Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 171 (2006) 811–829

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Solving Irregular Strip Packing problems by hybridising simulated annealing and linear programming ☆

A. Miguel Gomes [a,b,*], José F. Oliveira [a,b]

[a] Faculdade de Engenharia da Universidade do Porto, DEEC Rua Dr. Roberto Frias s/n, Porto 4200-465, Portugal
[b] Instituto de Engenharia de Sistemas e Computadores do Porto Campus da FEUP, Rua Dr. Roberto Frias 378, Porto 4200-465, Portugal

Available online 14 October 2004

Abstract

In this paper a hybrid algorithm to solve Irregular Strip Packing problems is presented. The metaheuristic simulated annealing is used to guide the search over the solution space while linear programming models are solved to generate neighbourhoods during the search process. These linear programming models, which are used to locally optimise the layouts, derive from the application of compaction and separation algorithms.

Computational tests were run using instances that are commonly used as benchmarks in the literature. The best results published so far have been improved by this new hybrid packing algorithm.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Cutting; Packing; Simulated annealing; Irregular Strip Packing; Nesting

1. Introduction

The Irregular Strip Packing problem belongs to the more general class of combinatorial optimisation problems—the Cutting and Packing problems. In these problems, one or more big items, either material or space, must be divided into smaller pieces. Usually the objective is the waste minimisation, i.e. the portions of the big items that are not used to produce small pieces. In Cutting and Packing problems, on the one hand, decisions have to be taken on which pieces to produce and from which big item, which is a hard combinatorial problem. On the other hand, a specific geometric problem arises: how to cut the small pieces. These two problems are interlaced and solutions must be feasible both from the "quantitative" viewpoint (i.e. minimum or maximum number of small pieces to produce, availability of big items) and from the geometric viewpoint (i.e. no overlapping between the small pieces, containment of the small pieces inside the big items).

☆ Partially supported by FCT, POSI and FEDER (POSI/40908/SRI/2001 (CPackMO)).
* Corresponding author.
E-mail address: agomes@fe.up.pt (A.M. Gomes).

Cutting and Packing problems are mainly characterised by: the number of relevant dimensions, the regularity or irregularity of the shapes of the small pieces and big items, the big items assortment and availability, the total number of small pieces and the number of different types of small pieces. For a detailed classification of Cutting and Packing problems see Dyckhoff et al. (1988). Additional information on Cutting and Packing is available online on the ESICUP—EURO Special Interest Group on Cutting and Packing web site (`http://www.apdio.pt/esicup`).

The economic and ecologic impact of the Cutting and Packing problems is obvious in many industrial production processes and better solving these problems contributes to a better usage of natural resources. Examples of industrial Cutting and Packing problems are:

- 1D Cutting-Stock problems in the Paper industry;
- 2D Rectangular Cutting in the Furniture, Home Textile and Glass industries;
- 2D Irregular Packing in the Furniture, Shoe and Garment industries;
- 3D Container and Truck Loading.

The focus of this work is on the 2D Irregular Strip Packing problem, also known as nesting problem, and specially on a variant of the problem that arises in the garment industry. In its most general formulation, the nesting problem is a two dimensional Cutting and Packing problem where both the small and the big pieces have irregular (non-rectangular) shapes. Small pieces can be placed with an arbitrary orientation.

Nesting problems in the garment industry have several characteristics that are specific to this industry. These characteristics are inherent to the production process and to several technological issues. In this type of industry, the plates are big rolls of fabric with a total length up to several kilometres. This leads to the consideration of a single rectangle with a fixed width and an infinite length (a strip). The objective is the minimisation of the layout length. In the garment industry usually only two orientations are allowed for the small pieces: the original and the one obtained by a 180° rotation. This restriction is due to the existence of drawing patterns and to intrinsic characteristics of the fabric's weave. Finally, in the garment industry the total number of small pieces can rise above 100, but usually from less than 20 different types, although with very different sizes. In this paper, an approach that takes advantage of these characteristics is proposed. An example of a nesting layout or cutting pattern is represented in Fig. 1. A more detailed description of nesting problems and cutting and packing problems can be found in Dowsland and Dowsland (1995).

In the industrial environment this problem is usually tackled by experienced workers that "manually" (with the help of CAD systems) build the layouts. The quality of the layouts produced by these specialised workers is high and, presently, automatic solutions can only barely match this level of quality. A list of (semi-)automatic commercial solutions is available in Hopper (2000).

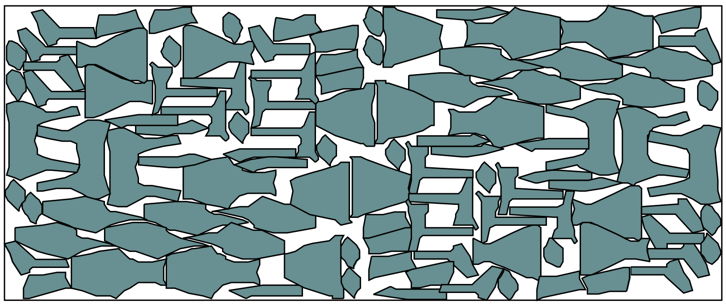


Fig. 1. Nesting layout example.

In the literature we can find several algorithms that follow different approaches to tackle the different issues present in nesting problems. For some of those issues there are techniques that clearly out-perform all the others available. That is the case of the no-fit-polygon concept, used to ensure the geometric feasibility of a layout, and the case of the algorithms based on linear programming compaction models, used to obtain layouts that are local optima (Li and Milenkovic, 1995; Stoyan et al., 1996; Bennell and Dowsland, 2001). There are also a few simple greedy heuristics that are very effective in achieving a "good" layout rapidly (Daniels and Milenkovic, 1996; Gomes and Oliveira, 2002). In order to avoid local optima, metaheuristics have become very popular in this field (Oliveira and Ferreira, 1993; Błażewicz and Walkowiak, 1995; Heckmann and Lengauer, 1995; Bennell and Dowsland, 1999; Gomes and Oliveira, 1999; Hopper, 2000; Gomes and Oliveira, 2001; Bennell and Dowsland, 2001; Błażewicz et al., 2004). A very promising approach is the use of hybrid algorithms that combine metaheuristics and linear programming compaction models (Bennell and Dowsland, 2001). This is the approach followed in this work, where a simulated annealing algorithm is used to guide the search through the solution space, being the neighbourhood structure based on linear programming compaction models.

This paper is structured as follows. The next section has a description of the basic blocks of the proposed nesting algorithm: the no-fit-polygon concept, the greedy bottom-left placement heuristic and the compaction/separation algorithms. The following section is used to present the hybrid algorithm, with a detailed description of its main components: the simulated annealing algorithm, the neighbourhood structure—LocalCompact—and the multi-stage approach. Finally, computational results are presented and some conclusions drawn.

2. Blocks to build nesting algorithms

Algorithms based on a high level search algorithm need to perform a fairly wide search over the solution space to ensure the quality of the results obtained. This implies that the basic operations of such algorithms must be efficiently performed. For the nesting problem, building and evaluating layouts are the most important basic operations. Tasks such as avoiding overlap between two pieces, layout compaction and placing pieces inside the plate are included in these operations. The no-fit-polygon is used to efficiently avoid overlapping among the pieces and to place pieces inside the plate. The task of generating the initial layout is performed by a greedy bottom-left placement heuristic. This heuristic uses a sequence of pieces and places each one on the plate, with the ability of filling holes. Linear Programming models are used to compact layouts (Compaction Model), improving their quality and removing in feasibilities (Separation Model). These models are intentionally kept linear and non-integer [1] so that they can be easily solved by a standard linear programming package. These models are the core of the Compaction and Separation algorithms.

It is also necessary to have an adequate geometric representation and manipulation of the pieces. Pieces are represented by convex and non-convex polygons, with oriented edges, and any curve is approximated by a set of exterior tangent edges. The edges are oriented in such a way that the interior of the polygon is on the right-hand side of the edge. Holes inside the pieces are not allowed. In a pre-processing phase, all the pieces are simplified to eliminate very narrow concavities and, therefore, avoid unnecessary computational effort.

2.1. The no-fit-polygon

The no-fit-polygon concept is used to ensure feasible layouts, i.e. layouts where the pieces do not overlap and fit inside the plate. This concept was first introduced by Art (1966) and later on used by several authors. In other fields of knowledge this concept is also known as Minkowski sums. Mahadevan (1984) presents a comprehensive description of an algorithm for the no-fit-polygon generation based on a sliding scheme. More recently, Bennell et al. (2001) published an

[1] Although, as a consequence, in some situations the layouts obtained are not as compact as it would be possible.

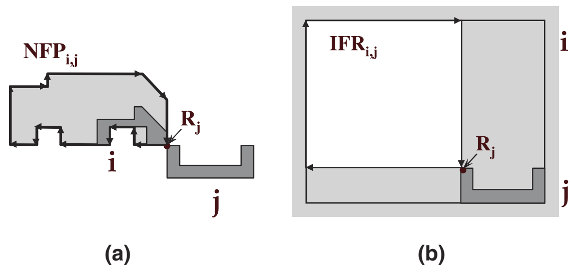


Fig. 2. No-fit-polygon and inner-fit-rectangle examples: (a) the no-fit-polygon, $NFP_{i,j}$ and (b) the inner-fit-rectangle, $IFR_{i,j}$.

alternative algorithm to compute the no-fit-polygon based on Minkowski sums. In the present work a geometric library with an implementation of Mahadevan's algorithms, previously developed and used by the authors Oliveira et al. (2000) and Gomes and Oliveira (2002), was used.

The no-fit-polygon of piece j relative to piece i ($NFP_{i,j}$) is the locus of points traced by the reference point[2] of piece j (R_j), when j slides along the external contour of piece i. The relative orientations of pieces i and j are kept fixed during this orbital movement. Piece j (the orbital piece) must never intersect piece i (the stationary piece) and they must always be in contact (Fig. 2(a)). The no-fit-polygons are represented by oriented polygons, where the feasible positioning points are on the left-hand side of the edges (the exterior of the polygon).

From this definition it follows that:

- if the reference point of piece j (R_j) is placed in the *interior* of $NFP_{i,j}$ then piece j *overlaps* piece i;
- if the reference point of piece j (R_j) is placed on the *boundary* of $NFP_{i,j}$ then piece j *touches* piece i;
- if the reference point of piece j (R_j) is placed in the *exterior* of $NFP_{i,j}$ then piece j does *neither overlap nor touch* piece i.

The problem of finding the relative position of two polygons is transformed into the simpler problem of finding the relative position of one point and one polygon.

To achieve a feasible (without overlap) and tight layout, each piece should have its reference point on the boundary of at least one no-fit-polygon (relative to another piece) and in the exterior or in the boundary of all the other no-fit-polygons (relative to the remaining pieces).

The inner-fit-rectangle is a concept derived from the no-fit-polygon concept and represents the feasible set of points for the placement of one polygon inside a rectangle. This concept is used to ensure that all the pieces are placed inside the plate. The inner-fit-rectangle of piece j relative to rectangle i ($IFR_{i,j}$) is obtained when the piece j (the orbital piece) slides along the internal contour of the rectangle i (Fig. 2(b)). It is assumed that the plate is larger than the biggest piece to place, i.e. that the $IFR_{i,j}$ always exists. The inner-fit-rectangle is also represented by an oriented polygon, where the feasible positioning points are on the right-hand side of the edges (the interior of the inner-fit-rectangle).

2.2. Greedy bottom-left placement heuristic

The greedy bottom-left heuristic is a placement heuristic that converts a sequence of pieces into a feasible layout. Pieces are placed inside the plate, one by one, at the most bottom-left feasible placement point. When positioning one piece the heuristic takes into account the previously placed pieces, both to avoid overlap and to fill holes left empty at earlier stages. In Gomes and Oliveira (2002), different criteria to generate the sequence of pieces are discussed. In this work, a new criterion is proposed: *random weighted length*. This criterion generates the sequence of pieces by randomly selecting the next piece to add to the sequence. The probability of selecting one piece is proportional to that piece's length.

To place each piece, the greedy heuristic builds a set of placement point candidates, which includes all the following points:

1. the vertices of the no-fit-polygons ($NFP_{i,k}$) where k is the next piece to place and i stands for all the previously placed pieces ($i = 1, \ldots, k-1$);

[2] The reference point of a piece is the origin of the coordinate system (0,0), over which the other vertices coordinates, of the same piece, are defined.

2. the vertices of the inner-fit-rectangle ($IFR_{p,k}$) between the next piece to place (k) and the plate (p);
3. the intersections between any two edges of any pair of no-fit-polygons described in item 1 ($NFP_{i,k}$ and $NFP_{j,k}$; $i = 1,\ldots, k-1$; $j = 1,\ldots, k-1$; $i \neq j$);
4. the intersections between one edge of any no-fit-polygon described in item 1 ($NFP_{i,k}$; $i = 1,\ldots, k-1$) and one edge of the inner-fit-rectangle described in item 2 ($IFR_{p,k}$).

From this set, the points which lead to unfeasible placements, i.e. points that are in the interior of any no-fit-polygon or on the exterior of the inner-fit-rectangle, are eliminated, so that a set of admissible placement points is obtained. It is trivial to obtain the most bottom-left placement point of this set. This heuristic is described in detail in Gomes and Oliveira (2002).

Finally, it should also be mentioned that the no-fit-polygons and the inner-fit-rectangles can be computed off-line, once they only depend on the shape of the pieces and do not depend on the actual places occupied by the pieces in the layout. When different orientations are allowed it is necessary to calculate no-fit-polygons for each pair of different orientations and an inner-fit-rectangle for each orientation.

2.3. Compaction algorithm

One of the most powerful methods available to obtain high quality layouts is the use of linear/integer programming compaction models. Compaction models can improve layouts by applying a set of coordinated continuous motions to the pieces, achieving layouts that are local optima (Fig. 3). In a compacted layout the main structure of the non-compacted layout is preserved, i.e. the relative positions of pairs of pieces are kept, [3] although the exact position of each piece may be completely different. During this set of motions the pieces are not allowed to overlap. Moreover,

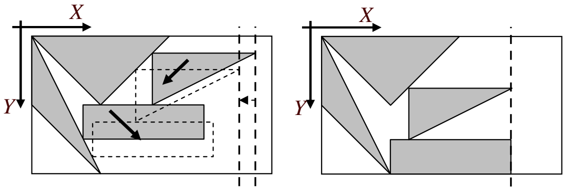


Fig. 3. Compaction example.

the motion of a piece must be continuous, i.e. pieces can not jump over other pieces. Rotations are not tackled. A proof of the Compaction Problem NP-hardness is available in Li and Milenkovic (1993).

To use compaction models it is necessary to have an initial feasible layout. [4] Then the first step is to analyse the layout and establish positioning relationships between each pair of pieces. These relationships are based on the corresponding no-fit-polygons. As stated in Section 2.1, piece j does not overlap piece i if the reference point of piece j is placed over or on the left-hand side of at least one of the oriented edges of $NFP_{i,j}$. [5] For a particular feasible placement of pieces i and j (i.e. without overlap) one or more edges of $NFP_{i,j}$ will satisfy the previous condition. As, for each pair of pieces, only one constraint is needed, different rules may be used to select the edge/constraint that is added to the model (the so called active constraint). These different rules lead to different implementations of the compaction model. In the second step the model is solved by a standard linear programming package and in the final step the layout is reanalysed in order to detect further compaction possibilities. This iterative process is needed because, after a successful compaction, pieces are positioned in different coordinates and the use of new sets of active constraints may allow further improvements. This iterative process stops when no improvement in the layout can be

[3] Preserving relative positions means that if piece i is on the left-hand side of piece j before the compaction, this relative position still holds after compaction.

[4] This initial layout can be obtained in several ways, e.g. a greedy heuristic, randomly placing the pieces on the plate or manually by an experienced worker. The initial layout can even be the result of a previous compaction operation.

[5] This is only generally true for convex pieces. The generalisation to non-convex pieces is presented in the end of this section.

achieved. Usually very few compaction iterations (less than 10) are necessary to completely compact a layout.

Different goals can instantiate the generic concept of "layout quality". From the compaction viewpoint they all have in common being modelled as an application of an adequate set of forces to the pieces. Examples of these goals are:

- increasing nesting efficiency by shortening the layout length, which is achieved by applying a right-left force to all the pieces;
- creating space, in the middle of the layout, to place new pieces without increasing the layout length. This can be obtained by applying forces with different directions to the pieces. The direction of the force for a particular piece depends on the relative position of that piece regarding the space to be opened (i.e. if the piece is above the space, we should apply a bottom-up force to the piece).

In the present work, we use the strategy of increasing the nesting efficiency by shortening the layout length.

The general linear programming model for compacting layouts is presented in Fig. 4. This general model has been used by other authors (Li and Milenkovic, 1995; Stoyan et al., 1996; Bennell and Dowsland, 2001). Its actual implementations differ from author to author and are explained at the end of this section. In this model, the decision variables x_i and y_i are the coordinates of the placement point of each piece, z is an auxiliary variable that stands for the layout length and N is the number of pieces in the layout. All the other constants in the model are described in Fig. 5. Eqs. (1) and (2) ensure the layout length minimisation. The set of constraints defined by Eq. (3) limits the movement of the pieces, in each one of the compaction iterations, to a maximum value ($DISTANCE_i$) that depends on the dimensions of each piece, ensuring a smooth compac-

Compaction Model:

$$\min z \tag{1}$$

subject to:

$$z \geq x_i + W_i \qquad i = 1,\ldots,N \tag{2}$$

$$\begin{array}{rcll} x_i - X_i^* & \leq & DISTANCE_i & i = 1,\ldots,N \\ y_i - Y_i^* & \leq & DISTANCE_i & i = 1,\ldots,N \\ -x_i + X_i^* & \leq & DISTANCE_i & i = 1,\ldots,N \\ -y_i + Y_i^* & \leq & DISTANCE_i & i = 1,\ldots,N \end{array} \tag{3}$$

$$\begin{array}{rcll} x_i & \geq & 0 + W_i^* & i = 1,\ldots,N \\ y_i & \geq & 0 + H_i^* & i = 1,\ldots,N \\ x_i & \leq & W_0 - W_i & i = 1,\ldots,N \\ y_i & \leq & H_0 - H_i & i = 1,\ldots,N \end{array} \tag{4}$$

$$\mathrm{f}(x_j - x_i, y_j - y_i) \leq 0 \qquad i,j = 1,\ldots,N; i \neq j \tag{5}$$

$$x_i, y_i, z \geq 0 \qquad i = 1,\ldots,N \tag{6}$$

Fig. 4. Linear programming model for compacting layouts.

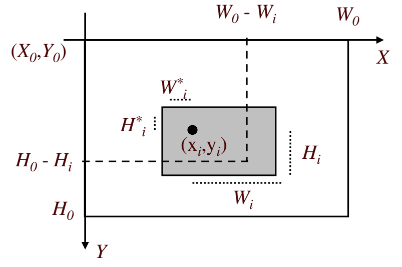


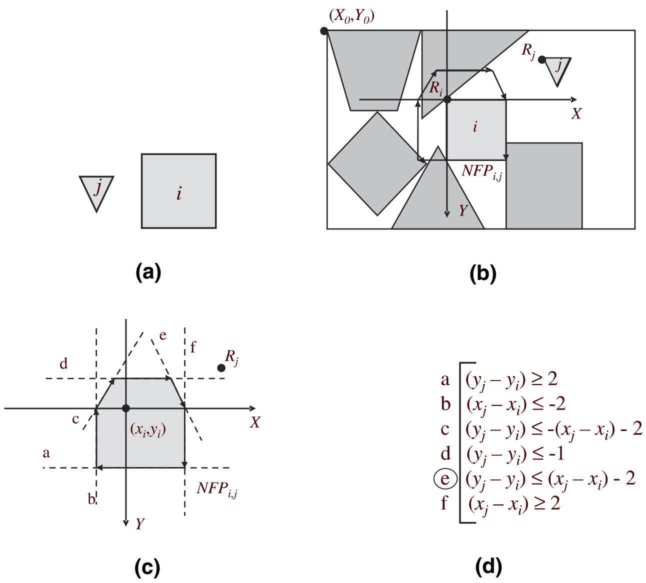
Fig. 5. Placing one piece inside the plate.

tion process. X^*_i and Y^*_i are constants that are initialised with the coordinates of the placement point of piece i. Placing all the pieces inside the plate is guaranteed by the set of constraints defined by Eq. (4). The set of constraints defined by Eq. (5) ensures that pieces do not overlap. When pieces i and j are placed at coordinates (x_i,y_i) and (x_j,y_j), respectively, $f(x_j - x_i, y_j - y_i)$ is defined as

$$f(x_j - x_i, y_j - y_i) = \begin{cases} -d_1^2, & \text{if pieces } i \text{ and } j \text{ do not overlap,} \\ 0, & \text{if pieces } i \text{ touches } j, \\ d_2^2, & \text{if pieces } i \text{ and } j \text{ overlap.} \end{cases} \quad (7)$$

In Eq. (7), d_1 is the distance between the placement point of piece j and the farthest edge of no-fit-polygon $NFP_{i,j}$ and d_2 is the distance between the placement point of piece j and the nearest edge of no-fit-polygon $NFP_{i,j}$. In Fig. 6 the instantiation of this model is illustrated. Two pieces i and j (Fig. 6(a)) do not overlap if and only if the placement point of piece j, R_j, is over or on the left-hand side of at least one oriented edge of $NFP_{i,j}$ (in the example of Fig. 6(c) this condition holds for edges d, e and f). When more than one oriented edge satisfies the above condition, the one that is more distant from R_j is selected. In the example of Fig. 6(c) edge e is selected and the constraint associated with edge e is added to the model (Fig. 6(d)).

Two particular cases need special attention, since they correspond to saddle points. The first one is when the placement point, R_j, is over a vertex of the no-fit-polygon $NFP_{i,j}$ (e.g. the intersection of edges a and b in Fig. 6(c)). In this case we should add to the model a disjunction of the constraints associated with the two edges (a,b). However, due to efficiency purposes (to avoid mixed-integer formulations) only one edge is added: the one with the smallest slope. The second case is similar and occurs when the placement



a $(y_j - y_i) \geq 2$
b $(x_j - x_i) \leq -2$
c $(y_j - y_i) \leq -(x_j - x_i) - 2$
d $(y_j - y_i) \leq -1$
$\boxed{e}$ $(y_j - y_i) \leq (x_j - x_i) - 2$
f $(x_j - x_i) \geq 2$

Fig. 6. Associated constraints selection: (a) pieces i and j; (b) no-fit-polygon, $NFP_{i,j}$ (c) edges of the no-fit-polygon and (d) associated constraints.

point R_j is over a vertical edge of the no-fit-polygon $NFP_{i,j}$ (e.g. edge b in Fig. 6(c)). Alike the previous case, we should add to the model a disjunction of the constraints associated with three edges (e.g. edges a, b and c in Fig. 6(c)) to the model. However, only one edge is in fact added to the model: the constraint associated with the vertical edge. These approximations can lead to slightly worse layouts, but they are necessary to keep the global performance of the nesting algorithm at a good level.

A different situation occurs when the no-fit-polygon is non-convex and the edge associated with the selected constraint belongs to a concavity. In these cases, instead of adding a single constraint, it is necessary to add a conjunction of constraints: all the constraints associated with the edges that bound the concavity. [6] This is not critical, because the addition of a conjunction of constraints does not imply the addition of any integer or binary variable to the model. This situation is illustrated in Fig. 7. In this example, the constraint associated with edge c is satisfied. However, the conjunction of constraints associated with edges d, e and f is also satisfied (note that those are the edges that bound the concavity). Since the distance between edge c and the placement point R_j is less than the distance between R_j and the nearest concavity border (edge d), the conjunction of constraints associated with edges d, e and f is chosen to be added to the model. This results in an increase on the total number of constraints, but without any relevant impact in the overall performance of the nesting algorithm.

Similar compaction models have also been used by other authors. Differences between the different implementations can be found on the objective function and on the rule used to select the constraint that is added to the model, when more than one fulfils the requirements. However, the major differences are on the associated separation model and on how the compaction model is used. Later on these differences are made clear. Stoyan et al. (1996) proposes an approach based on the generation of different initial layouts. Each one of these initial layouts must be feasible and they are independently compacted, in order to achieve the best layout. Li and Milenkovic (1995) proposes a model where different forces are applied to the pieces to enlarge the gaps between them, by using adequate coefficients in the objective function. However, the pieces are restricted to be "star-shaped". More recently, Milenkovic and Daniels (1999) propose another model using a mixed-integer formulation. Finally, Bennell and Dowsland (2001) propose a model based on Li and Milenkovic (1995), without any restriction on the shape of the pieces. The objective function is the minimisation of the layout length and redundant constraints are removed.

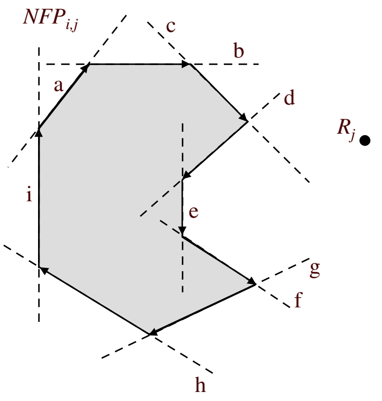


Fig. 7. Adding a conjunction of constraints.

2.4. Separation algorithm

Based on the compaction model, it is possible to develop another linear programming model, the separation model. The idea behind the separation model is to remove any infeasibility (i.e. overlapping situations) by moving apart the pieces that overlap. As in the compaction model this is done by applying a set of coordinated continuous motions to the pieces (Fig. 8).

In layouts with overlapped pieces, the construction of the compaction model fails because it is not possible to establish a valid relationship between two pieces that overlap (Fig. 9). If piece i overlaps piece j, the placement point of piece j (R_j) is always inside the no-fit-polygon $NFP_{i,j}$, i.e. R_j is on the

[6] This concept needs to be recursively extended if a "concavity inside another concavity" is present.

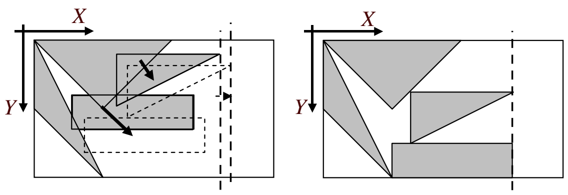


Fig. 8. Separation example.

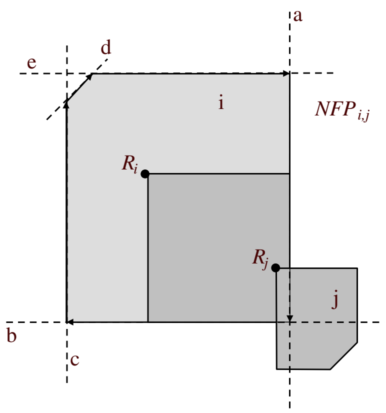


Fig. 9. Active edge example with overlap.

right-hand side of all edges of $NFP_{i,j}$. To obtain a complete model of the problem, the constraint associated with the edge closest to the placement point R_j should be added to the model. However, adding this constraint would naturally lead to an infeasible model. To overcome this difficulty, an artificial variable $a_{i,j}$ is introduced, turning the model feasible. In the example of Fig. 9, the constraint (Eq. (8)) associated with edge a is added to the model, with the artificial variable $a_{i,j}$. The objective function of the separation model is the minimisation of the sum of all artificial variables (Eq. (9)). A feasible layout is obtained when all the artificial variables have a value of zero. Unlike other separation models, by using artificial variables the elimination of overlap does not need to be achieved in just one step. This increases the chances of successfully removing overlap. The separation model has one additional difference when compared with the compaction model: the set of constraints used to keep track of the layout length (Eq. (2) in Fig. 4) is removed.

$$f(x_j - x_i, y_j - y_i) - a_{i,j} \leqslant 0, \tag{8}$$

$$\min \sum_{i,j=1,\dots,N}^{i\neq j} a_{i,j}. \tag{9}$$

Frequently it is not possible to remove the overlap in just one step and the separation model may have to be consecutively applied to obtain a feasible layout. In our computational experiments in 99% of the situations the overlap was removed in less than 10 steps. In each step, the separation model needs to be rewritten accordingly to the new positions of the pieces.

The main drawbacks of the separation model are the sparsity of the layouts after the separation process and the lack of guarantee that overlap is always removed. The first one is easily overcome by applying the compaction algorithm to the layout that results from the separation algorithm. The second drawback can only be overcome if it is possible to increase both dimensions of the plate. Unfortunately, this is not the case of nesting problems where the width of the plate is fixed. However, since the length of the plate is “infinite”, in practice almost all layouts are effectively separated.

Based on the respective compaction models, Li and Milenkovic (1995) and Bennell and Dowsland (2001) also propose separation models. However, the separation model proposed in this work presents a major innovation: the use of a set of artificial variables $a_{i,j}$ and the associated set of constraints defined by Eq. (8). With this modification, the separation process is done more smoothly and does not need to be achieved in just one step, thus increasing the chances of success.

3. Hybrid algorithm

The focus of this work is on the hybridisation of simulated annealing with linear programming models in the resolution of nesting problems. The linear programming models arise in the compaction and separation algorithms presented earlier. These algorithms can be used in a coordinated way, taking advantage of the strong points of each one, to create an efficient neighbourhood structure for nesting problems. The simulated annealing algorithm is used to guide the search over the solution space. Bennell and Dowsland (2001) propose a similar approach where a different metaheuristic, a tabu search algorithm, is hybridised with compaction and separation models. Besides the use of a different metaheuristic, additional differences between the two hybrid approaches can be found on the compaction and separation linear programming models, as pointed out in the previous section, and on the neighbourhood structure. Two additional differences can be found on a more strategic level: the type of movement used to move between neighbour solutions and the acceptance, or not, of unfeasible solutions along the search. In Bennell and Dowsland (2001), moving from one solution to another is done by moving one piece in the layout (insert movement) and infeasible layouts (i.e. layouts with overlap) are allowed. In this work, moving to another solution is done by exchanging the positions of two pieces in the layout (swap movement) and unfeasible layouts are not allowed along the search. [7]

In the next subsections, the main components of the hybrid algorithm are presented: the simulated annealing algorithm and the neighbourhood structure. The last subsection is dedicated to present a multi-stage approach that tries to explore the variety, in terms of size, of the pieces that usually arises in the garment industry.

3.1. Simulated annealing

The origins of simulated annealing can be found in Metropolis et al. (1953), where an algorithm to simulate the cooling of material in a heat bath, a process known as annealing, is presented. Later on, Kirkpatrick et al. (1983) proposed the use of simulated annealing as an approach to tackle combinatorial optimisation problems. Since then, there has been an enormous quantity of work where simulated annealing has been applied to various combinatorial optimisation problems.

Pure local search approaches frequently end up entrapped in local optimal solutions, since they can only move to better solutions in the neighbourhood. Simulated annealing can be seen as an evolution of those approaches, by allowing some controlled uphill movements, in order to achieve global optimality. Accepting a movement to a worse solution depends on a control parameter (the temperature) and on the magnitude of the variation of the objective function, i.e. how worse the solution is.

Many real-life problems have been successfully tackled by approaches based on simulated annealing. The main reasons for this success are: the high quality of the solutions, the easy inclusion of real-life constraints and the robustness of these approaches. The main drawback is the large computational effort needed.

In this work, we use the simulated annealing algorithm described in Pirlot (1996). To apply a simulated annealing algorithm to a particular problem, several decisions must be taken. These decisions can be divided in generic decisions and specific decisions. The generic decisions include the initial temperature (t_0), the cooling scheme, the stopping condition and the plateau length. The specific decisions are the evaluation function, the neighbourhood structure and the initial solution.

After some preliminary tests, the following generic decisions were taken: an initial temperature t_0 calculated so that the probability of accepting worse solutions in the first plateau is greater than 50%; a geometric cooling scheme with a factor of 0.9; two plateaus without improvement as a stopping condition; the plateau length was set equal to the number of combinations of pairs of pieces, given by $\binom{N}{2}$, where N stands for the total number of pieces.

Problem specific decisions were taken based on the blocks presented earlier: the initial solution is obtained by the greedy bottom-left placement heuristic (Section 2.2), being the next piece to place selected according to the *random weighted length* criterion. The neighbourhood structure used in this work is based on the compaction and separation algorithms, presented in the previous section, and is discussed in the following subsection. There is no need to implement any additional evaluation

[7] Layouts with overlap are allowed within a neighbourhood movement, but the new solution is only accepted if the overlap is completely removed (see Section 3.2 for further details).

function to measure the layout length, since it is already calculated both by the greedy bottom-left placement heuristic and by the compaction algorithm.

3.2. Neighbourhood structure—LocalCompact

The use of an adequate neighbourhood structure is a crucial factor to obtain a successful meta-heuristic algorithm. Special care was put on the development of an efficient neighbourhood structure—LocalCompact. This new neighbourhood structure uses the compaction and separation models presented earlier.

The main advantage of the compaction model is the ability to efficiently obtain local optimal layouts. However, it lacks the ability to perform any kind of global optimisation, once the pieces are not allowed to jump over other pieces and temporary infeasible layouts are not allowed. On the other hand, the main advantage of the separation model is the ability of removing overlap and thus obtaining feasible, although sparse, layouts. With these ideas in mind, it is necessary to induce changes in the relative positions of pairs of pieces (i.e. having one or more pieces jumping over other pieces), so that the search produces layouts with different relationships between pairs of pieces.

Within LocalCompact, the small perturbation introduced in the current layout to generate a new neighbour, could be obtained by changing the placing point of one piece or, alternatively, by exchanging the position of two pieces in the layout (Fig. 10(a)–(b)). In both cases overlap is usually created, however, it can be removed by applying the separation model (Fig. 10(c)). Afterwards, the layout is compacted by the compaction model to obtain a local optimum (Fig. 10(d)). The acceptance of this new layout is ruled by a higher level search algorithm.

In preliminary tests, the supremacy of the pieces exchanging perturbation was clear. This happened mainly because the total number of times the separation model failed (i.e. a feasible layout was not achieved) was smaller. The behaviour may be explained as follows: when exchanging two pieces, there is always some free space on the plate (the space occupied by the other piece that is being exchanged), when changing the position of just one piece (an insertion move) the existence of free space at the new position is not guaranteed. Another issue is the decision of what to do to achieve a feasible layout when the separation model fails. Two options arise: try again exchanging two different pieces, or accept the layout and penalise it with an "infinite" length. The first option was chosen.

A neighbourhood structure for nesting problems necessarily needs to be able to rotate pieces in layouts. This is quite easy to achieve with the LocalCompact neighbourhood: when a particular piece is part of a neighbourhood movement, it

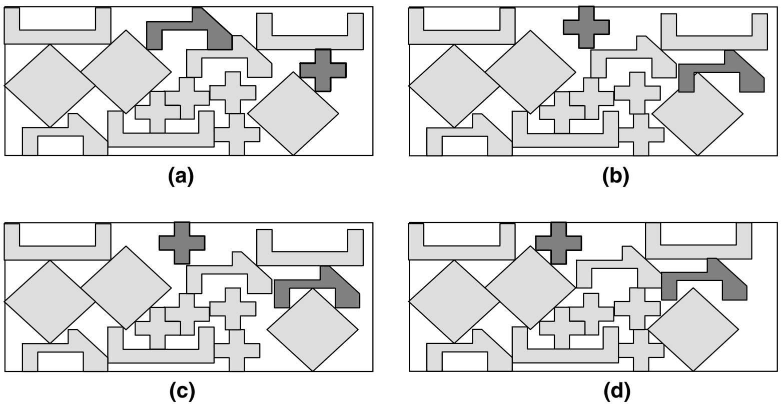


Fig. 10. Neighbourhood structure example: (a) select two pieces randomly; (b) exchange the positions of the selected pieces; (c) layout separation and (d) layer compaction.

is possible to select a different orientation for that piece, which means that the piece may end up rotated. The orientation selection, for a pair of pieces that take part on a neighbourhood movement, is done by trying all combinations of pairs of admissible orientations. The pair of orientations that leads to a better layout is selected. For instance, if for a certain instance two orientations are allowed for the pieces (0° and 180°), then, each time a pair of pieces (P_1 and P_2) is selected to be swapped, 4 alternatives are evaluated: (P_1, 0°; P_2, 0°), (P_1, 0°; P_2, 180°), (P_1, 180°; P_2, 0°), and (P_1, 180°; P_2, 180°).

The LocalCompact neighbourhood structure proved to be a very powerful tool when used on the development of metaheuristics for nesting problems. The use of a linear programming compaction model ensures compacted layouts (i.e. locally optimal for a particular set of relative positions between the pieces). The type of movements allowed in the neighbourhood structure (exchanging positions between two pieces in the layout), ensures that different sets of relative positions between the pieces are investigated.

3.3. Multi-stage approach

As mentioned earlier, one of the specificities of nesting problems in the garment industry is the existence of a very big diversity on the piece sizes. In these cases, the layout length is totally or almost totally determined by the placement of the bigger pieces, while the smaller ones can be easily packed in gaps among them. A multi-stage approach is proposed to take advantage of this characteristic, which aims to reduce the computational effort without compromising the layout quality.

The multi-stage approach divides the available pieces in two groups, accordingly to their relative sizes: one group with the bigger pieces and other group with the smaller ones. A parameter, which is a percentage of the size of the biggest piece, controls this division process. The size of a piece can be measured in many different ways (area, length, width, perimeter,...). However, since the idea is to fill small gaps in the middle of a layout, the obvious choice is the area. This process of dividing pieces in two groups can be recursively used to create more groups. However, the idea is not to divide the pieces in more than 3 or 4 groups. At each stage, a different group of pieces is packed, starting by the groups with the bigger pieces. For the second and subsequent stages, the algorithm chosen to pack the pieces has to take into account the pieces that were packed in the previous stages.

As the results of the first stage are crucial for the overall quality of the final layout, in the implementation proposed in this paper the powerful hybrid algorithm, based on simulated annealing, is used in this stage. However, for the following stages, a faster hybrid algorithm, based on a pure local search mechanism, is used: the 2-exchange heuristic proposed in Gomes and Oliveira (2002). The 2-exchange heuristic searches over sequences of pieces and relies on the previously described greedy bottom-left placement heuristic to generate the actual layouts. Pure local search is used to control the search process. Given a sequence of pieces, each piece is selected and exchanged with all the other pieces of the sequence. In the implementation proposed in this paper, hybridisation is also introduced in this algorithm by generating and solving linear programming compaction models after each run of the greedy bottom-left placement heuristic.

In more detail, each stage starts by using the greedy bottom-left placement heuristic to pack the pieces belonging to that stage. The pieces packed in the previous stages are considered as placed in the coordinates previously determined and gaps among them may be filled by the pieces of the current stage. Then, the faster hybrid algorithm tries to improve the layout, by iteratively exchanging pairs of pieces in the input sequence. In each iteration, the pieces of the current stage are removed from the current layout and the new iteration's sequence is packed again by the greedy bottom-left placement heuristic. At this point, the compaction algorithm is applied to the full layout, including the pieces placed in previous stages. In fact, the positions of these pieces are not considered fixed and can suffer small adjustments, during this compaction phase, that can lead to an overall improvement of the layout. The new layout is accepted if the layout length is less than the previous one. All exchanges between pairs of pieces, in the sequence of pieces to place, are tried and the process is repeated until a full neighbourhood is searched without any improvement. The search also stops when a layout length no greater than the length obtained in the previous stage is achieved. Better solutions might be obtained if the search was not stopped by this reason. However, as achieving better solutions is highly improbable and in order to keep running times as low as possible, the search is effectively stopped when a layout length not greater than the previous stage layout length is achieved. A final remark, about the greedy bottom-left placement heuristic used in the faster hybrid algorithm, should be made. After preliminary tests the original *area* criterion (Gomes and Oliveira, 2002) showed to be more adequate to fill the gaps in the middle of the layout. This is the criterion used in all stages after the first one.

4. Computational results

Preliminary computational experiments were carried out in early stages of this work. These experiments allowed to take several decisions about the implementation and the parameters tuning of the hybrid approach, namely the ones regarding the simulated annealing algorithm, the bottom-left greedy placement heuristic and the compaction and separation algorithms, which have already been discussed in previous sections. What remains to be discussed are $DISTANCE_i$ parameters that used in the compaction model (Fig. 4). This set of parameters restricts the movement allowed for the pieces in each iteration of the compaction and separation algorithms. Observations made during these preliminary experiments have shown their dependence on the size of the pieces, but also the algorithm robustness for small variations of the values of these parameters. Given this, each parameter $DISTANCE_i$ with $i = 1, \dots, N$ (N stands for the number of pieces in the layout) was set equal to half the size of the biggest dimension of the enclosing rectangle of piece i. Additionally, and together with the set of constraints defined by Eq. (3), these parameters have also a positive secondary effect as they allow the removal of several constraints of the linear programming model that will never be active as pieces will never move far enough. These redundant constraints are of two types:

- containment constraints (Eq. (4))—constraints where the smallest distance between piece i and the plate border is bigger than $DISTANCE_i$;
- overlap constraints (Eq. (5))—constraints where the smallest distance between piece i and piece j is bigger than $DISTANCE_i + DISTANCE_j$.

Special care was put on the design of the computational tests to evaluate the performance of the hybrid algorithm. The hybrid algorithm benchmark was done using problem instances taken from the literature and against the best results published for these instances. To evaluate the relative impact of the neighbourhood structure and of the simulated annealing search strategy in the overall quality of the results, a greedy local search hybrid algorithm was also tested. This algorithm was directly obtained from the simulated annealing algorithm by setting the initial temperature to zero: during the search only equal of better solutions are accepted. As both algorithms use the same neighbourhood structure, the comparison of their computational results will allow to reach conclusions about the influence of the simulated annealing search strategy on the quality of the final solutions.

4.1. Problem instances

Nesting problem instances available in the literature were used to evaluate the performance of the hybrid algorithm. These instances have already been used as benchmarks by other researchers. A total of 15 nesting problem instances were collected from Hopper (2000) and Oliveira et al. (2000). The actual data files can be downloaded from the ESICUP web site—*http://www.apdio.pt/esicup*. The main characteristics of these instances are summarised in Table 1.

Problem instances marked with * in Table 1 were scanned from sample layouts and processed with digitising software. This process implies a

Table 1
Nesting problem instances characteristics

Problem instance	Number of different pieces	Total number of pieces	Vertices by piece (average)	Admissible orientations (degrees)	Plate width	Problem type
FU[*]	12	12	3.58	0, 90, 180	38	Artificial, convex
JAKOBS1[*]	25	25	5.60	0, 90, 180	40	Artificial
JAKOBS2[*,a]	25	25	5.36	0, 90, 180	70	Artificial
SHAPES0[†]	4	43	8.75	0	40	Artificial
SHAPES1[†]	4	43	8.75	0, 180	40	Artificial
SHAPES2[†b]	7	28	6.29	0, 180	15	Artificial
DIGHE1[*,c]	16	16	3.87	0	100	Jigsaw puzzle
DIGHE2[*,c]	10	10	4.70	0	100	Jigsaw puzzle
ALBANO[*]	8	24	7.25	0, 180	4900	Garment
DAGLI[*]	10	30	6.30	0, 180	60	Garment
MAO[*]	9	20	9.22	0, 90, 180	2550	Garment
MARQUES[*,d]	8	24	7.37	0, 90, 180	104	Garment
SHIRTS[†]	8	99	6.63	0, 180	40	Garment
SWIM[†]	10	48	21.90	0, 180	5752	Garment
TROUSERS[†]	17	64	5.06	0, 180	79	Garment

[a] Scaling factor of 2 when compared with the original publication.
[b] Problem instance also known as *BLAZ*.
[c] Problem instance similar to the original publication.
[d] Scaling factor of 5 when compared with the original publication.
[†] Problem instance coordinates stated in Oliveira et al. (2000).
[*] Problem instance scanned from sample layout (Hopper, 2000).

degree of inaccuracy, which, however, does not compromise the evaluation of the algorithms performance. See Hopper (2000) for details on the scanning process and for references to the papers where the layouts have been published.

The nesting instances of Table 1 can be divided in three different types: artificial, jigsaw puzzle and garment. Instances belonging to the first type have been artificially created. Jigsaw puzzles are instances where all the pieces fit perfectly and so it is possible to achieve a 100% efficiency. Instances of this type are interesting because the optimum is known in advance. Finally, garment type instances are real-life instances taken from the garment industry. In all 15 instances the pieces are represented by polygons. In some cases the polygonal shape is an approximation of the real shape.

4.2. Results

The computational tests were performed on a personal computer with a *Pentium IV* processor running at 2.4GHz and with 512Mb of RAM. The program was coded in *C* and *ILOG CPLEX 8.0* was used to solve the linear programming models.

The computational tests consisted on 20 runs for each instance, both for the simulated annealing hybrid algorithm (SAHA) and the greedy local search hybrid algorithm (GLSHA). The results obtained are summarised in Table 2. In this table, the first column is the instance name. The next eight columns refer to the results of the two algorithms. Each group of four columns contains the results obtained for one of the algorithms, respectively: the best layout length, the best layout efficiency, [8] the average layout length of the 20 runs and the average running time of the 20 runs. Finally, the last two columns present the best layout length published in the literature and the algorithm that achieved that result. In Table 3 a comparison between the two hybrid algorithms and between these and the best result published in the literature (BRP) is presented. The first set of three columns refers to the comparison of the best solutions, while the

[8] The efficiency is measured as the quotient between the area of the pieces packed and the used rectangular area of the plate.

Table 2
Hybrid algorithms results

Problem instance	GLSHA				SAHA				Best result published in the literature	
	Layout			Average time (seconds)	Layout			Average time (seconds)		
	Best		Average length		Best		Average length			
	Length	Efficiency (%)			Length	Efficiency (%)			Length	Algorithm
FU	32.54	87.57	33.17	47	31.33	90.96	32.70	296	34.00	Hybrid[a]
JAKOBS1	12.00	81.67	13.30	37	12.42	75.89	12.93	332	13.00	Caga[b]
JAKOBS2	26.00	74.23	26.37	55	24.97	77.28	25.86	454	28.20	HoopperSA[c]
SHAPES0	62.00	64.35	64.09	621	60.00	66.50	63.15	3914	63.00	Jostling[d]
SHAPES1	57.00	70.00	58.71	1944	56.00	71.25	58.17	10314	59.00	2-Exchange[e]
SHAPES2	26.40	81.82	26.92	348	25.84	83.60	26.53	2257	27.30	2-Exchange
DIGHE1	120.96	82.67	130.87	18	100.00	100.00	122.00	83	138.13	NestLib[f]
DIGHE2	100.00	100.00	130.45	3	100.00	100.00	119.53	22	134.05	HopperGA[g]
ALBANO	10074.08	86.41	10476.80	209	9957.41	87.43	10280.05	2257	10122.63	HopperSA
DAGLI	59.32	85.49	61.11	797	58.20	87.15	59.41	5110	65.60	NestLib
MAO	1819.40	81.01	1851.68	667	1785.73	82.54	1842.70	8245	2058.60	HopperGA
MARQUES	80.49	85.94	82.44	791	78.48	88.14	79.63	7507	83.60	HopperNE[h]
SHIRTS[*]	62.21	86.80	63.09	5290	62.22	86.79	63.03	10391	63.13	2-Exchange
SWIM[†]	6040.25	73.24	6156.68	2724	5948.37	74.37	6121.39	6937	6568.00	NestLib
TROUSERS[†]	242.89	89.67	246.34	5080	242.11	89.96	244.68	8588	245.75	2-Exchange

[a] Hybrid approach in Fujita et al. (1993).
[b] CAGA algorithm in Hifi and M'Hallah (2003).
[c] Simulated annealing algorithm in Hopper (2000).
[d] Jostling algorithm in Dowsland et al. (1998).
[e] 2-Exchange algorithm in Gomes and Oliveira (2002).
[f] Commercial nesting software NestLib cited in Hopper (2000).
[g] Genetic algorithm in Hopper (2000).
[h] Naive evolution algorithm in Hopper (2000).
[†] Solved with the multi-stage approach with two stages.
[*] Solved with the multi-stage approach with three stages.

Table 3
Hybrid algorithms comparisons

Problem instance	Best solution improvement (%)			Average solution improvement (%)			Average time $\left(\frac{\text{SAHA}}{\text{GLSHA}}\right)$
	GLSHA vs. BRP	SAHA vs. BRP	SAHA vs. GLSHA	GLSHA vs. BRP	SAHA vs. BRP	SAHA vs. GLSHA	
FU	4.28	7.84	3.72	2.43	3.82	1.43	6.24
JAKOBS1	7.69	4.44	−3.53	−2.31	0.52	2.76	8.88
JAKOBS2	7.80	11.44	3.95	6.49	8.31	1.95	8.29
SHAPES0	1.59	2.54	0.97	−1.73	−0.23	1.47	6.30
SHAPES1	3.39	5.08	1.75	0.49	1.41	0.93	5.31
SHAPES2	3.30	4.39	1.13	1.39	2.81	1.43	6.14
DIGHE1	12.43	27.60	17.33	5.25	11.68	6.78	5.57
DIGHE2	25.40	25.40	0.00	2.69	10.83	8.37	6.33
ALBANO	0.48	1.63	1.16	−3.50	−1.56	1.88	10.82
DAGLI	9.57	11.29	1.90	6.85	9.44	2.78	6.41
MAO	11.62	13.23	1.82	10.05	11.29	1.37	10.61
MARQUES	3.72	6.12	2.49	1.39	4.75	3.41	9.50
*SHIRTS**	1.46	1.44	−0.01	0.06	0.15	0.09	6.40
*SWIM**	8.03	8.70	0.72	6.26	6.80	0.57	5.05
*TROUSERS**	1.16	1.48	0.32	−0.24	0.43	0.67	6.68
Average	6.80	8.84	2.25	2.37	4.70	2.39	7.13

* Solved with the multi-stage approach.

following set of three columns concerns the comparison between the averages of the 20 runs. The last column shows the quotient of the computational times of SAHA and GLSHA algorithms.

The overall quality of the layouts produced by SAHA and GLSHA is outstanding. The best results published in the literature were improved by both hybrid algorithms for all the instances used in the computational tests. The average improvement of the best result published, when considering all problem instances, is 8.84% for SAHA and 6.80% for GLSHA. Furthermore, the average of the 20 runs of SAHA still improves the best result published in 13, out of 15, instances and the average of the 20 runs of GLSHA improves the best result published for nine instances. In what concerns the comparison between SAHA and GLSHA, SAHA achieved the best layout in 13 instances and only in two instances it was beaten by GLSHA (*JAKOBS1* and *SHIRTS*). This result was not expected, given the more complex search strategy used in SAHA, but as SAHA has an important random component, this can always happen. However, when average values of the 20 runs are considered, SAHA outperforms GLSHA for all problem instances.

As expected, the computational times of both algorithms are very different and, within each algorithm, the computational times are greatly dependent on the characteristics of the problem instances. The computational times are directly related with the main characteristics of the problem instances (Table 1). This relation is stronger when the total number of pieces, admissible orientations and number of vertices by piece are considered, and is weaker in what concerns the number of different pieces. The computational times of SAHA (based on simulated annealing) and GLSHA (based on greedy local search) are very different: SAHA takes 5–11 times more computational time than GLSHA. The average quotient between GLSHA average computational time and SAHA average computational time is 7.13. To keep this compari-

Table 4
Number of pieces in each stage

Problem instance	Stage		
	First	Second	Third
SHIRTS	24	45	30
SWIM	24	24	–
TROUSERS	28	36	–

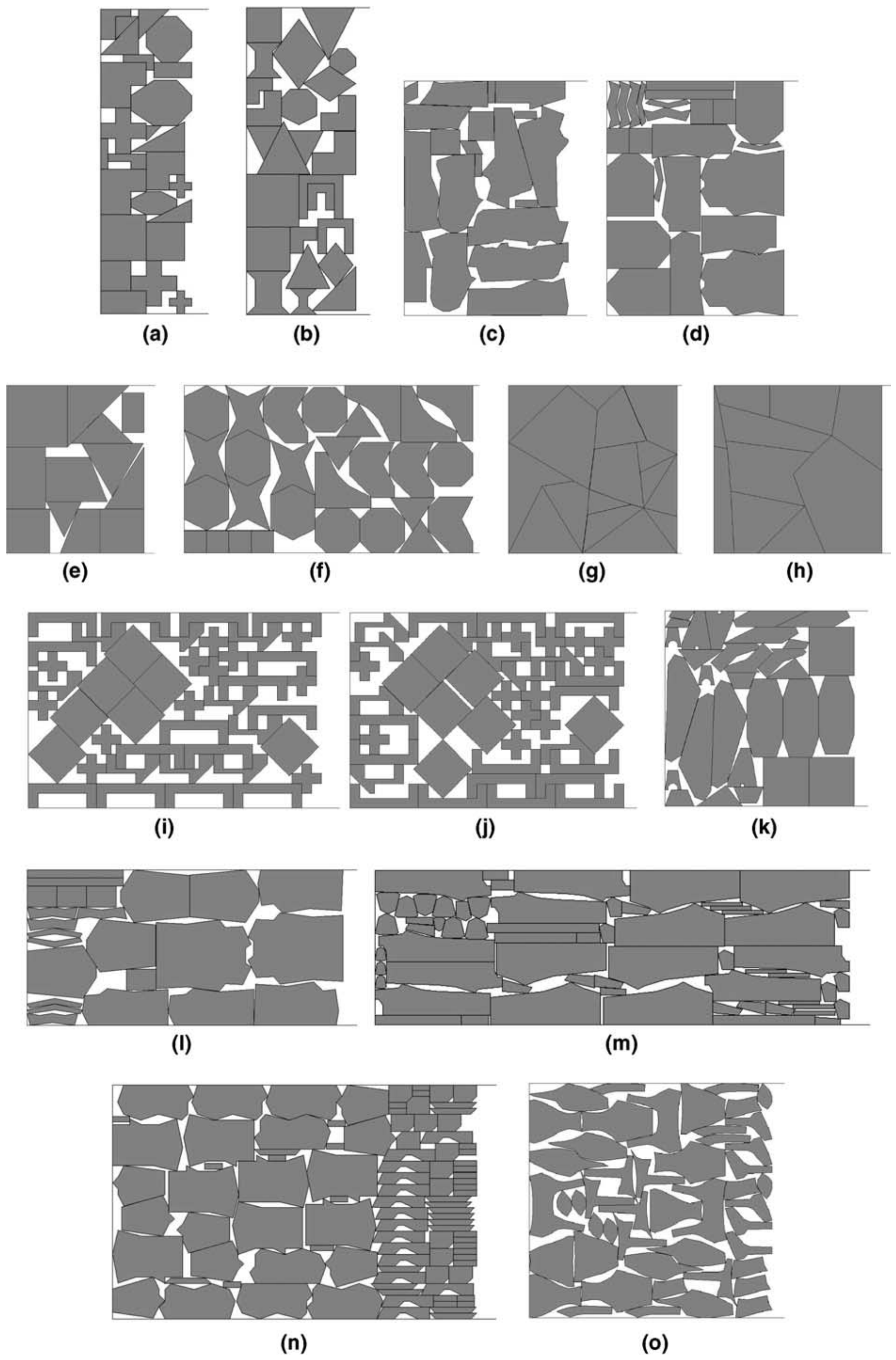


Fig. 11. Best layouts: (a) *JAKOBS1*; (b) *JAKOBS2*; (c) *MAO*; (d) *MARQUES*; (e) *FU*; (f) *SHAPES2*; (g) *DIGHE1*; (h) *DIGHE2*; (i) *SHAPES0*; (j) *SHAPES1*; (k) *DAGLI*; (l) *ALBANO*; (m) *TROUSERS*; (n) *SHIRTS*; (o) *SWIM*.

son fair, in the instances marked with * in Table 3, where the multi-stage approach has been used, only the computational times of the first stage were considered.

Instances *SHIRTS, SWIM* and *TROUSERS* were solved using the multi-stage approach, because they are very large and include pieces of very different sizes. By doing so, the running times were kept within reasonable limits. The number of pieces selected to be packed in each stage, for each problem instance, is presented in Table 4. The pieces were grouped according to their area. For these instances the results presented in Table 2 include all the stages, i.e. computational times presented are the sum of computational times of all stages and layout lengths refer to the layout length obtained after the last stage.

The overall conclusion that can be taken from these results is the impressive adequacy of the neighbourhood structure LocalCompact for the resolution of nesting problems. With a simple greedy local search procedure, very good layouts can be obtained in a fair amount of time, while outstanding results are achieved with a more complex search strategy, as simulated annealing, in a larger amount of time. Finally, the best layout obtained in the computational tests for each problem instance is presented in Fig. 11.

5. Conclusions

In this work a new hybrid nesting algorithm is proposed. The use of a neighbourhood structure based on the exchange of pieces on the layout, followed by compaction and separation algorithms, which imply the resolution of several linear programming models in each step, proved to be an extremely effective approach when used together with a simulated annealing algorithm to guide the search over the solution space. The best results previously published in the literature were improved for all problem instances used in the computational tests. The simulated annealing hybrid algorithm computational times are relatively high, specially with the large instances taken from the garment industry. However, a greedy local search strategy, with the same neighbourhood structure, can be used to produce very good layouts in a more reasonable amount of time.

Acknowledgment

The authors would like to thank the anonymous referees for their valuable remarks that helped us to enhance the quality of this paper.

References

Art Jr., R.C., 1966. An approach to the two-dimensional, irregular cutting stock problem. Technical Report 36.Y08, IBM Cambridge Scientific Centre.

Bennell, J.A., Dowsland, K.A., 1999. A tabu thresholding implementation for the irregular stock cutting problem. International Journal of Production Research 37 (18), 4259–4275.

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., Dowsland, K.A., Dowsland, W.B., 2001. The irregular cutting-stock problem—a new procedure for deriving the no-fit polygon. Computers and Operations Research 28 (3), 271–287.

Błażewicz, J., Moret-Salvador, A., Walkowiak, R., 2004. Parallel tabu search approaches for two-dimensional cutting. Parallel Processing Letters 14 (1), 23–32.

Błażewicz, J., Walkowiak, R., 1995. A local search approach for two-dimensional irregular cutting. OR Spektrum 17, 93–98.

Daniels, K., Milenkovic, V., 1996. Column-based strip packing using ordered and compliant containment. In: Selected papers from the Workshop on Applied Computational Geometry, Towards Geometric Engineering, Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 91–107.

Dowsland, K.A., Dowsland, W.B., 1995. Solution approaches to irregular nesting problems. European Journal of Operational Research 84 (3), 506–521.

Dowsland, K.A., Dowsland, W.B., Bennell, J.A., 1998. Jostling for position: Local improvement for irregular cutting patterns. Journal of the Operational Research Society 49 (6), 647–658.

Dyckhoff, H., Kruse, H.-J., Abel, D., Gal, T., 1988. Classification of real world trim loss problems. In: Fandel, G., Dyckhoff, H., Reese, J. (Eds.), Essays on Production Theory and Planning. Springer-Verlag, pp. 191–208, chapter 12.

Fujita, K., Akagi, S., Kirokawa, N., 1993. Hybrid approach for optimal nesting using a genetic algorithm and a local minimization algorithm. Proceedings of the 19th Annual ASME Design Automation Conference, Albuquerque, NM, 19–22 September 1993, vol. 1. ASME, New York, pp. 477–484.

Gomes, A.M., Oliveira, J.F., 1999. Nesting irregular shapes with simulated annealing. In: Proceedings of 3rd Metaheuristics International Conference, Angra dos Reis, Rio de Janeiro, Brazil, 19–23 July, pp. 235–239.

Gomes, A.M., Oliveira, J.F., 2001. A GRASP approach to nesting problems. In: Proceedings of 4th Metaheuristics International Conference, Porto, Portugal, July 16–20, pp. 47–52.

Gomes, A.M., Oliveira, J.F., 2002. A 2-exchange heuristic for nesting problems. European Journal of Operational Research 141 (2), 359–370.

Heckmann, R., Lengauer, T., 1995. A simulated annealing approach to the nesting problem in the textile manufacturing industry. In: Burkard, R.E., Hammer, P.L., Ibaraki, T., Queyranne, M. (Eds.), Annals of Operations Research, volume 57. J.C. Baltzer AG Science Publishers, pp. 103–133.

Hifi, M., M'Hallah, R., 2003. A hybrid algorithm for the two-dimensional layout problem: The cases of regular and irregular shapes. International Transactions in Operational Research 10 (3), 195–216.

Hopper, E., 2000. Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods. Ph.D. Thesis, Cardiff University.

Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Li, Z., Milenkovic, V.J., 1993. The complexity of the compaction problem. In: Lubiw, L., Urrutia, J. (Eds.), Proceedings of the Fifth Canadian Conference on Computational Geometry, University of Waterloo, Ont., Canada, 5–9 August 1993, pp. 7–11.

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.

Mahadevan, A., 1984. Optimization in Computer-Aided Pattern Packing. Ph.D. Thesis, North Carolina State University.

Metropolis, N., Rosenbluth, A., Rosenbluth, M.N., Teller, E., Teller, E., 1953. Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092.

Milenkovic, V.J., Daniels, K., 1999. Translational polygon containment and minimal enclosure using mathematical programming. International Transactions in Operational Research 6 (5), 525–554.

Oliveira, J.F., Gomes, A.M., Ferreira, J.S., 2000. TOPOS—A new constructive algorithm for nesting problems. OR Spektrum 22 (2), 263–284.

Oliveira, J.F.C., Ferreira, J.A.S., 1993. Algorithms for nesting problems. In: Vidal, R.V.V. (Ed.), Applied Simulated Annealing, Lecture Notes in Economics and Mathematical Systems, vol. 396. Springer-Verlag, Berlin, pp. 255–273.

Pirlot, M., 1996. General local search methods. European Journal of Opeartional Research 92 (3), 493–511.

Stoyan, Y.G., Novozhilova, M.V., Kartashov, A.V., 1996. Mathematical model and method of searching for a local extremum for the non-convex oriented polygons allocation problem. European Journal of Operational Research 92 (1), 193–210.