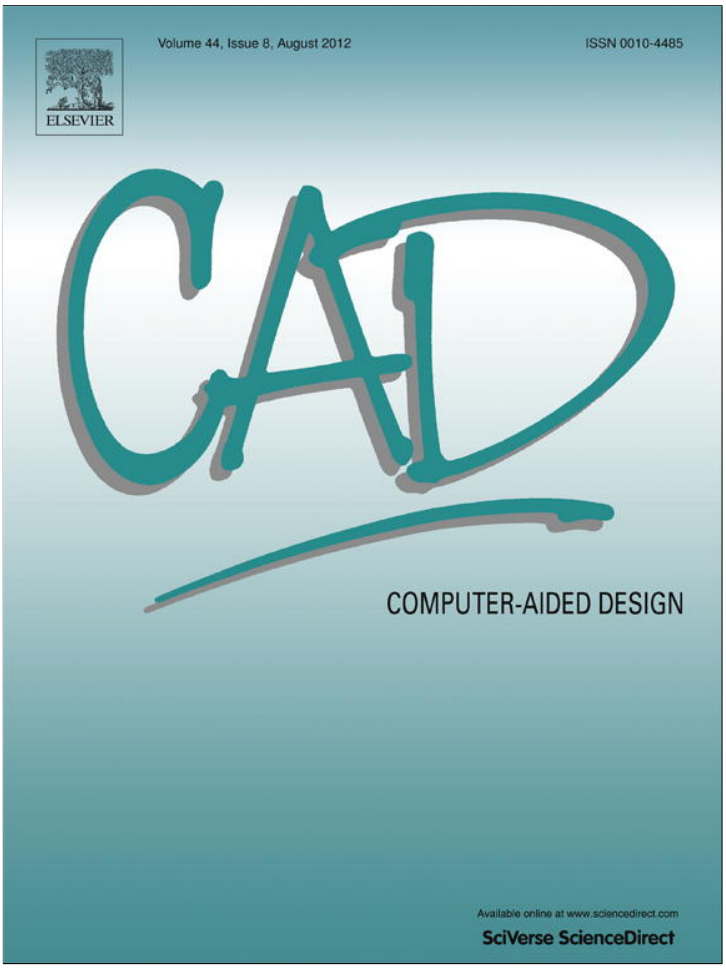

Computer-Aided Design 44 (2012) 766–777

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

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

An algorithm for the strip packing problem using collision free region and exact fitting placement

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

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

ARTICLE INFO

Article history:
Received 13 September 2011
Accepted 3 March 2012

Keywords:
Packing
Simulated annealing
Combinatorial optimization

ABSTRACT

The irregular shape packing problem is approached. The container has a fixed width and an open dimension to be minimized. The proposed algorithm constructively creates the solution using an ordered list of items and a placement heuristic. Simulated annealing is the adopted metaheuristic to solve the optimization problem. A two-level algorithm is used to minimize the open dimension of the container. To ensure feasible layouts, the concept of collision free region is used. A collision free region represents all possible translations for an item to be placed and may be degenerated. For a moving item, the proposed placement heuristic detects the presence of exact fits (when the item is fully constrained by its surroundings) and exact slides (when the item position is constrained in all but one direction). The relevance of these positions is analyzed and a new placement heuristic is proposed. Computational comparisons on benchmark problems show that the proposed algorithm generated highly competitive solutions. Moreover, our algorithm updated some best known results.

1. Introduction

Cutting and packing problems are classical problems of finding the most efficient layout given an input set of items and a container with the objective of minimizing waste. These problems have significant economical and ecological impact. Cutting and packing problems arise in several industries such as garment, wood, ship and glass.

According to the typology proposed by Wäscher et al. [1], the problem considered here is the two-dimensional irregular open dimension problem (ODP). The problem can be stated as the minimization of the length of a rectangular container with fixed width. Each irregular item is represented by a polygon and can be rotated by a finite set of angles. An example of its application is found in the garment factory. The items, in this case, referred to as markers or stencils, are irregular and need to be arranged in a very long sheet of fabric with a fixed width. In order to minimize fabric waste, items must be arranged in such a way that they occupy a rectangular container with minimal length, which can be considered a variable. Additionally, in the garment problem, there is a limitation with regard to the rotations of items. Items are not allowed to rotate freely, as the weave of the cloth and drawing patterns should be considered.

In cutting and packing problems involving irregular items, the task of obtaining layouts in which the geometric conditions hold, i.e. all the items must lie entirely inside the container and not overlap, is a very complex one. Bennell and Oliveira [2] investigated existing approaches in the literature.

Fowler et al. [3] demonstrated that the considered problem is NP-complete. As a consequence, most proposed solutions in the literature adopt heuristics, either deterministic or probabilistic. In this work, a probabilistic heuristic, simulated annealing is adopted. Simulated annealing was proposed by Kirkpatrick et al. [4] in the field of combinatorial optimization and is largely used to solve combinatorial problems of several areas.

Numerous solution methods have been proposed for the irregular packing problem. Dowsland and Dowsland [5] reviewed those methods; however, a more recent survey was performed by Hopper and Turton [6].

Gomes and Oliveira [7] proposed a 2-exchange mechanism to search over a placement sequence. A bottom–left procedure was adopted as the placement heuristic and only feasible placements were considered. Gomes and Oliveira [8] considered the relaxed placement problem, in which pieces may overlap, and used a separation algorithm to obtain feasible layouts. Length minimization was obtained by hybridizing simulated annealing to guide the search over the solution space and a compaction algorithm based on linear programming. An initial solution was obtained using a constructive heuristic TOPOS [9]. This heuristic was later improved by Bennell and Song [10], and combined with a beam search to deterministically obtain the final solution.

* Corresponding author at: Computational Geometry Laboratory, Department of Mechatronics and Mechanical Systems Engineering, Escola Politécnica da Universidade de São Paulo, Brazil. Tel.: +55 11 3091 5759.
E-mail address: mtsuzuki@usp.br (M.S.G. Tsuzuki).

0010-4485/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2012.03.004

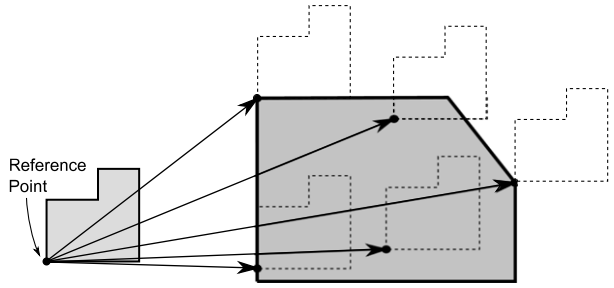

Fig. 1. Translations applied to a polygon (left) represented by a region in space (right).

Burke et al. [11] proposed a new bottom–left heuristic and implemented a new shape representation that incorporated circular arcs and holes. Egeblad et al. [12] formulated a polynomial time algorithm to determine the amount of overlap. Considering only vertical and horizontal translation, they proposed a fast neighborhood search in order to obtain minimum overlap and, consequently, feasible layouts. Imamichi et al. [13] combined a swap procedure with iterative local search and a separation algorithm based on nonlinear programming to solve the overlapping minimization problem. Leung et al. [14] adopted a similar approach, but used a tabu search to escape local optima.

Some recent works adopted the collision free region in order to determine feasible layouts in containers with fixed dimensions [15]. Sato et al. [16] developed a robust algorithm to determine the collision free region and improved previously published results [17,18].

This paper is structured as follows. Section 2 describes the concepts needed for understanding the proposed solution: the no-fit polygon and the inner-fit polygon. Section 3 describes the collision free region and its external edges and vertices. Section 4 presents the proposed two-level algorithm in which the inner level is a simulated annealing algorithm. In this section, the proposed placement heuristic that detects, the presence of exact fits (when the item is fully constrained by its surroundings) and exact slides (when the item position is constrained in all but one direction) for a moving item, is explained. Finally, computational results are presented and conclusions are drawn.

2. Supporting concepts

Before describing the collision free region, some concepts are introduced, namely the no-fit polygon and the analogous concept of inner-fit polygon. These concepts were proposed to deal with packing problems involving irregular items.

2.1. No-fit polygon

For two-dimensional irregular packing problems, obtaining a feasible layout, i.e. in which no items collide or protrude from the container, is a complex task. There is the need to employ a geometric tool to determine whether two items collide, touch or are separated. The raster method, direct geometry and no-fit polygons are some of the tools proposed in the literature. The no-fit polygon, which was first introduced by Art [19], is chosen as it is more precise than the raster method and has a lower computational cost as compared to the direct geometry method [2].

The no-fit polygon represents the set of forbidden translations that, when applied to the movable item, causes it to overlap the fixed item. It is said that the no-fit polygon is induced by the fixed item to the movable item. The translations are mathematically represented by a set of vectors and can be graphically represented by a region in space defined by a reference point (see Fig. 1). For an

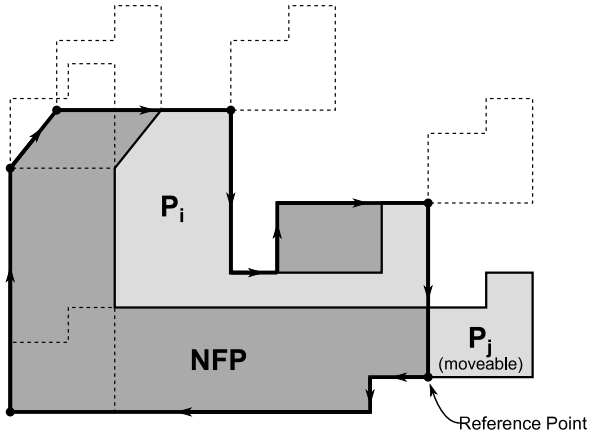

Fig. 2. No-fit polygon induced by item P_i to item P_j determined using a sliding scheme.

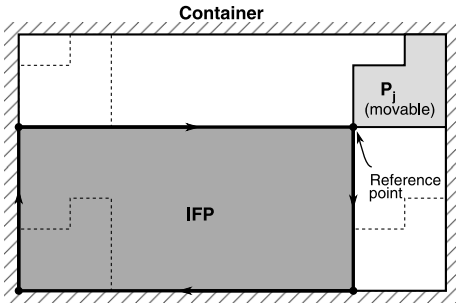

Fig. 3. Inner-fit polygon for a container and a movable item P_j.

item P, let $i(P)$ be its interior, $\partial(P)$ be its boundary and $c(P)$ be its complement.

Definition 2.1. Consider two fixed polygons P_i and P_j. The no-fit polygon induced by item P_i to item P_j, denoted by $\mathrm{NFP}(P_i, P_j)$, is the set of translation vectors applied to P_j that leads it to a collision with P_i. Thus,

$$\mathrm{NFP}(P_i, P_j) = \left\{\vec{\mathbf{v}} \mid \exists\, \mathbf{a} \in i(P_j), \mathbf{a} + \vec{\mathbf{v}} \in i(P_i)\right\}.$$

The no-fit polygon can be obtained through a sliding scheme [20]. This is accomplished by sliding the movable item along the contour of the fixed item. The no-fit polygon is defined by the trace of the reference point. Fig. 2 shows an example of a no-fit polygon obtained by using this sliding scheme.

Minkowski sums are used here to determine the no-fit polygon. When only convex polygons are involved, the no-fit polygon can be computed in linear time [21]. In order to compute the Minkowski sum of non-convex items, a convex decomposition is performed in a preprocessing stage.

2.2. Inner-fit polygon

The inner-fit polygon is a concept derived from the no-fit polygon and it represents the set of translations that places an item entirely inside a container. The inner-fit polygon can be obtained by sliding the item along the internal contour of the container (see Fig. 3).

Definition 2.2. The inner-fit polygon induced by container $\mathcal{C}$ to item P_j, denoted by $\mathrm{IFP}(\mathcal{C}, P_j)$, is the set of translation vectors applied to P_j that leads it to be inside the container. Thus,

$$\mathrm{IFP}(\mathcal{C}, P_i) = \left\{\vec{\mathbf{v}} \mid \forall\, \mathbf{a} \in i(P_i), \mathbf{a} + \vec{\mathbf{v}} \in \mathcal{C}\right\}.$$

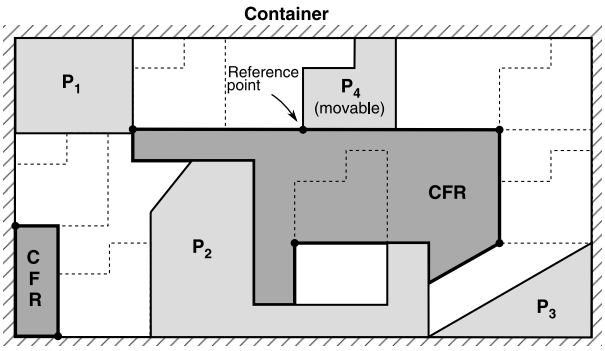

Fig. 4. Collision free region (CFR) for movable item P_4 given a subset of placed items $\{P_1, P_2, P_3\}$.

In the studied problem, only rectangular containers are considered. In this particular case, the inner-fit polygon of any given item is rectangular and, moreover, is only dependent on the bounding box and reference point of the item.

3. Collision free region

The collision free region, that was originally introduced by Martins and Tsuzuki [15], is the main tool employed in this work to achieve feasible layouts. For each item, it represents all the possible positions for its placement. This allows the construction of any given solution by placing items sequentially without overlap.

Consider a feasible layout with a container $\mathcal{C}$ and a set of placed items $\mathcal{P} = \{P_1, \ldots, P_n\}$. A new item P_m, $m = n+1$, will be inserted into the container, keeping the layout feasible. The collision free region represents a set of translations for item P_m (see Fig. 4).

Definition 3.1. The collision free region is the set of all the translations, that, when applied to a specific item, places the specific item inside a container without colliding with the already placed items.

When there are no placed items, the collision free region is the inner-fit polygon. The first step to determine the collision free region is to obtain the corresponding inner-fit polygon. The next step is to subtract the no-fit polygons induced by the placed items. The collision free region, denoted $\mathrm{CFR}(\mathcal{C}, \mathcal{P}, P_m)$, can be determined by using

$$\mathrm{CFR}(\mathcal{C}, \mathcal{P}, P_m) = \mathrm{IFP}(\mathcal{C}, P_m) - \bigcup_{P_i \in \mathcal{P}} \mathrm{NPF}(P_i, P_m). \quad (1)$$

Gomes and Oliveira [7] mathematically defined the set A of admissible points for the placement of item P_k as:

$$A = \{(x, y) : (x, y) \in i(\mathrm{IFP}(\mathcal{C}, P_k)) \wedge (x, y) \notin i(\mathrm{NFP}(P_i, P_k)), \; P_i \in \mathcal{P}\}. \quad (2)$$

Set A represents the collision free region for the movable item P_k, given container $\mathcal{C}$ and a subset of already placed items $\mathcal{P} = \{P_1, \ldots, P_n\}$. Gomes and Oliveira [7] only considered the bottom–left vertex of the collision free region. As explained in the following section, it is possible to determine vertices and edges from the collision free region that respectively represent exact fits and exact slides.

3.1. Degenerated edges and vertexes

A collision free region may be composed of polygons, degenerated edges and degenerated vertices. These degenerated elements, edges and vertices, are always external to the polygons in the collision free region, as they represent additional allowed positions for the item placement.

When placing an item in a degenerated edge, that represents an exact slide, one can observe that the item will only be able to move in one direction in the layout (see Fig. 5). If a placement occurs in a degenerated vertex, that represents an exact fit, no movement is possible (see Fig. 6).

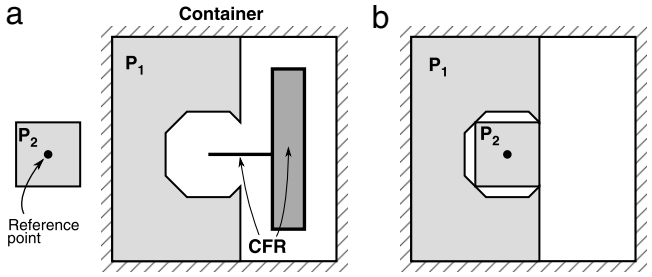

Fig. 5. Placement of an item on a degenerated edge, that represents an exact slide. (a) Collision free region (CFR) for item P_2. (b) Placement in a degenerated edge. It is possible to observe that the item can only move horizontally across the degenerated edge.

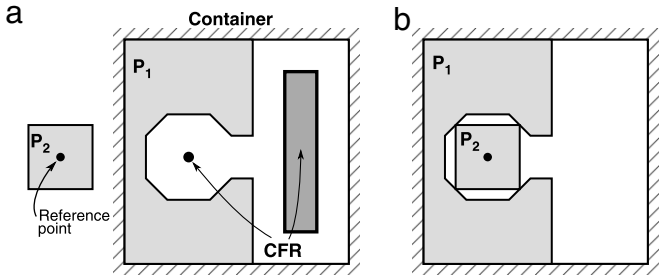

Fig. 6. Placement of an item at a degenerated vertex, that represents an exact fit. (a) Collision free region (CFR) for item P_2. (b) Placement in a degenerated vertex. It is possible to observe that the item is fixed in the degenerated vertex.

Degenerate elements in the collision free region have three possible sources. Based on their source, a possible classification of degenerated edges and vertices is:

- No-fit polygon generated: No-fit polygons involving non-convex items might contain degenerated edges and vertices. They represent positions where the two items lock together. For no-fit polygons, degenerated edges and vertices are always internal to polygons (see Fig. 7(a)).
- Obstructed region generated: Obstructed region is the union of two or more no-fit polygons and can be considered an intermediate step to determine the collision free region using Eq. (1). It is possible to find degenerated elements that are exclusive to the obstructed region, different from the degenerated elements in the individual no-fit polygon. They also represent positions where items lock together, but it is dependent on the placement of the items (see Fig. 7(b)).
- Container generated: Positions where the items fit exactly inside the container (see Fig. 7(c)).

Detection of degenerated edges and vertices was made possible by employing non-manifold Boolean operations to determine the collision free region. The Boolean operations (union and difference) were implemented using fixed precision with 128 bits and edge intersections were determined using the sweep algorithm proposed by Hobby [22]. Degenerated edges represent exact slides and degenerated vertices represent exact fits.

4. Proposed approach

A new algorithm is proposed in this work so as to solve the two-dimensional irregular open dimension problem. Each item can be rotated by a finite set of angles. Simulated annealing is the adopted probabilistic metaheuristic in this approach and it is used

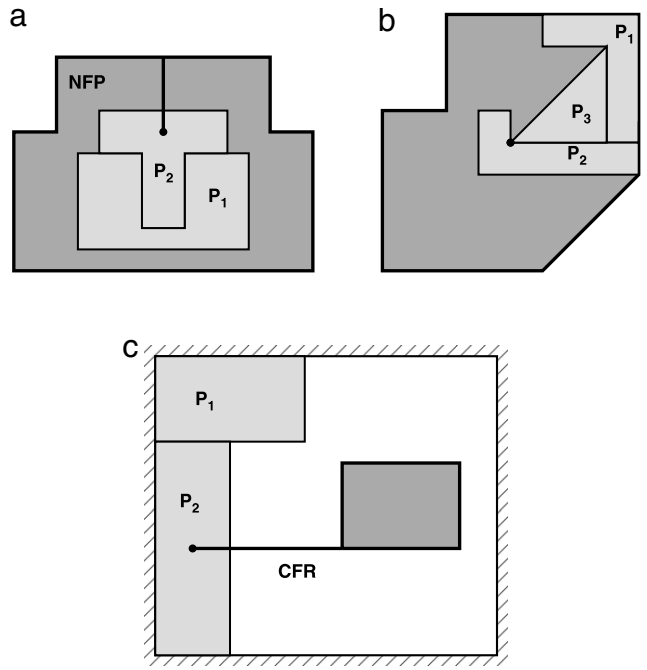

Fig. 7. Degenerated elements examples. (a) No-fit polygon generated, P_1 is fixed and P_2 is movable. (b) Obstructed region generated, P_1 and P_2 are fixed and P_3 is movable. (c) Container generated, P_1 is fixed and P_2 is movable.

to solve an intermediate problem in which the container has fixed dimensions. The objective is to place all items inside the container without overlap. An outer level is then responsible for controlling the container length.

As the placement of items is performed sequentially, the collision free region dictates the placement procedure. In order to restrict and guide the search in the solution space, a placement heuristic is adopted. It takes advantage of the particular properties of degenerated edge and vertex placement.

4.1. Simulated annealing

Simulated annealing was proposed by Kirkpatrick et al. [4] for the combinatorial optimization. It is based on the Metropolis algorithm [23], which is capable of simulating the atoms configuration in equilibrium at a given temperature. Kirkpatrick et al. [4] adapted the Metropolis algorithm to reflect the behavior of atoms during an annealing process. During the process, atoms naturally migrate to a minimum energy configuration, even if a higher energy configuration occurs during the process. The idea is that the annealing process can be considered analogous to a process of solving a multivariate optimization problem, in which the objective is to minimize the value of a function.

During the simulated annealing execution, random modifications are applied to the solution and the objective function is recalculated. The new solution is immediately accepted if the value of its function is lower than the previous solution. Otherwise, the solution can be accepted given the following probability:

$$P(\Delta E) = e^{-\frac{\Delta E}{kt}}, \tag{3}$$

ΔE is the difference between the current function value and the previous value, $P(\Delta E)$ is the probability of accepting a solution with a higher objective function value than its predecessor, k is a parameter analogous to the Stefan–Boltzmann parameter and t is the temperature.

A simulated annealing solution x is a set of placements for all items, in which the placement sequence, orientation and position

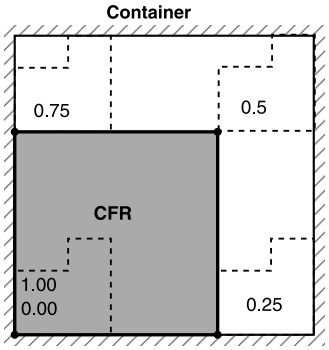

Fig. 8. Example of items placement in which some values of the position parameter and the associated placement position are shown.

of each item are represented. The initial solution is randomly chosen, placement sequence is a set of integer numbers that identifies the items, orientation is an integer number from the set of allowable orientations and position is a pair of real numbers from the interval [0, 1].

The first position number represents a placement along the collision free region boundary. Fig. 8 shows the relationship between the position parameter and the actual layout placement of an item. The origin corresponds to the bottom–left placement. When the collision free region is represented by more than one contour, degenerated edge or degenerated vertex, it is necessary to use the second parameter to determine the item position. This parameter determines at which contour, or degenerated element, the placement occurs, each with equal probability of being chosen. If an item has an associated empty collision free region, then the item cannot be placed inside the container with the already placed items.

At each iteration, just one randomly chosen modification is applied to the solution. When the placement sequence is modified, two items are swapped in the list. Otherwise, a random item is chosen and either its orientation or its position is changed. The three types of modification (placement sequence, orientation and position modifications) have the same probability of being chosen.

A container with fixed dimensions is adopted and the objective function is the container wasted space, i.e. unoccupied area. This objective function assumes only a discrete set of values [18]. A geometric cooling scheme, in which the temperature is multiplied by a factor $\alpha < 1$ after the occurrence of a given number of accepted solutions, is used. The local stop condition is satisfied when, at a given temperature, the number of accepted solutions or objective function evaluations reach a threshold value. The global stop condition coincides with the simulated annealing convergence, which is checked before each temperature change, and is satisfied when, at the given temperature, computed objective function values for all solutions are equal to the lowest found (see inner loops of Fig. 9).

4.2. Two-level algorithm

A two-level algorithm is proposed to solve the open dimension packing problem. The inner level consists of the simulated annealing algorithm, which considers the container dimensions as fixed.

The external level controls the value of the open dimension and the initial temperature of the simulated annealing algorithm. If the internal level completes its execution and a feasible layout with all the irregular items placed is found, the external level shrinks the container and resets the simulated annealing algorithm. When no such layout is found, the open dimension of the container is increased and the internal level is restarted. For the external level, two parameters r_{dec} and r_{inc} are used to respectively control

```
x ← <Initial random solution>
T ← T_0, L ← L_0
while <Not finished> do
  while <Global stop condition not satisfied> do
    T ← T * α
    while <Local stop condition not satisfied> do
      val ← random(0, 1)
      if val < 1/3 then
        x* ←<Modify placement sequence>
      else
        <Select the moveable item>
        if val < 2/3 then
          x* ←<Select a new item orientation>
        else
          x* ←<Select a vertex from CFR to place the item>
        end if
      end if
      ΔE = F(x*) − F(x)
      if ΔE < 0 then
        x ← x*
      else
        if random(0, 1) < e^(−ΔE/kT) then
          x ← x*
        end if
      end if
      if <All items are placed inside the container> then
        L ← L * (1 − r_dec)
        <Global and local stop condition satisfied>
      end if
    end while
  end while
  if <All items are not placed inside the container> then
    L ← L * (1 + r_inc), T ← T_0
  end if
end while
<Display solution x >
```

Fig. 9. The proposed algorithm with an internal simulated annealing algorithm.

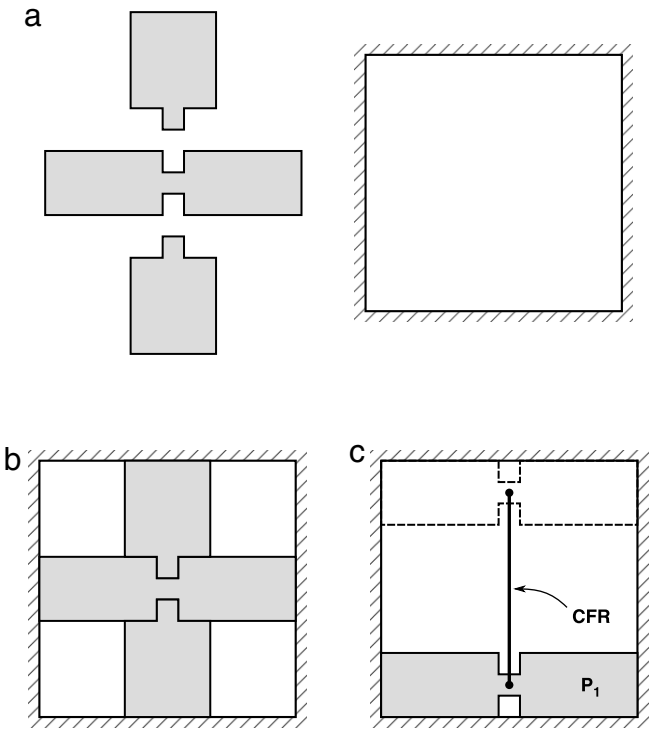

Fig. 10. (a) This placement problem has three items on the left and a rectangular container on the right. (b) The only possible optimal solution using all three items. (c) Collision free region for the largest item, which consists of a single degenerated edge. In order to obtain the optimal solution, the item should be placed in the midpoint of the degenerated edge. Placing the first item on a collision free region vertex will always result in a failed attempt to reach the optimal solution, regardless of the chosen placement order.

the shrinkage and expansion of the container. Fig. 9 shows the proposed algorithm.

Egeblad et al. [12], Imamichi et al. [13] and Leung et al. [14] used a similar two-level approach to solve the strip packing problem. Nevertheless, a relaxed placement method is considered and the objective is to minimize the amount of overlap. In these approaches, the container dimensions are fixed and a search for a feasible layout is performed. When considering only feasible placements, it is possible to directly minimize the length of rectangular enclosure of the layout, thus eliminating the need for an iterative process. In this work, however, a two-level approach is considered in order to take advantage of the adopted placement heuristic, as the container influences the generation of degenerated elements and, as a consequence, also affects the number of exact fits and exact slides. This is further discussed in the following section.

5. Proposed placement heuristics

The collision free region represents all the possible positions for the placement of a new item. Martins and Tsuzuki [18] studied the placement of items at the collision free region boundary. Deterministic placement heuristics are often adopted in the literature with the intention of decreasing the solution space. The primary deterministic placement heuristic is the bottom left heuristic, which adopts the bottom left placement for every item [24].

The collision free region was originally introduced by Martins and Tsuzuki [15]. They determined the collision free region using conventional Boolean Operators and considered containers with fixed dimensions, which were 20% larger than the total area of items. The movable item was placed along the collision free region boundary. Martins and Tsuzuki [25] combined a simulated annealing algorithm with deterministic heuristic (bottom–left and larger first). They showed that it is possible to create problem instances in which a global optimum layout is not reachable once a deterministic heuristic is adopted. Martins and Tsuzuki [18] increased the probability of placing the movable item in one of the collision free region vertices and Martins and Tsuzuki [17] exclusively placed items on vertices of the collision free region. Sato et al. [16] implemented a robust algorithm to determine the collision free region using 2D non-manifold Boolean operations.

Results from these previous works showed that placing items exclusively in the collision free region vertices greatly improved the performance without compromising the quality of final layouts. Accordingly, this limitation is also applied in this work. There are, however, special cases in which the algorithm is not capable of finding the optimal solution when adopting this limitation (see Fig. 10). Nonetheless, these cases are unlikely to appear in real situations. In the proposed placement heuristics, a collision free region vertices classification dictates the movable item placement, as some types of vertices are prioritized while others are discarded as will be explained as follows.

5.1. Ignoring concave vertices

Consider only two convex items, one fixed and the other movable. The no-fit polygon induced by the fixed item to the movable item is also a convex polygon. It is possible to note that, when the movable convex item is placed in a no-fit polygon vertex, the items have only one contact point (see Fig. 11).

Now consider the case in which there are more convex fixed items, and one convex movable item. To obtain the forbidden translations, one can determine the obstructed region, a result of the union of all no-fit polygons induced by fixed items. The result of a union operation of convex polygons may result in a

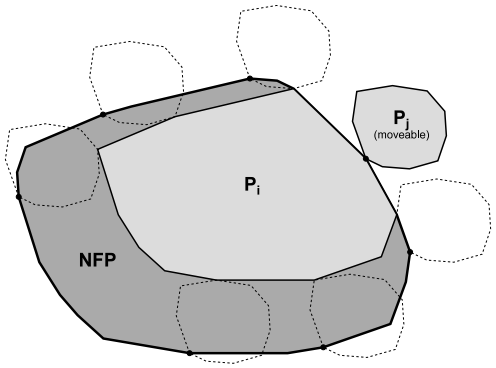

Fig. 11. No-fit polygons of two convex items. Placement at no-fit polygon vertices causes items to touch. Regardless of which vertex is chosen, items P_i and P_j always have only one contact point.

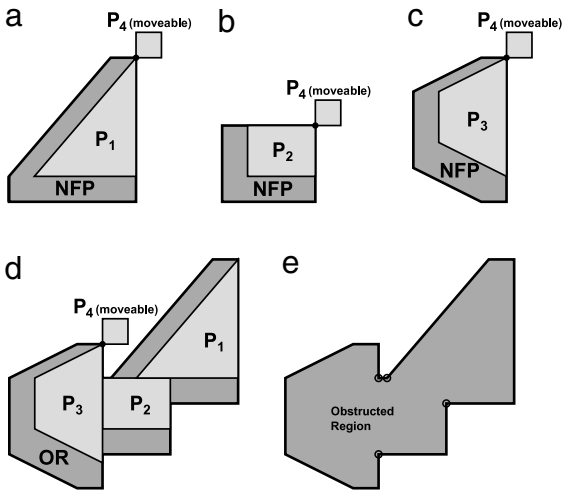

Fig. 12. No-fit polygon induced by convex fixed item (a) P_1; (b) P_2; (c) P_3. (d) Obstructed region for item P_4 and the set of items $\{P_1, P_2, P_3\}$. (e) Circled vertices indicate intersection points between fixed items. It can be observed that all these vertices are concave. Also, all convex vertices originate from the no-fit polygons and represent one contact point positions.

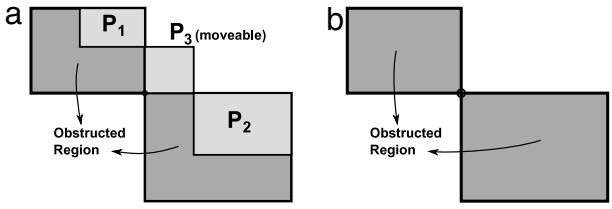

Fig. 13. (a) Item placed at an intersection point of two regions belonging to the obstructed region. P_3 has two contact points (upper right and bottom left corners) even when placed at a convex vertex of the obstructed region. (b) Obstructed region with a circled vertex showing the intersection point.

non-convex polygon. It can be noted that obstructed region vertices are either vertices from original polygons or intersection points. Vertices created from intersection points can be observed to be concave vertices (see Fig. 12). It can thus be said that convex vertices originated from the input no-fit polygons and represent one-contact point positions. One exception is when two separated regions touch, as can be seen from the example from Fig. 13. In this case, the vertex can be treated as concave.

For concave items, the same conclusions can be drawn if observed that, by performing a convex decomposition, the obstructed region will still be the result of a union of no-fit polygons induced by convex fixed items to a convex movable item.

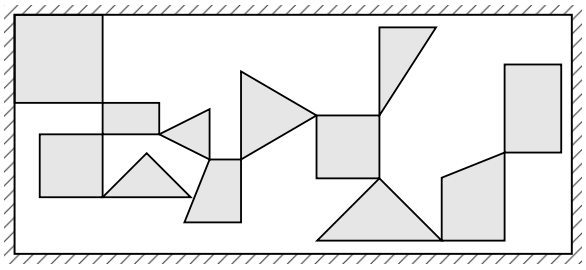

Fig. 14. Items placed at concave vertices of the collision free region. The layout has a very low density.

The collision free region is obtained by subtracting the obstructed region from the inner-fit polygon. The concave vertices of the collision free region always originate from the convex vertices of the obstructed region. Hence, vertices of the collision free region represent positions in which the item touches other items at only one point. These are generally undesired placements, as it usually leaves more unoccupied space between pieces that may not be fulfilled (see Fig. 14). For this reason, in the proposed algorithm, concave vertices are ignored in the placement of an item.

It can be noticed that the final layout shown in Fig. 15 can be obtained through several different placement sequences. Some of the possible sequences use concave vertex placement, as shown in Fig. 15(b), and several others use convex vertices exclusively, i.e. the case illustrated in Fig. 15(c)–(e). Even though the solutions are different, as distinct placement sequences represent exactly the same layout. In the current time, a problem instance in which the placement at a concave vertex is strictly necessary to obtain the best solution was not found. This fact supports the supposition that concave vertices can be ignored.

5.2. Exact fit and exact slide

An exact fit occurs when the item position is fully constrained by its surroundings, and an exact slide is when the item position is constrained in all but one direction. The idea is that layouts in which most items are fixed are more prone to having higher compaction.

Degenerated edges and vertices represent such positions. As explained in Section 3.1, there are three types of degenerated elements depending on their origin: no-fit polygon generated, obstructed region generated and container generated.

A no-fit polygon degenerated element represents positions where two items lock together and are desirable positions. Some authors proposed algorithms that obtain these type of degeneracies [10,26]. Although it is considered to be of great importance, it is not as frequently as other types of degeneracies, as the items must match in shape and size. Artificial data sets are more likely to have two items locking together, as the Jakobs2 data set (see Fig. 16) [27], than real garment factory sets. Another condition is that at least one of the items must be concave in order to obtain a degenerated no-fit polygon.

Obstructed region degeneracies depend on the placement order and translations applied to items. For this type of degeneracy, the existence of a concave item is not required. However, when there are only convex items, identical layouts may be obtained without exact fit or exact slide by changing placement order (see Fig. 17).

Container generated degeneracies are required to obtain a satisfactory solution. They represent positions where the item is placed touching the container. In a fixed width container, it is usually a good strategy to limit items vertical movement, which is often the result of a compaction algorithm. Fig. 18 shows a compaction example from [8] in which the result is a layout with

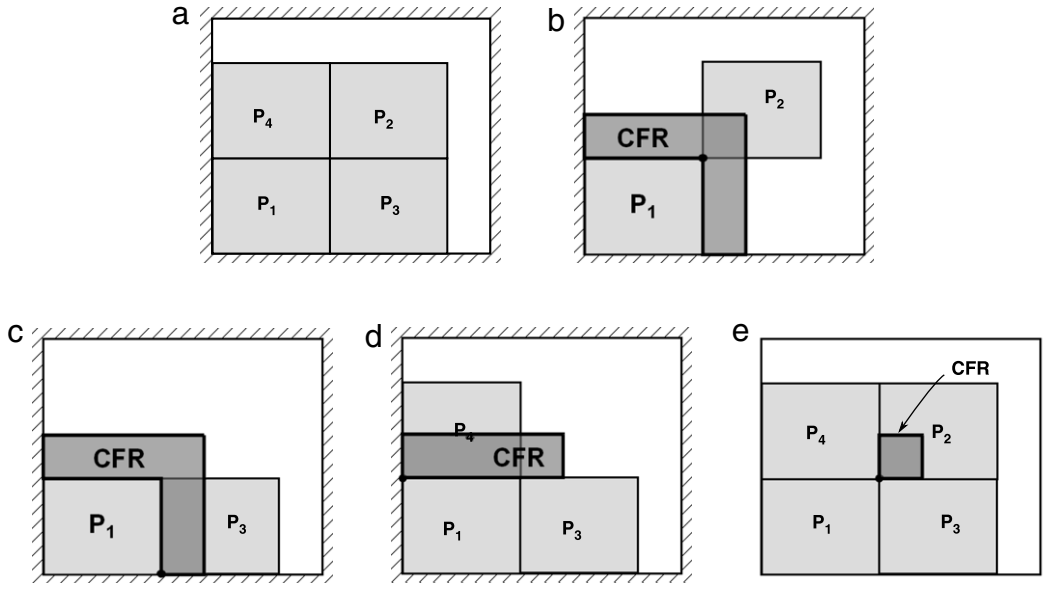

Fig. 15. Example that can be solved using concave placement. (a) Final solution, all four items placed. (b) Possible placement of a second item in a concave vertex of the collision free region. The second and third items can only be placed at exactly sliding positions. (c)–(e) Same solution obtained by performing only convex vertex placement.

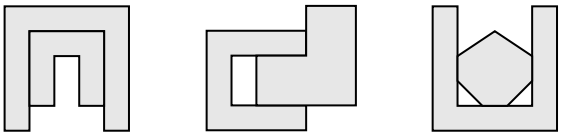

Fig. 16. Example of exact slides in which two items lock together. Items extracted from Jakobs2 data set [27].

more vertically fixed items. This behavior can also be encouraged by placing items in container-generated degeneracies. As with obstructed region generated degeneracies, its existence is very dependent on the placement order. Consider a set of contacting fixed items with only horizontal movement; it is possible to note that the last item was placed in a degenerated element, and that element is either an obstructed region generated or container generated (see Fig. 19). Therefore, both types of degeneracies are important to obtain more compact layouts.

Previous works dealt with puzzle problems, in which the container has fixed dimensions and the optimal solution is known to have no wasted space. For these problems, exact fitting placement was prioritized and results showed great improvement as compared to other works with no priority placement [16].

5.3. Adopted placement heuristic

The adopted placement heuristics have the following order of priority: degenerated vertex, degenerated edge, convex contour vertex. If the collision free region has more than one degenerated vertex, the algorithm chooses one randomly. If no degenerated vertex is found, then the placement should occur in one degenerated edge vertex, randomly chosen. If no degenerated elements exist, one convex vertex from the boundary randomly chosen is selected, with no priority order.

6. Computational results

Benchmark data sets found on ESICUP's (EURO Special Interest Group on Cutting and Packing) website[1] were used to test the proposed algorithm. The sets consist of irregular items and a container with fixed width. Possible orientations for the items are also specified. These data sets are frequently used to evaluate algorithms for the irregular open dimensional problem. Table 1 shows the characteristic of the data sets.

The algorithm was implemented in C++ and compiled by GCC 4.4.4. Computational tests were conducted on a PC with a core i7 860 processor and 4 GB memory.

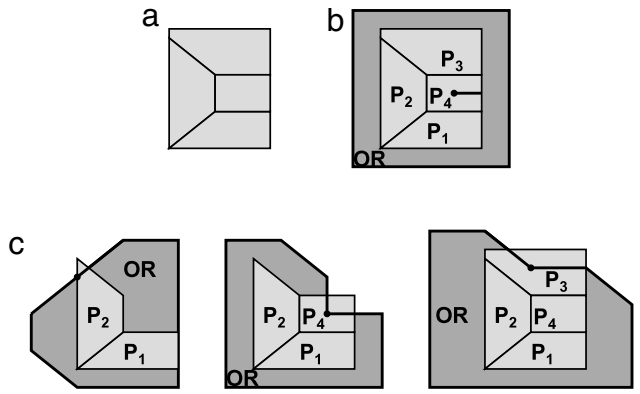

Fig. 17. Example of an obstructed generated (OR) exact slide. (a) Final layout. (b) Placement of item P_4 on a degenerated edge. (c) When the placement order is changed, no item is placed on degenerated elements.

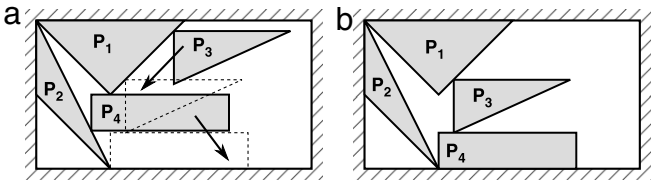

Fig. 18. Compaction example from [8]. (a) Original layout. (b) Layout after compaction. Item P_4 is now limited to horizontal movement, the same as items P_1 and P_2.

6.1. Parameters

The proposed algorithm adopts a two-level approach. Both levels have their own parameters. The inner level, a simulated

[1] http://paginas.fe.up.pt/~esicup/tiki-index.php.

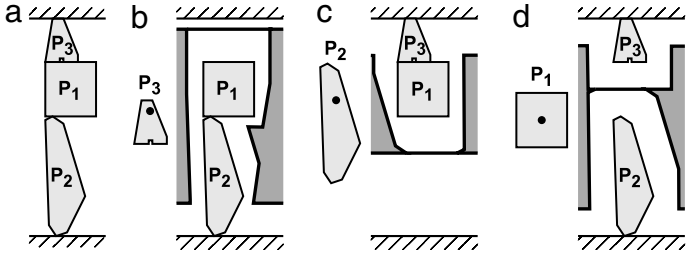

Fig. 19. Collision free region (filled with dark gray) with a degenerated edge for the final item using different placement orders. (a) Final layout. (b) and (c) Container generated degenerated edge. (d) Obstructed region generated degenerated edge.

Table 1
Benchmark data sets information. *TNP* = total number of polygons, *ANV* = average number of vertices per polygon, *AO* = admissible orientations.

Case	*TNP*	*ANV*	*AO*
Albano	24	7.25	0°, 180°
Dagli	30	6.30	0°, 180°
Dighe1	16	3.87	0°
Dighe2	10	4.70	0°
Fu	12	3.58	0°, 90°, 180°, 270°
Jakobs1	25	5.60	0°, 90°, 180°, 270°
Jakobs2	25	5.36	0°, 90°, 180°, 270°
Mao	20	9.22	0°, 90°, 180°, 270°
Marques	24	7.37	0°, 90°, 180°, 270°
Shapes0	43	8.75	0°
Shapes1	43	8.75	0°, 180°
Shapes2	28	6.29	0°, 180°
Shirts	99	6.63	0°, 180°
Swim	48	21.90	0°, 180°
Trousers	64	5.06	0°, 180°

annealing algorithm, has the following parameters:

- T_0: initial temperature.
- α: geometric cooling schedule factor.

In previous works that used the same simulated annealing algorithm and aimed to solve puzzle cases, tests were performed to investigate the influence of these parameters [18]. From these tests, we adopted $\alpha = 0.99$ and $N_{acc} = 2000$. Also based on these tests, the value of T_0 was calculated such that the number of accepted solutions at initial temperature is approximately 90% of the total number of solutions.

The outer level has the following parameters:

- L_0: initial length of container.
- r_{dec}: ratio by which the length of the container is decreased. $(0 < r_{dec} < 1)$.
- r_{inc}: ratio by which the length of the container is increased. $(0 < r_{inc} < 1)$.

Ratios were considered fixed and attributed values of 0.01 and 0.003 for r_{dec} and r_{inc}, respectively. The decreasing procedure was encouraged by giving a higher r_{dec} value as opposed to r_{inc}. The initial length of the container was set to approximately 50% greater than the length of the best solutions published previously in the literature.

6.2. Results

Results found by the proposed algorithm are compared with the best in the literature in Table 2. Both density and minimum length are compared. Best results are found in [28,12,8,13,14].

The Albano, Dagli, Jakobs and Marques solutions found by the proposed algorithm are the best results published in the literature. As for the Jakobs1 set, it has the same density as the one published by Egeblad et al. [12]. The algorithm was also capable of achieving 100% density for the problems Dighe1 and Dighe2. Final layouts are presented in Fig. 20 and execution times in Table 8.

Table 2
Solutions for benchmark data sets. I: Imamichi et al. [13]. B: Bennell and Song [28]. E: Egeblad et al. [12] G: Gomes and Oliveira [8]. L: Leung et al. [14].

Case	Proposed algorithm		Best in literature	
	ML	Density (%)	*ML*	Density (%)
Albano	**9758.70**	**89.21***	9838.70 (L)	88.48
Dagli	**57.40**	**88.36***	57.56 (L)	88.11
Dighe1	**100.00**	**100.00***	**100.00** (*BGL*)	**100.00***
Dighe2	**100.00**	**100.00***	**100.00** (*BGL*)	**100.00***
Fu	30.99	91.96	**30.97** (*E*)	**92.03***
Jakobs1	**11.00**	**89.09***	**11.00** (*EL*)	**89.09***
Jakobs2	**22.75**	**84.83***	23.39 (I)	82.51
Mao	1749.88	84.23	**1731.26** (*E*)	**85.15***
Marques	**76.85**	**90.01***	77.04 (E)	89.82
Shapes0	59.03	67.59	**58.30** (*I*)	**68.44***
Shapes1	55.02	72.52	**53.00** (*L*)	**75.29***
Shapes2	25.93	83.30	**25.64** (*I*)	**84.25***
Shirts	61.65	87.59	**60.18** (*B*)	**89.69***
Swim	6162.43	71.78	**5864.24** (*L*)	**75.43***
Trousers	241.83	90.07	**241.23** (*E*)	**90.46***

* The data sets are the best results in the literature (*ML* = Minimum length).

Table 3
Beam search [28] algorithm compared with the proposed method. Beam search reached better results in 6 cases and the proposed method in 12 cases.

Case	Proposed algorithm		Beam search	
	ML	Density (%)	*ML*	Density (%)
Albano	**9758.70**	**89.21**	9905.88	87.88
Dagli	**57.40**	**88.36**	57.65	87.71
Dighe1	**100.00**	**100.00**	**100.00**	**100.00**
Dighe2	**100.00**	**100.00**	**100.00**	**100.00**
Fu	**30.99**	**91.96**	31.57	90.28
Jakobs1	**11.00**	**89.07**	11.40	85.96
Jakobs2	**22.75**	**84.83**	24.01	80.40
Mao	**1749.88**	**84.23**	1753.20	84.07
Marques	**76.85**	**90.01**	77.79	88.92
Shapes0	**59.03**	**67.59**	62.00	64.35
Shapes1	55.02	72.52	**55.00**	**72.55**
Shapes2	**25.93**	**83.30**	26.57	81.29
Shirts	61.65	87.59	**60.21**	**89.69**
Swim	6162.43	71.78	**5894.72**	**75.04**
Trousers	241.83	90.07	**241.00**	**90.38**

Individual algorithm comparisons are displayed in Tables 3–7. Results show that the proposed algorithm performed better in most comparisons. In the most critical case, ELS produced 6 better layouts whereas the proposed algorithm produced 6 more efficient solutions.

6.3. Discussion

Table 9 shows a comparison of execution times. Gomes and Oliveira [8] measured the average time of execution of 20 instances. As Bennell and Song [28] use a deterministic approach, the execution time for each benchmark problem is unique. Egeblad et al. [12] and Imamichi et al. [13] use a two-level approach and,

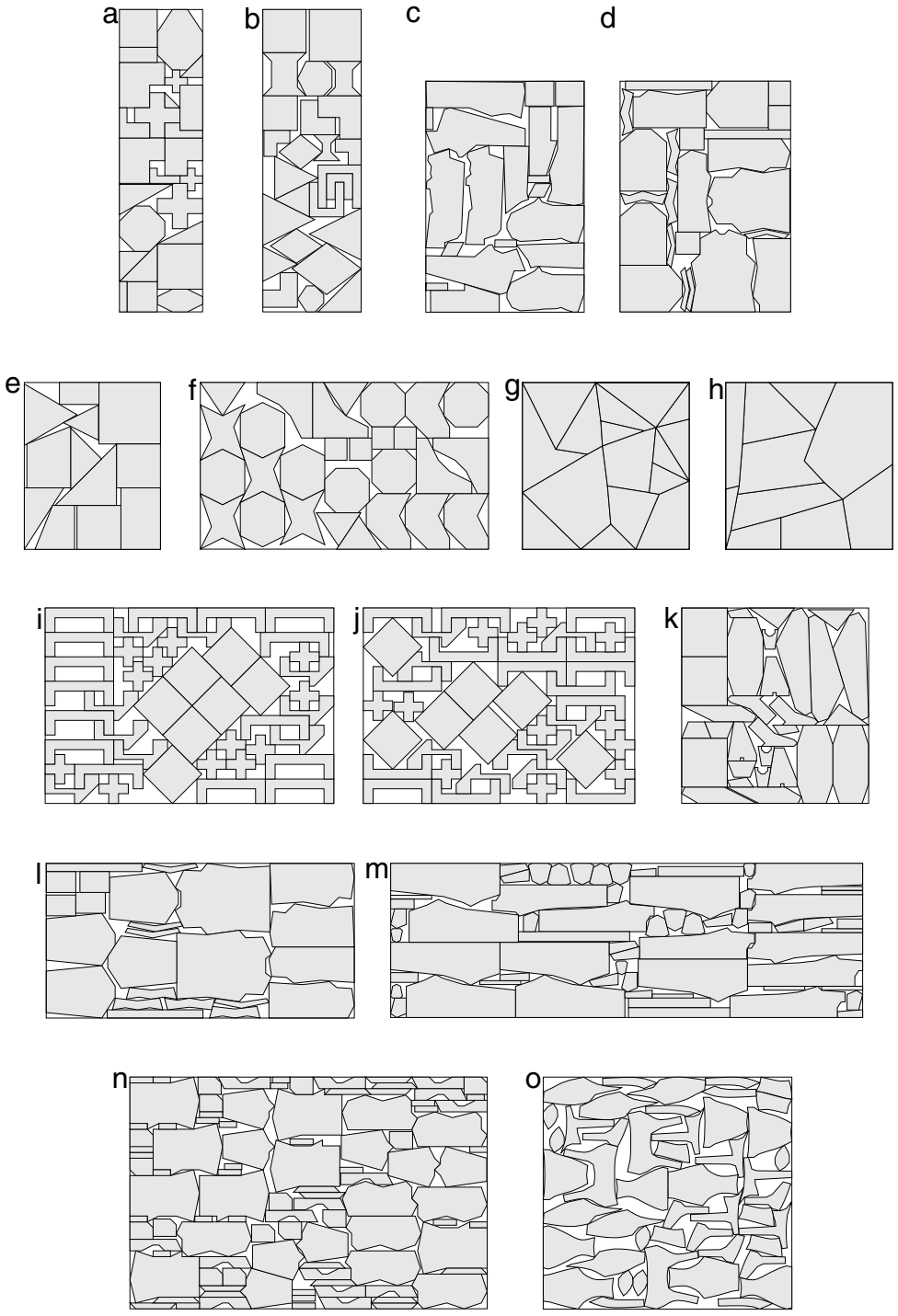

Fig. 20. Results for benchmark data sets: (a) Jakobs1; (b) Jakobs2; (c) Mao; (d) Marques; (e) Fu; (f) Shapes2; (g) Dighe1; (h) Dighe2; (i) Shapes0; (j) Shapes1; (k) Dagli; (l) Albano; (m) Trousers; (n) Shirts; (o) Swim.

therefore, it can run indefinitely. In both cases, a time limit was set. Imamichi et al. [13] ran the tests for 10 min, while Egeblad et al. [12] adopted two limits, 10 min and 6 h.

For the proposed algorithm in this work, the most accepted time measure procedure would be to adopt a single or, at maximum, two different time limits, akin to other two-level approaches. However, initial tests indicated that execution time for the internal level greatly varied for each benchmark problem instance. It was hence not possible to adopt one adequate global time limit, as adopting smaller values would imply that no solution for larger problems could be found and for greater values then smaller problems would run for much more time than needed to find a satisfactory solution. Different from other approaches, an initial feasible layout is not necessary; thus a smaller container was adopted, with its length set to 5% greater than the best solution in the literature. In order to achieve a reasonable execution time, which allows comparison with known algorithms, the total running time was limited to 21,600 s, except for the more complex cases Shapes1, Trousers, Shirts and Swim. This value was adopted by Egeblad et al. [12], and is the highest found in the studied literature tests. So as to increase the number of internal level executions, α was set to 0.90 for most cases, again excluding Shapes1, Trousers, Shirts and Swim.

Table 4
SAHA [8] algorithm compared with the proposed method. SAHA reached better results in 3 cases and the proposed method in 14 cases.

Case	Proposed algorithm		SAHA	
	ML	Density (%)	*ML*	Density (%)
Albano	**9758.70**	**89.21**	9957.41	87.43
Dagli	**57.40**	**88.36**	58.20	87.15
Dighe1	**100.00**	**100.00**	**100.00**	**100.00**
Dighe2	**100.00**	**100.00**	**100.00**	**100.00**
Fu	**30.99**	**91.96**	31.33	90.96
Jakobs1	**11.00**	**89.07**	12.41[a]	78.89[a]
Jakobs2	**22.75**	**84.83**	24.97	77.28
Mao	**1749.88**	**84.23**	1785.73	82.54
Marques	**76.85**	**90.01**	78.48	88.14
Shapes0	**59.03**	**67.59**	60.00	66.50
Shapes1	**55.02**	**72.52**	56.00	71.25
Shapes2	**25.93**	**83.30**	25.84	83.60
Shirts	**61.65**	**87.59**	62.22	86.79
Swim	6162.43	71.78	**5948.37**	**74.37**
Trousers	**241.83**	**90.07**	242.11	89.96

[a] The value has been corrected from the one reported in [8] according to [13].

Table 5
2DNEST [12] algorithm compared with the proposed method. 2DNEST reached better results in 6 cases and the proposed method in 10 cases.

Case	Proposed algorithm		2DNEST	
	ML	Density (%)	*ML*	Density (%)
Albano	**9758.70**	**89.21**	9905.94	87.88
Dagli	**57.40**	**88.36**	58.24	87.05
Dighe1	**100.00**	**100.00**	99.86	99.86
Dighe2	**100.00**	**100.00**	99.95	99.95
Fu	30.99	91.96	**30.97**	**92.02**
Jakobs1	**11.00**	**89.07**	**11.00**	**89.07**
Jakobs2	**22.75**	**84.83**	23.80	81.07
Mao	1749.88	84.23	**1731.26**	**85.15**
Marques	**76.85**	**90.01**	77.04	89.82
Shapes0	**59.03**	**67.59**	59.52	67.09
Shapes1	55.02	72.52	**54.04**	**73.84**
Shapes2	**25.93**	**83.30**	26.48	81.59
Shirts	**61.65**	**87.59**	61.77	87.38
Swim	6162.43	71.78	**6097.78**	**72.49**
Trousers	241.83	90.07	**241.23**	**90.46**

Table 6
ILSQN [13] algorithm compared with proposed method. ILSQN reached better results in 5 cases and the proposed method in 10 cases.

Case	Proposed algorithm		ILSQN	
	ML	Density (%)	*ML*	Density (%)
Albano	**9758.70**	**89.21**	9874.48	88.16
Dagli	**57.40**	**88.36**	58.08	87.40
Dighe1	**100.00**	**100.00**	100.11	99.89
Dighe2	**100.00**	**100.00**	100.01	99.99
Fu	**30.99**	**91.96**	31.43	90.67
Jakobs1	**11.00**	**89.07**	11.28	86.89
Jakobs2	**22.75**	**84.83**	23.39	82.51
Mao	**1749.88**	**84.23**	1766.43	83.44
Marques	**76.85**	**90.01**	77.70	89.03
Shapes0	59.03	67.59	**58.30**	**68.44**
Shapes1	55.02	72.52	**54.04**	**73.84**
Shapes2	25.93	83.30	**25.64**	**84.25**
Shirts	61.65	87.59	**60.83**	**88.78**
Swim	6162.43	71.78	**5875.17**	**75.29**
Trousers	**241.83**	**90.07**	242.56	89.79

The literature results from Table 9 correspond to the best in the literature listed in Table 2, not necessarily the minimum average length found. It can be observed that Dagli, Fu, Dighe1 and Dighe2 obtained equal or better average lengths. Albano, Jakobs1, Jakobs2, Marques, Mao, Shapes0, Shapes2 reached competitive results, with less than 2% length increase when compared with the literature cases. It is difficult to perform the same type of comparison for the other 4 cases, due to their considerably higher execution time.

Table 7
ELS [14] algorithm compared with the proposed method. ELS reached better results in 9 cases and the proposed method in 9 cases.

Case	Proposed algorithm		ELS	
	ML	Density (%)	*ML*	Density (%)
Albano	**9758.70**	**89.21**	9838.70	88.48
Dagli	**57.40**	**88.36**	57.56	88.11
Dighe1	**100.00**	**100.00**	**100.00**	**100.00**
Dighe2	**100.00**	**100.00**	**100.00**	**100.00**
Fu	**30.99**	**91.96**	31.00	91.94
Jakobs1	**11.00**	**89.07**	**11.00**	**89.10**
Jakobs2	**22.75**	**84.83**	23.00	83.92
Mao	1749.88	84.23	**1747.80**	**84.33**
Marques	**76.85**	**90.01**	77.09	89.73
Shapes0	59.03	67.59	**58.99**	**67.63**
Shapes1	55.02	72.52	**53.00**	**75.29**
Shapes2	25.93	83.30	**25.65**	**84.23**
Shirts	61.65	87.59	**61.09**	**88.40**
Swim	6162.43	71.78	**5864.24**	**75.43**
Trousers	**241.83**	**90.07**	243.01	89.63

Table 8
Execution times for benchmark data sets.

Case	Initial length	Minimum		
		Length	Time (s)	Iterations
Albano	11 000.00	9758.70	190,342	6,882,412
Dagli	60.57	57.40	629,047	24,092,833
Dighe1	200.00	100.00	1,704	1,160,013
Dighe2	200.00	100.00	1,229	1,792,371
Fu	33.00	30.99	32,497	22,939,567
Jakobs1	12.00	11.00	7,497	544,047
Jakobs2	25.80	22.75	79,496	3,902,892
Mao	1800.00	1747.12	3,195,538	48,421,265
Marques	78.40	76.85	74,614	1,623,420
Shapes0	66.00	59.03	181,204	2,886,548
Shapes1	62.00	55.02	491,459	5,191,092
Shapes2	28.00	25.93	261,004	12,844,782
Shirts	63.80	61.65	2,528,972	2,638,165
Swim	6250.00	6162.43	5,287,061	2,513,751
Trousers	247.00	241.83	1,016,331	3,843,754

The studied data sets can be classified into puzzles, with no wasted area (Dighe1 and Dighe2), and conventional problems (the rest). The puzzles are data sets where the layout associated with the global optimum is known beforehand. The approach proposed here was the first based on a container with fixed dimension manipulation that solved puzzles.

In NP-complete problems, it is very hard to state which method represents the state of art. The benchmarks from the literature are open problems in the sense that no one knows which the best solution is, except for dighe1 and dighe2 (that are puzzles). It was shown that it is possible to create problem instances that cannot be solved by deterministic heuristics, such as bottom–left and larger first. Fig. 10 shows a problem instance in which the heuristic of placing items at collision free region vertices cannot be solved. Thus, it is normal to think that a specific technique can find better solutions than others, but cannot reach the best solution for all problem instances.

Fig. 21 shows the container length at each iteration of the external level. Seeing that the container is only reduced when a solution is found, only points from the graph that lie on a negative slope line represent valid solutions. Analyzing such solutions, one can observe the different times required to reach a good solution, equal or less than the computed average from Table 9, and the minimum solution. A minimum solution is reached on iteration number 44, whereas a solution with length equal to 58.17 was found on the 18th iteration, requiring 59.19% less external

Table 9
Execution comparison for benchmark data sets. Time in seconds. *NoE* = Number of Executions. *Avg Length* = Average length. *Avg Time* = Average time (total time/NoE). I: Imamichi et al. [13]. B: Bennell and Song [28]. E: Egeblad et al. [12] G: Gomes and Oliveira [8]. L: Leung et al. [14].

Case	Proposed algorithm			Best in literature		
	NoE	Avg length	Avg time	NoE	Avg length	Avg time
Albano	7	10 086.52	21,600	10	9962.56 (L)	1203
Dagli	7	58.21	21,600	10	58.79 (L)	1205
Dighe1	4	100.00	600	1	100 (B)	1.4
Dighe2	4	100.00	600	1	100 (B)	0.3
Fu	4	31.83	600	1	31.95 (E)	21,600
Jakobs1	4	11.06	1,800	20	11.02 (E)	600
Jakobs2	7	24.07	5,400	10	23.98 (I)	1200
Mao	7	1816.55	21,600	20	1783.20 (E)	600
Marques	7	78.87	5,400	1	77.99 (E)	21,600
Shapes0	7	61.09	21,600	10	60.02 (I)	1200
Shapes1	7	55.64	908,175	10	53.75 (L)	1212
Shapes2	7	26,73.	5,400	10	26.44 (I)	1200
Shirts	6	61.94	1,812,654	1	60.8 (B)	6217
Swim	6	6246.20	5,993,945	10	5969.49 (L)	1246
Trousers	7	248.26	86,400	1	244.39 (E)	21,600

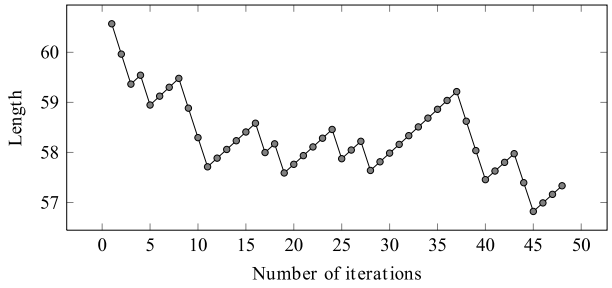

Fig. 21. Container length variation during a sequence of external level iterations for the Dagli case.

level iterations. As each iteration corresponds to a full simulated annealing execution, this disparity is very significant.

7. Conclusions

The problem of minimizing wasted material in the irregular open dimension problem is dealt with. Using a strategy that constructs the solution by placing items sequentially, it is possible to define the collision free region, which indicates permitted placements. The collision free region may contain degenerated edges and vertices that, respectively, represent the exact fits and exact slides for items. These positions are investigated and a new placement heuristic is proposed and applied to the algorithm. The results from the benchmark data sets from the literature showed to be very competitive, finding the best result in the literature in some cases. The two-level proposed algorithm optimizes two variables: wasted space and container length. As future work, it is possible to use a multiobjective technique; and other metaheuristics can also be tested (memetic algorithms, tabu search and others).

Acknowledgments

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

References

[1] Wäscher G, Haussner H, Schumann H. An improved typology of cutting and packing problems. European Journal of Operational Research 2007;183:1109–30.
[2] Bennell JA, Oliveira JF. The geometry of nesting problems: a tutorial. European Journal of Operational Research 2008;184:397–415.
[3] Fowler RJ, Paterson M, Tanimoto SL. Optimal packing and covering in the plane are NP-complete. Information Processing Letters 1981;12(3):133–7.
[4] Kirkpatrick S, Gellat CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220:671–80.
[5] Dowsland K, Dowsland W. Solution approaches to irregular nesting problems. European Journal of Operational Research 1995;84(3):506–21.
[6] Hopper E, Turton BCH. A review of the application of meta-heuristic algorithms to 2D strip packing problems. Artificial Intelligence Review 2001;16:257–300.
[7] Gomes AM, Oliveira JF. A 2-exchange heuristic for nesting problems. European Journal of Operational Research 2002;141:359–70.
[8] Gomes AM, Oliveira JF. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. European Journal of Operational Research 2006;171:811–29.
[9] Oliveira JF, Gomes AM, Ferreira JS. TOPOS—a new constructive algorithm for nesting problems. OR Spectrum 2000;22:263–84.
[10] Bennell JA, Song X. A comprehensive and robust procedure for obtaining the no-fit polygon using Minkowski sums. European Journal of Operational Research 2008;35:267–81.
[11] Burke EK, Hellier RSR, Kendall G, Whitwell G. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. Operations Research 2006;54:587–601.
[12] Egeblad J, Nielsen BK, Odgaard A. Fast neighborhood search for two- and three-dimensional nesting problems. European Journal of Operational Research 2007;183:1249–66.
[13] Imamichi T, Yagiura M, Nagamochi H. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. Discrete Optimization 2009;6:345–61.
[14] Leung SC, Lin Y, Zhang D. Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. Computers & Operations Research 2012;39(3):678–86.
[15] Martins TC, Tsuzuki MSG. Simulated annealing applied to the rotational polygon packing. In: Proceedings of the 12th IFAC symposium information control problems in manufacturing. 2006. p. 475–80.
[16] Sato AK, Martins TC, Tsuzuki MSG. Rotational placement using simulated annealing and collision free region. In: Proceedings of the 10th IFAC workshop on intelligent manufacturing systems. 2010. p. 253–8.
[17] Martins TC, Tsuzuki MSG. Placement over containers with fixed dimensions solved with adaptive neighborhood simulated annealing. Bulletin of the Polish Academy of Sciences Technical Sciences 2009;57:273–80.
[18] Martins TC, Tsuzuki MSG. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. Expert Systems with Applications 2010;37:1955–72.
[19] Art RC. An approach to the two-dimensional irregular cutting stock problem. Tech. rep. IBM Cambridge Scientific Centre. 1966.
[20] Mahadevan A. Optimization in computer-aided pattern packing (marking, envelopes). Ph.D. Thesis. North Carolina State University. 1984.
[21] Agarwal PK, Flato E, Halperin D. Polygon decomposition for efficient construction of Minkowski sums. Computational Geometry 2002;21:39–61.
[22] Hobby JD. Practical segment intersection with finite precision output. Computational Geometry: Theory and Applications 1999;13(4):199–214.
[23] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. The Journal of Chemical Physics 1953;21:1087–92.

[24] Dowsland KA, Vaid S, Dowsland BW. An algorithm for polygon placement using a bottom–left strategy. European Journal of Operational Research 2002; 141:371–81.
[25] Martins TC, Tsuzuki MSG. Irregular rotational placement of shapes over non-convex containers with fixed dimensions. In: Proceedings of the 8th IFAC workshop on intelligent manufacturing systems. 2007.
[26] Burke EK, Hellier RSR, Kendall G, Whitwell G. Irregular packing using the line and arc no-fit polygon. Operations Research 2010;58:948–70.
[27] Jakobs S. On genetic algorithms for the packing of polygons. European Journal of Operational Research 1996;88:165–81.
[28] Bennell JA, Song X. A beam search implementation for the irregular shape packing problem. Journal of Heuristics 2010;16:167–88.