inc} control the reduction and expansion of the container, respectively. In order to encourage compaction, r_{dec} is usually higher than r_{inc}. The length value is rounded to the nearest grid point. Algorithm 1 details the separation and compaction technique. The round function rounds the number according to grid step g given by the second parameter.

Algorithm 1 SeparateAndCompact.

```
(x, o) ← <generate bottom left layout>
L, L_best ← <container length>
while <time limit not reached> do
    if SeparateLayout(v, o) = successful then
        L_best ← round(L, g)
        L ← (1 − r_dec) · L
    else
        if (1 + r_inc) · L < L_best then
            L ← (1 + r_inc) · L
        else
            L ← (L + L_best)/2
        end if
    end if
    if round(L) ≥ L_best then
        L ← L_best − g
    end if
    <container length> ← round(L, g)
end while
```

A modification proposed here is to only allow lengths that are inferior to the best feasible solution found. The aim is to reduce the exploration of less compact solutions. When expanding the container, if it infringes this restriction, the midway point between the current and the best length is adopted. Due to the grid constraint, it is possible that the value is rounded to the best length. In this case, the length is reduced by one grid step.

When the length of the container is reduced, some items of the layout may protrude from the new container. In this case, before the next iteration of the algorithm, all the protruding items are translated horizontally into new positions inside the container. This behavior of the algorithm is displayed in the example in Fig. 6.

So as to apply the compaction technique, an initial length must be determined. We employed a bottom left heuristic to obtain a feasible initial layout. The initial solution is constructed using a sequential placement with a random ordering of items. For each item, discrete inner-fit polygon points are evaluated for overlap in a bottom-up, left-to-right sequence. When a position with zero overlap is found, the item is placed and the next item is processed. After all the items are placed, the container length is set to the length of the layout.

5.1. Separation algorithm

Heuristic approaches usually modify the layout by applying one or two operators at a time, with some exceptions (Imamichi et al., 2009; Leung et al., 2012). The two main operators are swapping the position of two items (swap operator) and moving a single item (insert operator). In both cases, an operator that changes the orientation of the item is applied in conjunction with the main modification.

When an overlap function is adopted, it is possible to assess the quality of a single item placement by analyzing its overlap value. By using this property, insert operators can be considered the best option to generate a new solution, as they handle each item individually. A straightforward local search based on this operator usually quickly leads to local minimum solutions. The solution adopted herein is to use the guided local search metaheuristic.

5.1.1. Guided local search separation

We adopted the guided local search used by Umetani et al. (2009) to avoid local minimum solutions. It applies local modifications to the objective function by using weights. The modified total overlap function can be expressed as

$$\mathcal{F}'(v,o)=\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{ij}\cdot\delta(P_i(o_i)\oplus v_i, P_j(o_j)\oplus v_j)=\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{ij}\cdot f_{ij} \tag{6}$$

where w_{ij} and f_{ij} are the weight and overlap associated with the item pair (P_i, P_j), respectively.

At the start of the packing algorithm, the value of all weights is 1. In order to escape local minimum configurations, after each generation of a new solution, the weights are updated using the following rule

$$w_{ij}=w_{ij}+\frac{f_{ij}}{\max_{k,l}(f_{kl})} \tag{7}$$

where $1\le k,\ l\le n$. Algorithm 2 details the proposed raster guided local search separation. At each iteration, the solution is modified so as to minimize the value of the modified total overlap function. The algorithm continues until a feasible solution is found, i.e. the total overlap is zero, or a number of iterations without improvement (N_{mi}) is reached.

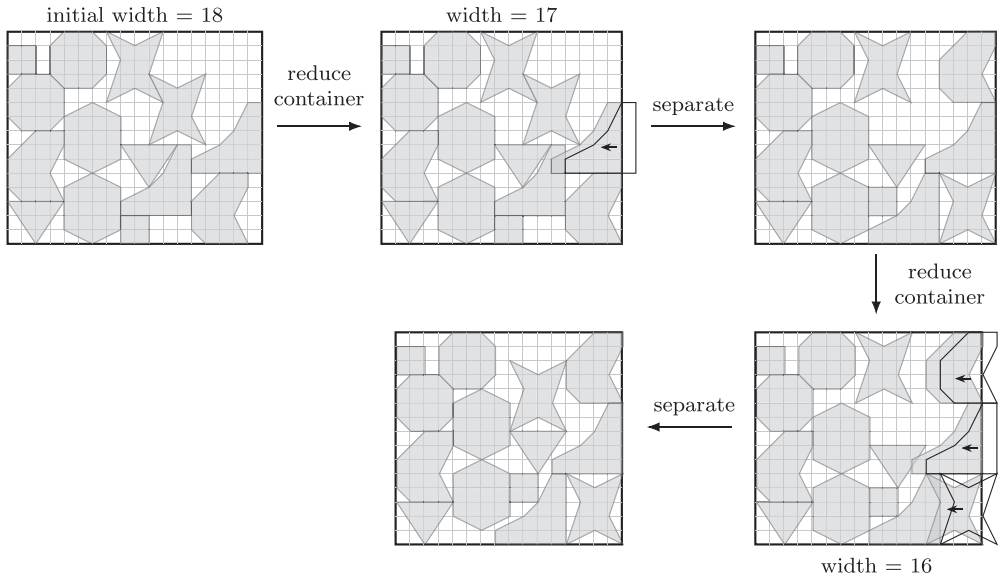

Fig. 6. Example of the compaction strategy (Imamichi et al., 2009).

Algorithm 2 SeparateLayout(v,o).

```
numIterations ← 0
minOverlap ← +∞
while numIterations ≤ N_mi do
    (v, o) ← ModifySolution(v, o)
    cost = Σ_{i=1}^{n} Σ_{j=i+1}^{n} f_ij
    if cost = 0 then
        return (x, o)
    end if
    if cost < minOverlap then
        minOverlap = cost
        numIterations = 0
    else
        numIterations ← numIterations + 1
    end if
    w_ij ← <updated weigths>
end while
```

5.1.2. Modifying the solution and the obstruction map

The solution is modified by translating one item at a time, reducing the overall overlap value of the layout. In order to aid the search for a new position, the item overlap function is defined as the sum of the overlap values between one item and the remaining items in the layout. Hence, an item overlap function $\mathcal{I}$ for an item P_i, translated by a vector z and rotated by an angle θ is defined as

$$\mathcal{I}(P_i, z, \theta, v, o) = \sum_{j=1, j\neq i}^{n} w_{ij} \cdot \delta(P_i(\theta) \oplus z, P_j(o_j) \oplus v_j). \quad (8)$$

Consider an item P_i and a container $\mathcal{C}$. The obstruction map associates all translations x contained in $\Lambda(\mathcal{C}, P_i)$ with its item overlap value given by $\mathcal{I}(P_i, x, o_i, v, o)$. It can be calculated by adding up all the corresponding raster penetration maps for a given item, restricting to positions from the discrete inner-fit polygon. This process is described in Algorithm 3 and illustrated in Fig. 7. Once the obstruction map is fully determined, all the positions are evaluated to define the minimum overlap placement, as shown in Fig. 8. The main advantage of the obstruction map proposed herein is that it allows a complete search over the placement space for every insert operator. Different approaches from the literature employ limited directional searches to determine a new position efficiently (Egeblad et al., 2007; Umetani et al., 2009) or adopt sub-optimal solutions (Elkeran, 2013; Imamichi et al., 2009; Leung et al., 2012).

Algorithm 3 CreateObstructionMap(P_i, θ, v, o, $\mathcal{G}$).

```
Map ← <new map filled with zeros>
for each P_k ∈ P where P_k ≠ P_i do
    for each pos ∈ G do
        if pos ∈ Λ(C, P_k, G) then
            Map{pos} ←Map{pos} + w_ik · δ(P_i(θ) ⊕ pos, P_k(o_k) ⊕ v_k)
        end if
    end for
end for
```

Algorithm 4 performs the proposed solution modification. Firstly, the items are randomly sequenced. For each item in an overlapped configuration, and for each admissible orientation, the obstruction map is created on line 8. After a map is constructed, a full search is performed to find the current minimum overlap value. When a minimum is found, the position and orientation of the item are immediately updated on line 11 and line 12, respectively. As the map construction does not depend on the current item (see Eq. (7)), it can be updated immediately and no other features need to be recalculated. As an improvement, the created obstruction maps can be stored in a cache and be later retrieved on line 8 to reduce the number of operations.

5.1.3. Multiresolution search

Finding the new position for an item has usually the highest computational cost among all the operations in an overlap minimization algorithm. As such, its performance heavily impacts the applicability of the proposed approach. When the placement of items is not continuous, the computational load is reduced by limiting the number of grid points. On the one hand, increasing the grid size in a raster approach limits the search space. On the other hand, it may adversely impact the final solution. Therefore, the multiresolution search is proposed to accelerate the search of the minimum obstruction map position while minimizing the impact on the quality of the final solution.

The multiresolution search is performed in two steps. Initially, an obstruction map with lower resolution, i.e. larger grid step, is

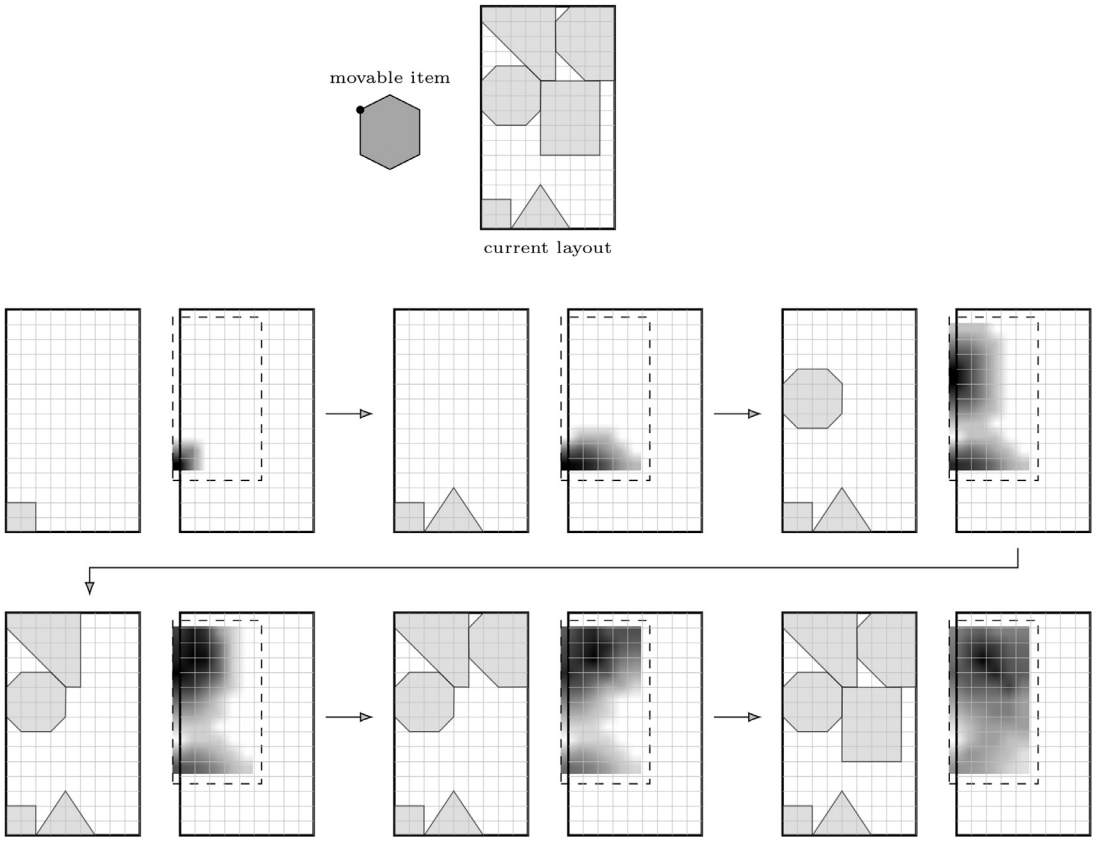

Fig. 7. Obstruction map determination. The item whose placement is being evaluated is the movable item.

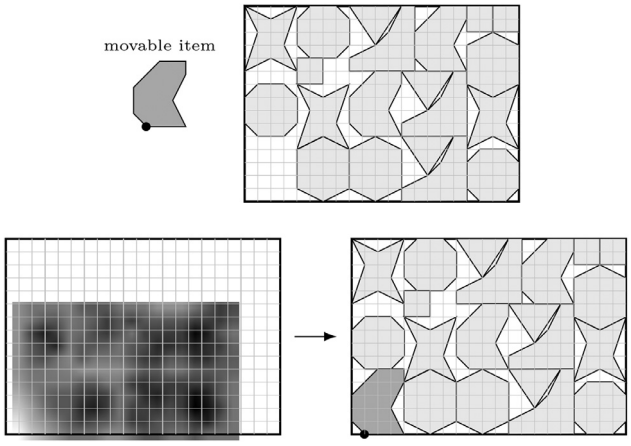

Fig. 8. Item placement using the obstruction map.

Algorithm 4 ModifySolution(v,o).

```
1: n ← number of items
2: Q ← random item sequence
3: for each P_k ∈ Q do
4:     currentOverlap ← Σ_{j=1, j≠k}^{n} f_kj
5:     if currentOverlap ≠ 0 then
6:         minOverlap = +∞
7:         for each orientation θ do
8:             M ← CreateObstructionMap(P_k, θ, v, o, G)
9:             for each (pos, overlap) ∈ M do
10:                if overlap < minOverlap then
11:                    v_k = pos
12:                    o_k = θ
13:                    minOverlap ← overlap
14:                end if
15:            end for
16:        end for
17:    end if
18: end for
```

created and its minimum position is determined. This position is defined as the center of a new reduced map with the original resolution, which is then searched to obtain the final item placement. As the higher resolution map is only locally determined and the lower resolution map is faster to determine, this search process can be significantly quicker than the original. Fig. 9 shows an example of a multiresolution search.

If the grid step for the lower resolution is a multiple of the original, the obstruction map search can be performed using the original set of raster penetration maps. Then, only a fraction of the original obstruction map is calculated at each iteration.

Algorithm 5, which replaces Algorithm 4, describes the multiresolution search using two resolutions, which have an integer ratio. There are two important parameters for the multiresolution search algorithm: lower resolution grid $\mathcal{G}_l$ and neighborhood size β for the refined search. The function SquareAt($posl, \beta, \mathcal{G}$) aggregates all the original grid points $\mathcal{G}$ contained inside a rectangle centered at *posl* with size β. If β is equal to the original grid step, the entire search space is considered and, thence, no placement is ignored.

Contrary to the original proposal, the multiresolution search does not guarantee that the global minimum grid position is obtained. However, the gain in performance may justify the adoption of the multiresolution approach, as more iterations are performed

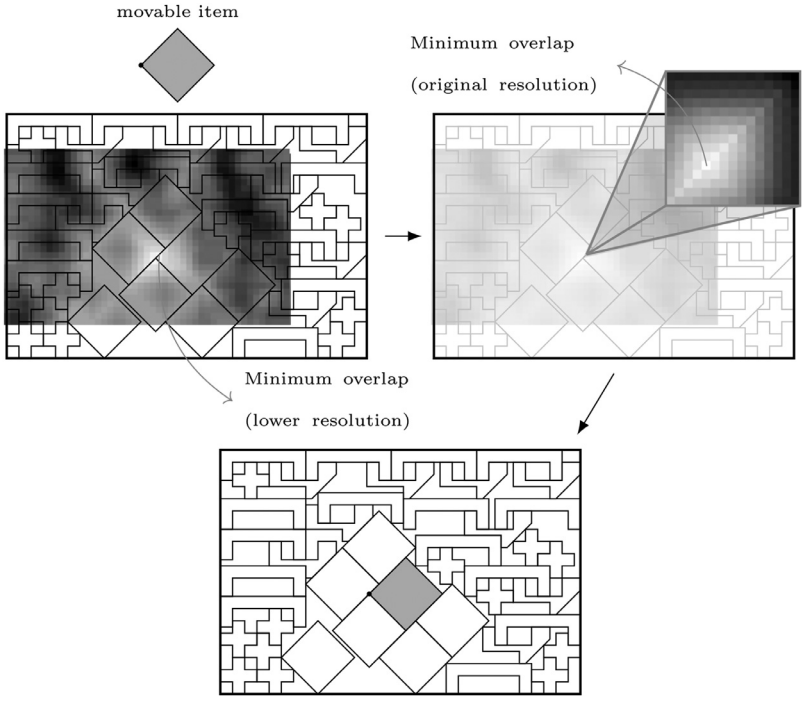

Fig. 9. Example of a multiresolution search.

Algorithm 5 FastModifySolution(v,o).

```
n ← number of items
Q ← random item sequence
for each P_k ∈ Q do
    currentOverlap ← Σ_{j=1, j≠k}^{n} f_kj
    if currentOverlap ≠ 0 then
        minOverlap = +∞
        for each orientation θ do
            minLowerOverlap = +∞
            M ← CreateObstructionMap(P_k, θ, v, o, G_l)
            for each (pos, overlap) ∈ M do
                if overlap < minLowerOverlap then
                    minLowerOverlap ← overlap
                    posl = pos
                end if
            end for
            for each pos ∈ SquareAt(posl, β, G) do
                overlap ← M(P_k, pos, angl, v, o)
                if overlap < minOverlap then
                    x_k = pos
                    o_k = θ
                    minOverlap ← overlap
                end if
            end for
        end for
    end if
end for
```

and, consequently, the space may be more thoroughly searched for the same execution time.

6. Results

The raster overlap minimization algorithm (ROMA) was developed to implement the proposed approach. Three parameters are used by the ROMA to solve an irregular strip packing problem. In order to execute the tests, compaction parameters r_{dec} and r_{inc} (shown in Algorithm 1) were defined as 0.04 and 0.01, respectively. For the separation strategy, the N_{mi} (shown in Algorithm 2) was set to 200 iterations. These values were also adopted by other algorithms (Elkeran, 2013; Imamichi et al., 2009).

Two groups of instances were tested using the ROMA: the dotted board instances and the adapted benchmark instances. The dotted board instances are the only available irregular strip packing instances with discrete placement in the literature and have known optimum in some cases. The classic benchmark instances were adapted with grid placement to better evaluate the performance of the ROMA. Unless stated otherwise, the grid for the rasterization and item placement was defined as a square grid with sides of size one. For each instance, independently of the group, the algorithm was executed 30 times.

The software was coded in C++ and generated using the Microsoft Visual Studio 12 compiler and was tested on a Core i9-7900X CPU, with 10 cores and 32GB of memory. The tests were performed on a single core, with the exception of the preprocessing stage. Some implementation optimizations were applied to achieve faster processing. One such improvement was that all raster penetration maps are read during the initialization and stored in the memory during execution. Furthermore, for each item, the last generated obstruction map is cached such that, in the next iteration, it may be reused.

The preprocessing stage was implemented as a separate module and its execution time is not accounted for in the tests. The results from this module can be saved once and then loaded for each run and, therefore, are only calculated once for each instance. As the raster penetration maps can be individually generated, the preprocessing is parallelized using the maximum number of cores available by the processor.

6.1. Dotted board instances

In Toledo et al. (2013), a mathematical solver for the irregular strip packing problem was proposed using the dotted board model. It was employed to optimally solve modified benchmark instances

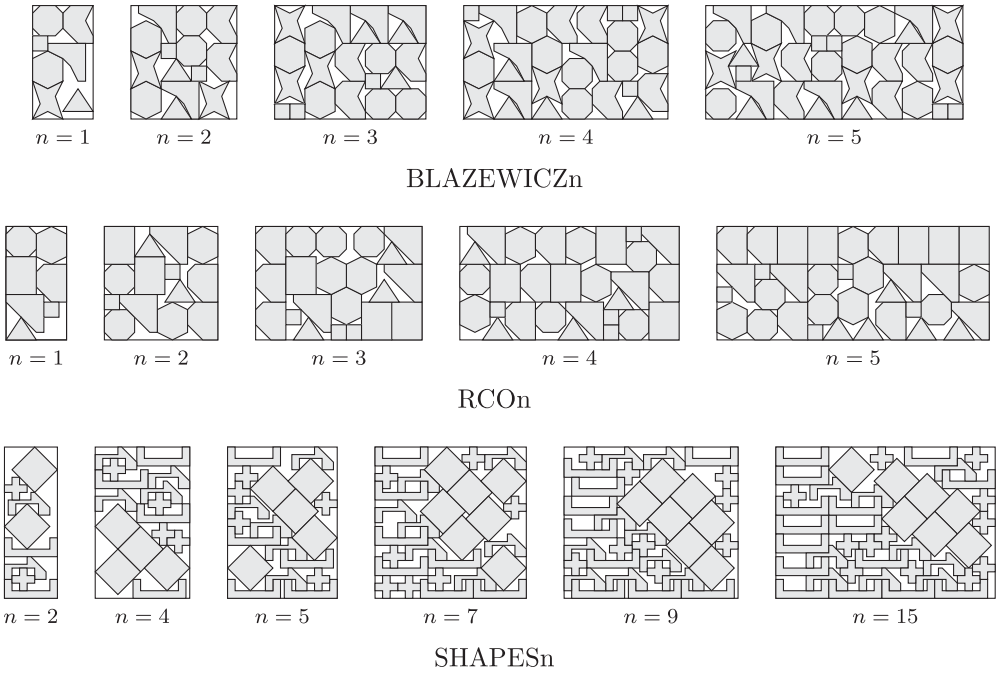

Fig. 10. Best ROMA solutions for the dotted board instances.

Table 1
Dotted board model instances data. **TNI**: total number of items. **NIT**: number of item types. **AO**: admissible orientations. **PPT**: preprocessing time.

Case	TNI	NIT	AO	PPT (s)
RCOn	7n	7	0°	0.04
BLAZEWICZn	7n	7	0°	0.06
SHAPES2	8	5	0°	0.03
SHAPES4	16	5	0°	0.03
SHAPES5	20	5	0°	0.03
SHAPES7	28	5	0°	0.03
SHAPES9	34	5	0°	0.03
SHAPES15	43	5	0°	0.03

of the irregular strip packing problem. Hence, this set of problems may provide a good measure of quality for the proposed approach. Table 1 shows some of the attributes of the problems. The time limit of ROMA was set to the maximum between the value reported by Toledo et al. (2013) and 5 minutes.

The results for the dotted board model cases are displayed in Table 2. The length average and variance were obtained by performing 30 executions of the algorithm. The last column shows the results reported in Toledo et al. (2013), which used a mathematical solver and ran on a workstation with two six-core Xeon 3.47 GHz processors. The marked results indicate global optimum values. It can be observed that, for all those cases, the ROMA obtained an optimal solution. For the other instances, the best compaction rate was equal to or higher for all the executions. The best solutions obtained by the ROMA are shown in Fig. 10.

6.2. Benchmark instances

The classic strip packing benchmarks offer a more varied and complex group of instances and were massively tested with other solutions in the literature. Therefore, these instances may be more adequate to evaluate the ROMA, as it was able to solve all the dotted board instances with the known solution. In order to adapt the instances to the ROMA, a square grid with a step size of one was applied to the benchmark instances.

The benchmark cases consist of 15 instances of varying levels of complexity, as shown in Table 3. Most cases admit more than one orientation, in multiples of 90°. The second to last column displays the total preprocessing time using 10 cores, which includes the generation of all nofit polygons and raster penetration maps. The preprocessing times were insignificant when compared to execution times for strip packing solutions, which usually range from 600 to 1200 seconds. The last column shows the memory requirements to store all the preprocessed raster penetration maps. It should be noted that such requirements may be reduced by exploring nofit polygon symmetries.

The parameters for the ROMA were set to the values used for the dotted board instance tests. For each instance, a time limit of 600 or 1200 seconds was imposed, subject to the complexity of the case. Table 4 displays the results, including the density average and variance for the 30 executions. It also contains results data for the GCS (*Guided Cuckoo Search*) (Elkeran, 2013) algorithm, which outperforms all the other solutions in the literature.

ROMA obtained the best compaction for the Marques instance, with an increase of 0.43%. Moreover, ROMA obtained densities equal to the best obtained by the GCS for 6 instances. As for the remaining cases, most are very competitive, with a difference inferior to 1.0%. The most underperforming cases involved the Albano, Mao, Swim and Shapes2 instances. The first three instances are more challenging for a dotted board approach, as the number of grid points is notably larger than the other instances. When the average densities are analyzed, ROMA improved upon the GCS results in 10 instances. Another noticeable behavior of the ROMA is the low variance for most cases, with the exception of the three most complex instances (Albano, Mao, Swim).

6.3. Multiresolution results

The adoption of a square grid with size one is usually an adequate choice for most benchmark instances, as indicated by the results from the benchmark tests. Nonetheless, a more refined grid allows for better exploring the solution space and may lead to improved results. The main drawback of improving the resolution of the dotted board is that the local search may become considerably

Table 2
Minimum lengths obtained for the dotted board model benchmark cases.

Case	ROMA i9 3.3 GHz (30 runs)				Toledo et al. (2013) Xeon 3.47 GHz	
	Average	Variance	Best	Time (s)	Best	Time (s)
RCO1	8.00	0.03	8	0.62	8[a]	0.62
RCO2	15.00	0.00	15	6.28	15[a]	6.28
RCO3	22.00	0.00	22	300.00	22[a]	2393.42
RCO4	29.00	0.00	29	300.00	29	18000.00
RCO5	36.33	0.24	36	300.00	37	18000.00
BLAZEWICZ1	8.03	0.00	8	0.69	8[a]	0.69
BLAZEWICZ2	14.00	0.00	14	15.98	14[a]	15.98
BLAZEWICZ3	20.23	0.20	20	300.00	20[a]	5583.82
BLAZEWICZ4	27.10	0.20	27	300.00	28	18000.00
BLAZEWICZ5	34.00	0.00	34	300.00	35	18000.00
SHAPES2	14.03	0.03	14	0.45	14[a]	0.45
SHAPES4	25.00	0.00	25	300.00	25[a]	17951.33
SHAPES5	29.00	0.00	29	300.00	30	18000.00
SHAPES7	40.00	0.03	40	300.00	45	18000.00
SHAPES9	46.43	0.24	46	300.00	54	18000.00
SHAPES15	58.60	0.26	58	300.00	67	18000.00

[a] optimal result.

Table 3
Benchmark instances data. **TNI**: total number of items. **NIT**: number of item types. **ANV**: average number of vertices. **AO**: admissible orientations. **PPT**: preprocessing time.

Case	TNI	NIT	ANV	AO	Preprocessing	
					Time (s)	Size (MB)
Albano	24	8	7.25	0°, 180°	11.05	8,110.54
Dagli	30	10	6.30	0°, 180°	0.06	0.71
Dighe1	16	16	3.87	0°	0.04	4.81
Dighe2	10	10	4.70	0°	0.03	2.37
Fu	12	12	3.58	0°, 90°, 180°, 270°	0.10	2.49
Jakobs1	25	25	5.60	0°, 90°, 180°, 270°	0.27	1.83
Jakobs2	25	25	5.36	0°, 90°, 180°, 270°	0.44	8.36
Mao	20	9	9.22	0°, 90°, 180°, 270°	7.20	5,643.72
Marques	24	8	7.37	0°, 90°, 180°, 270°	0.19	5.67
Shapes0	43	4	8.75	0°	0.02	0.02
Shapes1	43	4	8.75	0°, 180°	0.03	0.05
Shapes2	28	7	6.29	0°, 180°	0.05	0.04
Shirts	99	8	6.63	0°, 180°	0.05	0.14
Swim	48	10	21.90	0°, 180°	11.19	3,900.05
Trousers	64	17	5.06	0°, 180°	0.10	2.52

slow. As it directly impacts the obstruction map search, the multiresolution strategy is then a fitting approach for higher resolution versions of the benchmark instances.

In order to ensure that item vertices always coincide with grid points, g_o, the size of the refined instance grid $\mathcal{G}$, must be in the form of $1/n$, where n is an integer. For the tests performed with most benchmark cases, $1 \leq n \leq 10$. Due to memory restrictions, the Albano, Mao and Swim instances were only executed with the original grid. In these cases, the multiresolution was applied to improve the speed of the algorithm, which negatively influenced previous results. Therefore, for the multiresolution experiments, two test groups are defined: the accelerated instances (Albano, Mao and Swim) and the refined instances (the remaining cases).

There are two additional parameters needed for the multiresolution approach. The neighborhood for the refined search, controlled by parameter β, is set to the value of the original grid size g_o, allowing for fully exploring the search space. Parameter g_l is the grid size of the lower resolution grid $\mathcal{G}_l$ and dictates the size of the obstruction map. In the case of the refined instances, the unitary grid size $g_l = 1$ was adopted, as it achieved good results in the original ROMA tests. Nine g_l values were tested for the accelerated instances: 2, 4, 8, 16, 32, 64, 128, 256 and 512. The graph from Fig. 11 plots the average density obtained for 30 executions of each accelerated instance with varying g_l values and a time limit of 1200s. The best results among all the variants of ROMA and tested grid sizes are shown in Table 5 and in Table 6, which contain the best overall densities and the best average densities obtained, respectively. Fig. 12 displays the most compact layouts obtained by ROMA.

Table 4
Densities for the strip packing benchmark instances, given in percentage. The best densities in the literature are highlighted in bold. **Avg**: average. **Var**: variance.

Case	ROMA i9 3.3 GHz (30 runs)				GCS (Elkeran, 2013) i7 2.2 GHz (10 runs)		
	Best %	Avg %	Var %	Time (s)	Best %	Avg %	Time (s)
Albano	85.72	82.66	4.60	1200	**89.58**	**87.47**	1200
Dagli	88.73	**87.25**	0	1200	**89.51**	87.06	1200
Dighe1	**100.00**	**100.00**	0	600	**100.00**	**100.00**	600
Dighe2	**100.00**	**100.00**	0	600	**100.00**	**100.00**	600
Fu	91.94	**91.94**	0	600	**92.41**	90.68	600
Jakobs1	**89.09**	**89.09**	0	600	**89.09**	88.90	600
Jakobs2	**87.73**	**82.53**	3.99	600	**87.73**	81.14	600
Mao	83.61	81.08	1.31	1200	**85.44**	**82.93**	1200
Marques	**91.02**	**89.87**	0.05	1200	90.59	89.40	1200
Shapes0	**68.79**	**68.72**	0.09	1200	**68.79**	67.26	1200
Shapes1	**76.73**	**75.86**	0.52	1200	**76.73**	73.79	1200
Shapes2	80.00	80.00	0	1200	**84.84**	**82.40**	1200
Shirts	88.52	87.29	0.24	1200	**88.96**	**87.59**	1200
Swim	73.23	70.37	1.22	1200	**75.94**	**74.49**	1200
Trousers	90.75	**90.36**	0.03	1200	**91.00**	89.02	1200

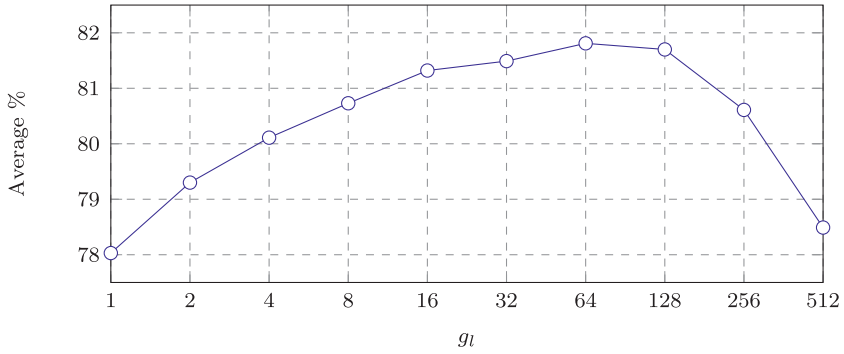

Fig. 11. Average densities of the original instances for different g_l values.

Table 5
Maximum densities for the strip packing benchmark instances obtained by the ROMA and other best solutions in the literature.

Instance	ROMA i9 3.3 GHz (30 runs)				GCS i7 2.2 GHz (10 runs)		FLD i5 2.6 GHz (10 runs)		ELS Pentium4 2.4 GHz (10 runs)	
	Best %	g_o	g_l	Time (s)	Best %	Time (s)	Best %	Time (s)	Best %	Time (s)
Albano	89.06	1	64	1200	**89.58**	1200	89.07	1200	88.48	1203
Dagli	88.73	1	1	1200	**89.51**	1200	88.20	1200	88.11	1205
Dighe1	**100.00**	1	1	600	**100.00**	600	**100.00**	1200	**100.00**	601
Dighe2	**100.00**	1	1	600	**100.00**	600	**100.00**	1200	**100.00**	600
Fu	92.31	1/8	1	600	**92.41**	600	92.11	1200	91.94	600
Jakobs1	**89.09**	1	1	600	**89.09**	600	**89.09**	1200	**89.09**	603
Jakobs2	**87.73**	1	1	600	**87.73**	600	85.25	1200	83.92	602
Mao	**86.05**	1	16	1200	85.44	1200	84.03	1200	84.33	1204
Marques	**91.02**	1	1	1200	90.59	1200	88.89	1200	89.73	1204
Shapes0	**68.79**	1	1	1200	**68.79**	1200	**68.79**	1200	67.63	1207
Shapes1	**76.73**	1	1	1200	**76.73**	1200	74.65	1200	75.29	1212
Shapes2	83.61	1/6	1	1200	**84.84**	1200		1200	84.23	1205
Shirts	88.52	1	1	1200	**88.96**	1200	88.96	1200	88.40	1293
Swim	75.66	1	128	1200	**75.94**	1200	75.48	1200	75.43	1246
Trousers	**91.06**	1/5	1	1200	91.00	1200	90.35	1200	89.63	1237

Table 6
Average densities for the strip packing benchmark instances obtained by ROMA and other best solutions in the literature.

Instance	ROMA i9 3.3 GHz (30 runs)				GCS i7 2.2 GHz (10 runs)		FLD i5 2.6 GHz (10 runs)		ELS Pentium4 2.4 GHz (10 runs)	
	Best %	g_o	g_l	Time (s)	Best %	Time (s)	Best %	Time (s)	Best %	Time (s)
Albano	87.55	1	128	1200	87.47	1200	**88.01**	1200	87.38	1203
Dagli	**87.50**	1/6	1	1200	87.06	1200	87.14	1200	86.27	1205
Dighe1	**100.00**	1	1	600	**100.00**	600	**100.00**	1200	91.61	601
Dighe2	**100.00**	1	1	600	**100.00**	600	**100.00**	1200	**100.00**	600
Fu	**91.95**	1/8	1	600	90.68	600	91.17	1200	90.00	600
Jakobs1	**89.09**	1	1	600	88.90	600	88.96	1200	88.35	603
Jakobs2	**83.56**	1/2	1	600	81.14	600	83.41	1200	80.97	602
Mao	**83.76**	1	64	1200	82.93	1200	82.28	1200	82.57	1204
Marques	**89.97**	1/5	1	1200	89.40	1200	88.38	1200	88.32	1204
Shapes0	**68.73**	1/2	1	1200	67.26	1200	67.39	1200	66.85	1207
Shapes1	**75.86**	1	1	1200	73.79	1200	73.91	1200	74.24	1212
Shapes2	**83.02**	1/6	1	1200	82.40	1200		1200	82.55	1205
Shirts	87.62	1/4	1	1200	87.59	1200	**88.21**	1200	87.20	1293
Swim	74.29	1	64	1200	74.49	1200	**74.66**	1200	74.10	1246
Trousers	**90.48**	1	1	1200	89.02	1200	89.17	1200	88.29	1237

The results for the multiresolution variant of the ROMA for the refined instances improved upon 3 best and 7 average compactions from the original algorithm. The accelerated instances results were even better, outperforming the single resolution ROMA every time.

Tables 5 and 6 also show a comparison with the five best solutions in the literature. These were selected by analyzing 14 approaches (Amaro Júnior et al., 2017; Amaro Júnior, Pinheiro, Saraiva, & Pinheiro, 2014; Bennell & Song, 2010; Burke et al., 2006; Egeblad et al., 2007; Elkeran, 2013; Hu, Fukatsu, Hashimoto, Imahori, & Yagiura, 2018; Imamichi et al., 2009; Leung et al., 2012; Mundim et al., 2017; Pinheiro et al., 2016; Sato et al., 2012; 2015; Wang, Xiao, & Wang, 2017). The puzzle cases Dighe1 and Dighe2 are solved to optimality by several solutions (Amaro Júnior et al., 2017; Amaro Júnior et al., 2014; Bennell & Song, 2010; Elkeran, 2013; Leung et al., 2012; Mundim et al., 2017; Pinheiro et al., 2016; Sato et al., 2012; Wang et al., 2017) and the same density is repeatedly achieved for the Jakobs1 instances (Amaro Júnior et al., 2017; Elkeran, 2013; Leung et al., 2012; Mundim et al., 2017; Wang et al., 2017). For all the other instances, the best layouts were obtained by GCS. Other important results derived from the flexible labour division (FLD) approach from Wang et al. (2017), which achieved the maximum density equal to the GCS for 4 instances, and by the extended local search (ELS) algorithm (Leung et al., 2012), with 4 best tied results. Therefore, these three strategies – GCS, FLD and ELS – are contemplated in the comparison study.

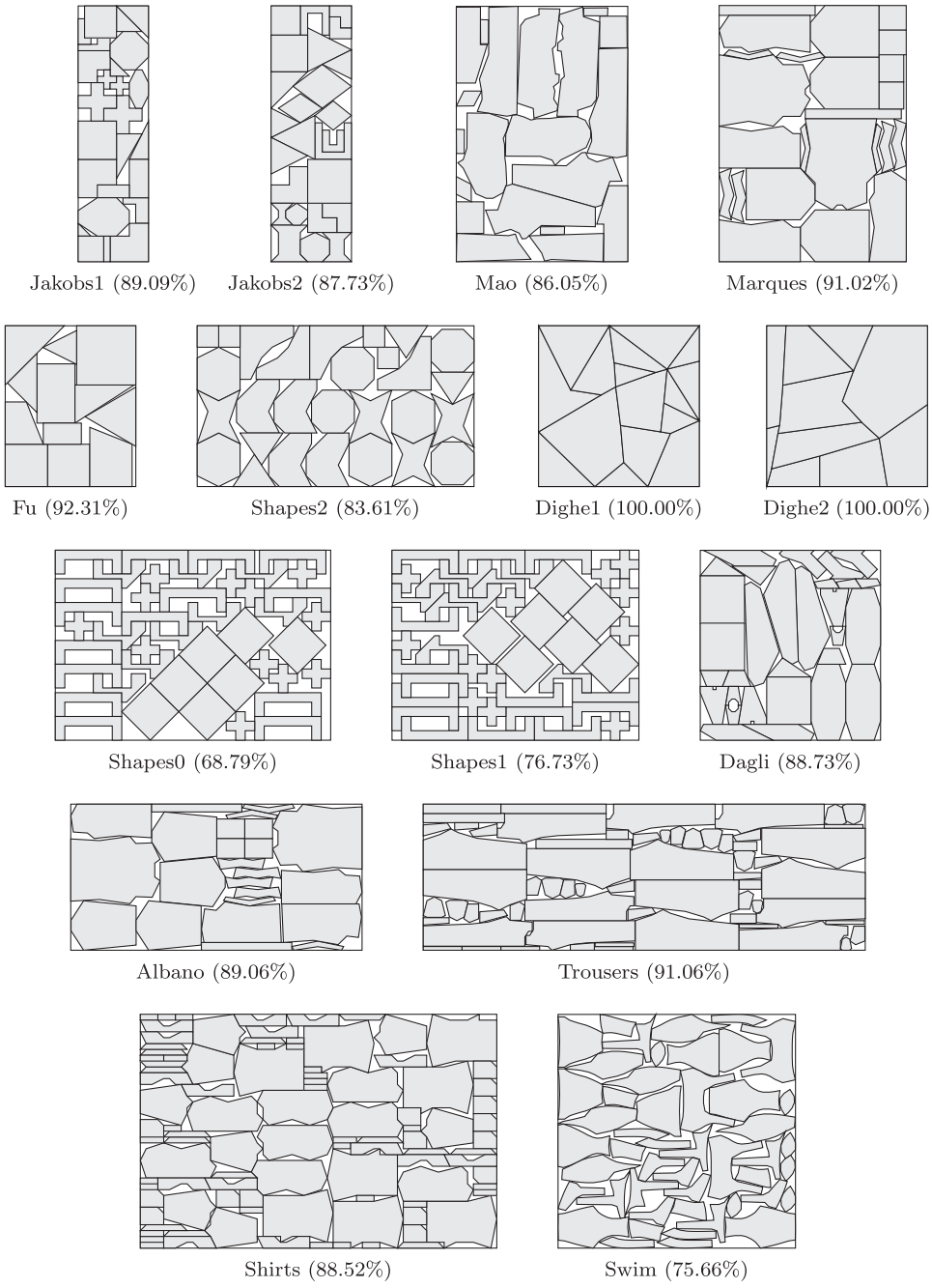

Fig. 12. ROMA best solutions for the benchmark instances.

When contrasted with the best solutions in the literature, the multiresolution ROMA showed to be very competitive. It achieved 9 out of 15 better or equal compactions for the benchmark instances among the best known solutions. One special mention among the non-improved cases is the Fu instance, whose most compact layout is visually similar to the solution obtained by Elkeran (2013). The small gap in density is due to the grid placement limitation, which forces the placement of some items slightly to the right of the layout. Moreover, it obtained the best average densities for 13 instances, which attests to its consistency.

6.4. Computational performance discussion

The ROMA obtained very good results when compared with other solutions in the literature. However, it is difficult to assess the quality of the results, as other solutions were executed in machines with different computational capabilities. In particular, the CPU used for the ROMA testing is vastly superior to the others adopted in the literature. Therefore, in order to enhance the comparison analysis, a computational performance study of the ROMA was conducted.

During the execution of the ROMA tests, the utilization of the solutions was determined at each 60-second interval and the average density of all the 30 executions was determined. This procedure is equivalent to adopting smaller time limits. Adopting the parameters shown in Table 6, the differences between the average densities obtained by sampling the ROMA and the best in the literature were calculated, and are shown in Figs. 13 and in14. From these graphs, it is possible to affirm that the 12 best results could be obtained by adopting a time limit of 480 seconds. The only

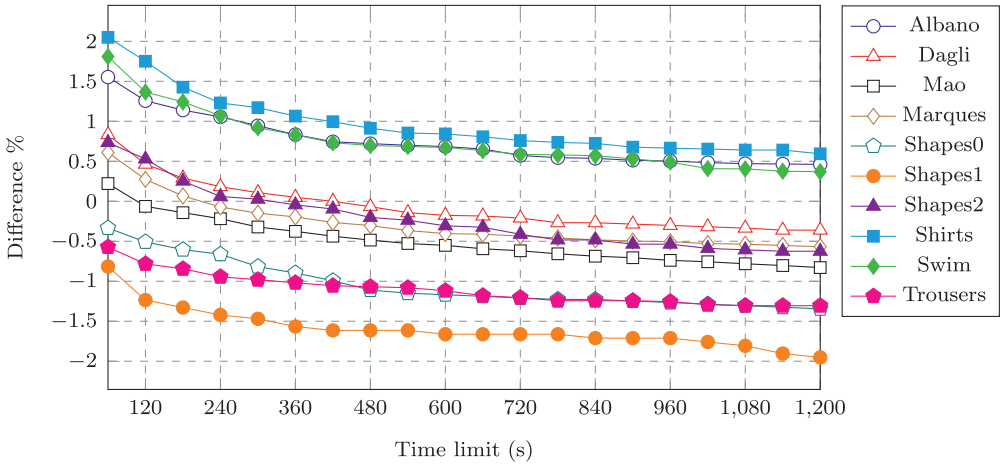

Fig. 13. Difference between the best compaction in the literature and the ROMA utilization for different time limits.

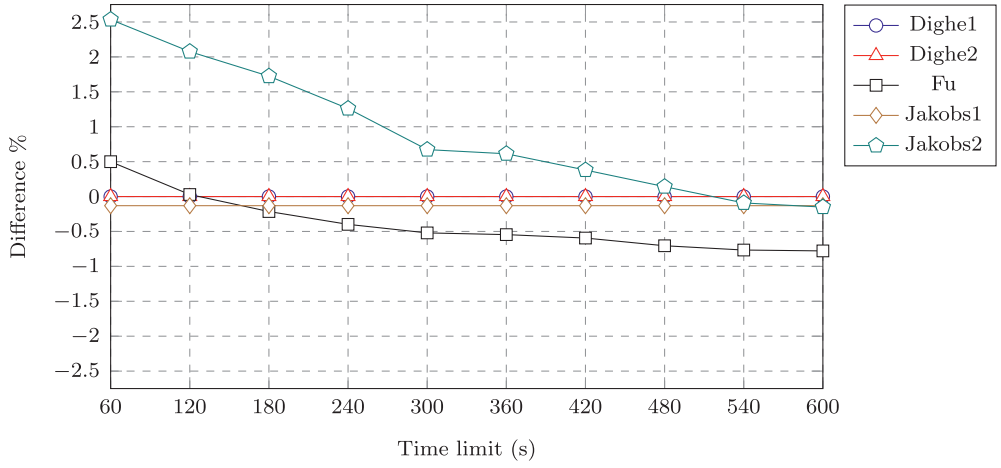

Fig. 14. Difference between the best compaction in the literature and the ROMA utilization for different time limits.

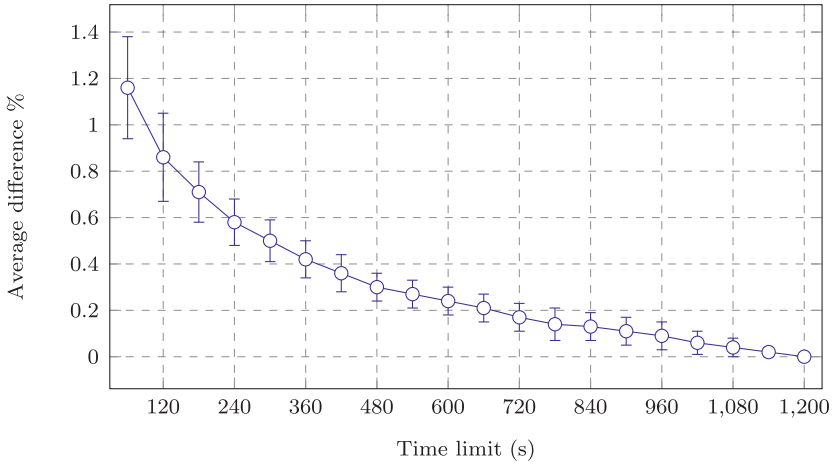

Fig. 15. Average relative gap (density distance to the best solution) for execution with different time limits. Error bars represent the standard deviation.

instance which required more time to surpass the best average compaction is Jakobs2. Moreover, a convergence study was conducted for the cases with a 1200-second time limit, which shows similar tendencies in Fig. 13. It resulted in the graph from Fig. 15, which displays the average density difference of all the 10 instances to the final compaction for different time limits. It can be noted that the difference and the deviation decreases more intensely in the first half of the execution. This is further evidence that a reduced time limit does not greatly impact the results for these cases.

6.5. Terashima instances

Terashima-Marín, Ross, Farías-Zárate, López-Camacho, and Valenzuela-Rendón (2010) artificially generated a large set of irregular instances for the 2D bin packing problem. A total of 540

Table 7
Terashima instances data and results. **TNI**: total number of items.

Type	Width	Optimal Length	TNI	Avg Best %	Convergence rate
Type A	1000	3000	30	94.29	0.00
Type B	1000	10000	30	99.92	76.67
Type C	1000	6000	36	94.72	3.33
Type D	1000	3000	60	86.45	0.00
Type E	1000	3000	60	86.31	0.00
Type F	1000	2000	30	95.13	10.00
Type G	1000	unknown	36	94.35	-
Type H	1000	12000	36	99.24	53.33
Type I	1000	3000	60	93.33	0.00
Type J	1000	4000	60	90.36	0.00
Type K	1000	6000	54	95.21	3.33
Type L	1000	3000	30	99.31	40.00
Type M	1000	5000	40	96.47	6.67
Type N	1000	2000	60	87.78	0.00
Type O	1000	7000	28	99.94	80.00
Type P	1000	8000	56	91.40	0.00
Type Q	1000	15000	60	94.55	0.00
Type R	1000	9000	54	92.98	0.00

instances were created using a “broken glass” approach, in which the items are generated by sequentially dividing the container using straight cuts. Thus, the known optimal solution for each case is known. The generator was able to control some aspects, such as the size variability and the irregularity of items and, through these parameters, 18 groups of 30 instances were defined as types A-R. No rotation is allowed and there is only a single item of each type. The instances were adapted to the strip packing problem by ignoring the container data, replacing it with a bin of the same width and a variable length. Table 7 shows the characteristics of the modified instances.

Due to the large containers, the Terashima instances have substantial memory and processing requirements for the ROMA, akin to the accelerated instances from the benchmark cases. Therefore, the multiresolution approach was executed with the original grid and the g_l parameter was determined through an extensive investigation: one instance of each type was executed for 10 minutes with different resolutions of the finer grid ($g_l = 2^1, 2^2, \ldots, 2^9$). The result of these evaluations was that the value of 32 was chosen for this parameter, as it yielded the best average compaction.

For the Terashima instances experiments, ten executions of each instance were performed with a time limit of 1200 seconds. For each instance, the best result was recorded and used to determine the average type density. These results are compiled in Table 7, where the last column represents the percentage of the instances in which the optimal solution was achieved. Although the convergence rate was low for most of the types, the average density was considerably high – more than 90 % in 15 cases. Note that the strip packing version is more difficult to solve than the original cases. Moreover, the low irregularity combined with the high resolution of the instances are disadvantageous characteristics for the ROMA. Nevertheless, the results for the Terashima instances further expand on the investigation of the ROMA and also serve as basis for further comparisons.

7. Conclusion

The raster overlap minimization algorithm (ROMA) was developed to solve the irregular strip problem using a raster method that limits the solution space. A placement grid was adopted to limit the positioning of items and an overlap minimization approach was adopted to simplify the multiple objective problem. Using the obstruction map, a complete investigation of the placement space was performed for each item, which is generally not possible when using a continuous space for placements. The proposed raster overlap map transfers much of the computational load of overlap determination to a preprocessing stage. The preprocessing step was parallelized and, for every instance, the execution time was minimal when compared to the time limit established for the ROMA algorithm.

Two sets of instances were tested using the proposed ROMA. The comparison performed with a mathematical approach solution using the dotted board instances showed that ROMA was always able to find the optimal solution. Tests using benchmark cases for the irregular strip packing problem yielded competitive solutions, producing solutions equally or more compact layouts for 9 out of 15 instances, including 3 best solutions in the literature.

Although different grid sizes were tested for each case, most of the best solutions were obtained using a grid step equal to one, with few exceptions. The multiresolution search, proposed to accelerate the algorithm, was notably useful in three instances, in which the original grid had a large number of points. Nevertheless, to obtain good performance with such instances, the lower resolution parameter had to be manually calibrated. The results, however, indicate that it might be possible to automatically detect its value, simplifying the parameter definition in such cases. Overall, good compaction for irregular packing was obtained by the ROMA even with a very constrained placement space. These results show the potential of applying similar approaches to other irregular packing problems.

Acknowledgments

This research was supported by FAPESP (Grants 2008/11132-7, 2010/18913-4 and 2013/26532-9) and by CNPq (Grant 456.180/2014-1). André Kubagawa Sato is supported by FAPESP (Grant 2010/18658-4) and CAPES/PNPD. Thiago Castro Martins is partially supported by CNPq (Grant 306.415/2012-7). Marcos Sales Guerra Tsuzuki is partially supported by CNPq (Grants 310.663/2013-0 and 305.959/2016–6).

References

Alvarez-Valdes, R., Martinez, A., & Tamarit, J. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics, 145*(2), 463–477.
Amaro Júnior, B., Pinheiro, P. R., & Coelho, P. V. (2017). A parallel biased random-key genetic algorithm with multiple populations applied to irregular strip packing problems. In *Mathematical problems in engineering* (p. 11). 2017
Amaro Júnior, B., Pinheiro, P. R., Saraiva, R. D., & Pinheiro, P. G. C. D. (2014). Dealing with nonregular shapes packing. In *Mathematical problems in engineering* (p. 10). 2014
Art, R. C. (1966). *An approach to the two-dimensional irregular cutting stock problem*. IBM Cambridge Centre. Technical Report 36.008
Bennell, J., & A. Dowsland, K. (2010). Tabu thresholding implementation for the irregular stock cutting problem. *International Journal of Production Research, 37*, 4259–4275. 11
Bennell, J. A., & Dowsland, K. A. (2001). Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science, 47*(8), 1160–1172.
Bennell, J. A., & Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research, 184*(2), 397–415.
Bennell, J. A., & Oliveira, J. F. (2009). A tutorial in irregular shape packing problems. *The Journal of the Operational Research Society, 60*, s93–s105.
Bennell, J. A., & Song, X. (2010). A beam search implementation for the irregular shape packing problem. *Journal of Heuristics, 16*, 167–188.
Burke, E., Hellier, R., Kendall, G., & Whitwell, G. (2006). A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research, 54*(3), 587–601.
Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M., Oliveira, J. F., & Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research, 253*(3), 570–583.
Egeblad, J., Nielsen, B. K., & Odgaard, A. (2007). Fast neighborhood search for two- and three–dimensional nesting problems. *European Journal of Operational Research, 183*, 1249–1266.
Elkeran, A. (2013). A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research, 231*(3), 757–769.
Faroe, O., Pisinger, D., & Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing, 15*(3), 267–283.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). *Distance transforms of sampled functions*. Cornell computing and information science. Technical Report

Fowler, R. J., Paterson, M., & Tanimoto, S. L. (1981). Optimal packing and covering in the plane are NP-complete. *Information Processing Letters, 12*(3), 133–137.

Gomes, A. M., & Oliveira, J. F. (2006). Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research, 171*, 811–829.

Hu, Y., Fukatsu, S., Hashimoto, H., Imahori, S., & Yagiura, M. (2018). Efficient overlap detection and construction algorithms for the bitmap shape packing problem. *Journal of the Operations Research Society of Japan, 61*(1), 132–150.

Imamichi, T., Yagiura, M., & Nagamochi, H. (2009). An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization, 6*, 345–361.

Jones, D. R. (2014). A fully general, exact algorithm for nesting irregular shapes. *Journal of Global Optimization, 59*(2), 367–404.

Leung, S. C., Lin, Y., & Zhang, D. (2012). Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research, 39*(3), 678–686.

Li, Z., & Milenkovic, V. (1995). Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research, 84*(3), 539–561.

Mundim, L. R., Andretta, M., & de Queiroz, T. A. (2017). A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. *Expert Systems with Applications, 81*, 358–371.

Oliveira, J. F., Gomes, A. M., & Ferreira, J. S. (2000). TOPOS – a new constructive algorithm for nesting problems. *OR Spectrum, 22*, 263–284.

Pinheiro, P. R., Amaro Júnior, B., & Saraiva, R. D. (2016). A random-key genetic algorithm for solving the nesting problem. *International Journal of Computer Integrated Manufacturing, 29*(11), 1159–1165.

Rodrigues, M. O., Cherri, L. H., & Mundim, L. R. (2017). MIP models for the irregular strip packing problem: new symmetry breaking constraints. In *Proceedings of the ITM Web conference: 14* (p. 00005).

Sato, A. K., Martins, T. C., & Tsuzuki, M. S. G. (2012). An algorithm for the strip packing problem using collision free region and exact fitting placement. *CAD, 44*, 766–777.

Sato, A. K., Martins, T. C., & Tsuzuki, M. S. G. (2015). A pairwise exact placement algorithm for the irregular nesting problem. *International Journal of Computer Integrated Manufacturing, 29*, 1177–1189.

Terashima-Marín, H., Ross, P., Farías-Zárate, C. J., López-Camacho, E., & Valenzuela-Rendón, M. (2010). Generalized hyper-heuristics for solving 2D regular and irregular packing problems. *Annals of Operations Research, 179*, 369–392.

Toledo, F. M., Carravilla, M. A., Ribeiro, C., Oliveira, J. F., & Gomes, A. M. (2013). The dotted-board model: A new MIP model for nesting irregular shapes. *International Journal of Production Economics, 145*(2), 478–487.

Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K., & Ibaraki, T. (2009). Solving the irregular strip packing problem via guided local search for overlap minimization. *International Transactions in Operational Research, 16*(6), 661–683.

Wang, A., Hanselman, C. L., & Gounaris, C. E. (2018). A customized branch-and-bound approach for irregular shape nesting. *Journal of Global Optimization, 71*(4), 935–955.

Wang, Y., Xiao, R., & Wang, H. (2017). A flexible labour division approach to the polygon packing problem based on space allocation. *International Journal of Production Research, 55*(11), 3025–3045.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research, 183*, 1109–1130.

Xu, J., Wu, X., Liu, H., & Zhang, M. (2017). An optimization algorithm based on no-fit polygon method and hybrid heuristic strategy for irregular nesting problem. In *Proceedings of the 36th Chinese control conference (CCC)* (pp. 2858–2863).