ELSEVIER

European Journal of Operational Research 141 (2002) 371–381

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

An algorithm for polygon placement using a bottom-left strategy

Kathryn A. Dowsland [a,*], Subodh Vaid [b], William B. Dowsland [a]

[a] *Gower Optimal Algorithms Ltd, 5 Whitestone Lane, Newton, Swansea SA3 4UH, UK* [1]
[b] *Radan Computational Ltd, Ensleigh House, Granville Road, Lansdown, Bath BA1 9BE, UK* [2]

Received 25 August 2000; accepted 17 May 2001

Abstract

This paper describes a fast and efficient implementation of a bottom-left (BL) placement algorithm for polygon packing. The algorithm allows pieces to be nested within the partial layout produced by previously placed pieces, and produces an optimal BL layout in the sense that the positions considered are guaranteed to contain the bottom-left position of the infinite set of possibilities. Full details of the way in which these positions are calculated are given. Computational experiments comparing the results of different orderings on a variety of datasets from the literature are reported, and these illustrate that problems having in excess of 100 pieces of several piece types can be solved within one minute on a modern desktop PC. The procedure can easily be incorporated into algorithms that apply more sophisticated piece selection procedures.

Keywords: Packing; Heuristics

1. Introduction

The problem of packing a given set of pieces into a sheet of fixed width in such a way as to minimise the length required occurs in a range of practical situations, including sheet-metal cutting and marker layout problems in the garment industry. A popular approach to solving such problems is to order the pieces and then place them in turn, choosing the leftmost feasible position, and breaking ties by selecting the lowest. This is known as a bottom-left (BL) placement policy. Early implementations usually involved one or more orderings based on the dimensions of the pieces, or a random sample of orderings from which the best solution was chosen. The advantages of this type of approach are its speed and simplicity, when compared with more sophisticated methods that may be able to produce solutions of higher quality. As a result there are still many commercial environments where such single pass placement policies are appropriate. Moreover interest in recent years has been boosted by implementations of modern

* Corresponding author.
E-mail address: k.a.dowsland@btconnect.com (K.A. Dowsland).
[1] (www.goweralg.co.uk).
[2] (www.radan.com).

0377-2217/02/$ - see front matter
PII: S0377-2217(02)00131-5

heuristics, such as tabu search or genetic algorithms, that use a bottom-left placement policy as the basis of cost/fitness evaluation.

Although there are a variety of slightly different interpretations of the bottom-left policy, these can be broadly partitioned into two classes. These are illustrated in Fig. 1, in which we assume that pieces 1–4 have already been placed and piece 5 is about to be placed using the relevant bottom-left definition. In the first class (Fig. 1(a)), pieces can only be placed to the right of the current packing front, in this case in-front of pieces 1, 3 or 4. The position shown is obviously the leftmost possibility within this region. Although this has the advantage of simplifying the calculations required, it will not allow smaller pieces later in the ordering to fill in gaps behind pieces already placed. The second class (Fig. 1(b)) remedies this by using a true leftmost placement policy, and allowing placements behind the packing front, a process sometimes referred to as hole-filling. When the pieces are rectangular the geometry in both cases is relatively simple. However, when irregular pieces are involved the calculation of the leftmost position for the next piece involves a complex geometric calculation, particularly if positions behind the packing front are to be considered. Although there are a number of published papers describing bottom-left algorithms for irregular pieces, many fail to provide details of the geometric calculations. Of those that do describe the geometry, most do not include hole-filling, or reduce the feasible positions to a finite set of points (often based on a grid). Others are restricted to very small problem instances or suggest prohibitive amounts of computational time for instances of moderate size. In this paper we present a bottom-left algorithm, complete with geometric details, that has proved to be both fast and effective on datasets of up to several hundred irregular pieces.

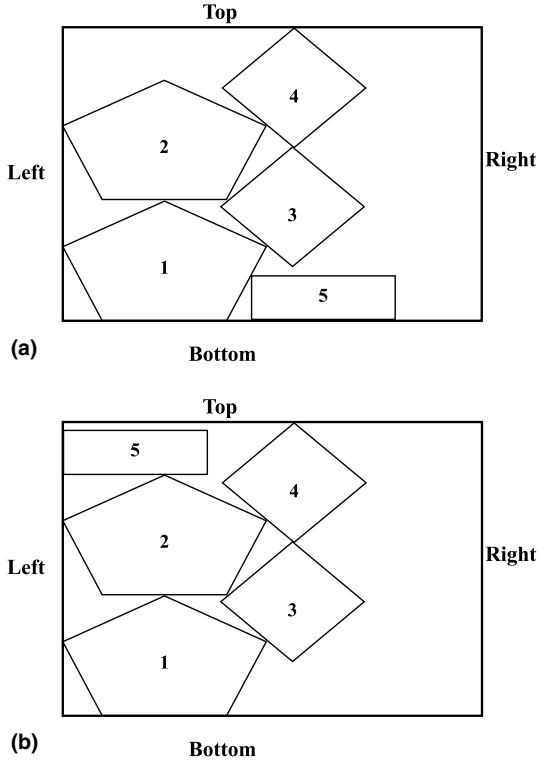

Fig. 1. Two classes of bottom-left placement policy: (a) without hole filling; (b) with hole filling.

The next section provides an overview of the problem and cites some of the published bottom-left algorithms for its solution. This is followed by an outline of the underlying geometric concept of our algorithm, the no-fit polygon. We describe the algorithmic framework, filling in the details in the following section, before going on to outline a series of modifications designed to reduce the computational effort required. Finally computational experiments, comparing different ordering rules on a range of different datasets, are presented.

2. The problem and BL solution approaches

2.1. The problem

The problem can be stated as follows.

Given: A stock sheet of infinite length and fixed width W, and a set of irregular pieces, $i = 1, \ldots, n$, represented as simple polygons (i.e. polygons without holes).

Objective: To pack all of the pieces onto the sheet without overlap, so as to minimise the length required.

We assume that rotation of the pieces is not allowed and that the n pieces constitute m piece

types or shapes, $k = 1, \ldots, m$, where there are b_k copies of type k, and $\sum_{k=1}^{m} b_k = n$. The vertices of each shape are defined by their co-ordinates relative to a 'reference point'. Here we assume that the reference point is defined by the bottom-left corner of the enclosing rectangle of the shape, as shown in Fig. 2. The position of a piece on the stock sheet can then be defined by the coordinates of the reference point.

2.2. Bottom-left heuristics

One of the simplest approaches to any two-dimensional cutting or packing problem is to define an ordering of the pieces and a placement policy used to allocate the pieces to the stock sheet in the given order. Here we consider a bottom-left placement policy which attempts to minimise the total length required by placing each piece as far to the left as possible, resolving ties by favouring positions nearer the bottom. The ideas behind this policy became very popular for rectangular packing problems during the 1980s (see for example Baker et al., 1980; Brown, 1980; and Coffman et al., 1984), and slightly different interpretations of the way in which the bottom-left position is defined/calculated continue to be published (for example Liu and Teng, 1999). In the case of irregular packing, the strategy used by human marker layout experts had a strong influence on early algorithms, and many were based on the formations of columns of pieces, which were then packed (see for example Art, 1966; Gurel, 1968). More recently leftmost placement policies have gained in popularity. Examples that do not allow hole-filling include the TOPOS algorithm of Oliveira et al. (2000), which successively adds new pieces to a partial solution, and the approach of Amaral et al. (1990), who partition the packing-front into a set of zones and then slide the piece to be packed into the leftmost zone. Examples allowing hole-filling include the work of Dowsland et al. (1998) who reduce the computational effort required by restricting placement positions to a grid, and that of Lamousin and Waggenspack (1997). The latter involves a comprehensive search of candidate positions, considering up to four different orientations for each piece. Calculations are based on the concept of the no-fit polygon outlined in the following section, but full computational details are not given, and computation times of several hours are quoted for problems of around 500 pieces.

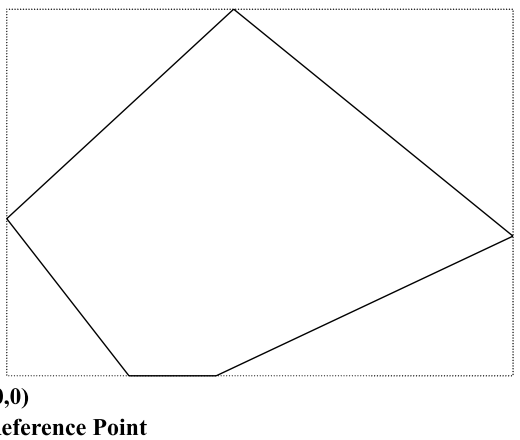

Fig. 2. Reference point for irregular piece.

Some of the above implementations consider pre-defined orderings of the pieces, while others select the next piece dynamically, so as to make good use of the space available. An alternative is to use a heuristic search technique to determine a good ordering of the pieces. Examples include Davis (1985) and Jacobs (1996) who use a genetic algorithm, Oliveira et al. (1996) who use tabu search, and the jostle approach of Dowsland et al. (1998). As such approaches need to perform a bottom-left packing at each iteration of the search, a fast implementation of the method is vital. One of the most effective tools for ensuring that the geometry is handled as efficiently is the no-fit polygon.

2.3. The no-fit-polygon

As is the case for many of the approaches above, our algorithm relies heavily on the concept of the no-fit polygon. Given any two shapes i and j the no-fit polygon of i and j, denoted NFP_{ij} defines the shape of the area in which placement of j will result in overlap with i. More formally NFP_{ij} is defined as the locus of the reference point of shape j, as it slides around shape i, when i is placed with

its reference point at the origin. The interior of NFP_{ij}, denoted $int(NFP_{ij})$, defines the positions where the shapes will overlap, and the boundary defines the set of positions where they will touch. The significance of this for any packing algorithm is that if piece i is already placed at $u = (x_i, y_i)$ and j is to be placed at $v = (x_j, y_j)$, then i and j will overlap if the point $v - u$ is inside NFP_{ij}, and will touch if it lies on the boundary. Fig. 3 illustrates this concept for two pieces A and B. As the pieces are not touching the point $P = u - v$ lies well outside NFP_{AB}. Point P can be moved a distance Δx in the x direction before it touches NFP_{AB}. This is reflected in the fact that piece B could be moved the same distance before touching piece A. A more detailed examination of the figure reveals that if point P is moved to a point inside the NFP, then translating piece B by the same distance and direction will result in overlap with A.

This can be used to advantage across the spectrum of irregular packing algorithms as it reduces the test for overlap between two pieces into a point inclusion test, which is computationally cheaper. In the case of a bottom-left policy it means that potential placement positions for the reference point of piece j can be reduced to the edges of the no-fit polygons NFP_{ij} translated by (x_i, y_i) for all previously placed pieces i.

When both shapes are convex the no-fit polygons are easy to calculate as described by Cuninghame-Green (1989). When one or more pieces have concavities the problem is more difficult, but there are a number of approaches available. These include decomposing the pieces into convex, or star-shaped, components as suggested by Li and Milenkovic (1995), calculating the NFP from first principles by executing a series of sliding operations as described in Mahadevan (1984) and utilised in Oliveira and Ferreira (1993), or using results based on the mathematical definition of the no-fit polygon as a Minkowski sum as suggested by Ghosh (1991) and implemented in Bennell et al. (2001). For the remainder of this paper we assume that the no-fit polygons of all pairs of shapes have been precalculated using some appropriate method.

3. The polygon placement algorithm

3.1. The basic algorithm

In this section we describe the framework that forms the basis of our algorithm. Shapes are packed onto the sheet starting from the left-hand edge of the sheet and moving towards the right-hand (open) end, such that each shape assumes a

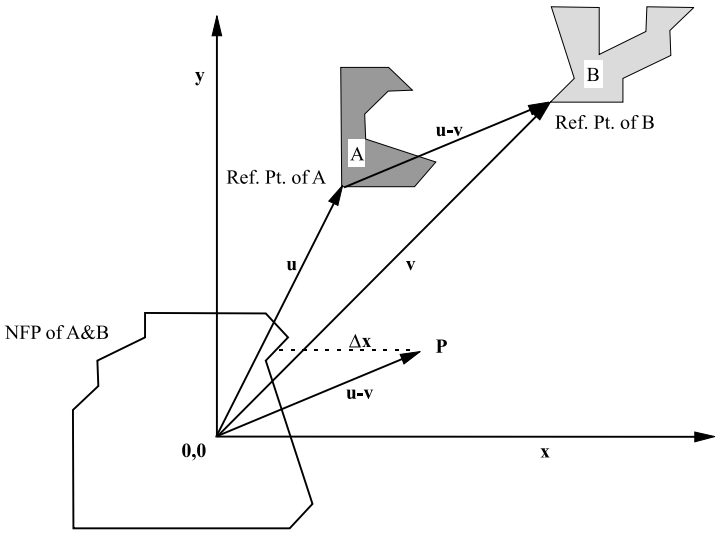

Fig. 3. Interpretation of the no-fit polygon.

bottom-left position. This process can be stated formally as follows:

> Given an ordering of pieces and a partial packing of pieces 1 to $(j-1)$, place piece j as far as to the left as possible, subject to the no overlap constraints. If there is more than one such position, then the one with minimum y co-ordinate is selected.

An obvious implication of the above is that piece j must either touch at least one of pieces 1 to $(j-1)$ or the left-hand edge of the stock sheet. For the moment, we ignore the latter possibility which will be dealt with later, and assume that piece j will touch one of the previously placed pieces. In the following we will refer to piece j as the *moving piece* and pieces 1 to $j-1$ as the *fixed pieces*. The set of feasible positions for a moving piece j with respect to a fixed piece i is found by inspecting the corresponding no-fit polygon NFP_{ij}. Let (x_i, y_i) and (x_j, y_j) be respectively the reference points for the fixed piece i and the moving piece j. The theory implies the following conditions:

> (1) piece j will touch piece i if the reference point (x_j, y_j) of piece j is placed on the boundary of $\mathrm{NFP}_{ij} + (x_i, y_i)$,
> (2) (x_j, y_j) is a feasible position for the reference point if (x_j, y_j) does not lie inside NFP_{kj} for any fixed shape k.

Thus we can define a procedure for placing a moving shape-j in a bottom-left policy as follows:

1. For each fixed piece i, let (x_i^*, y_i^*) be the leftmost point on the boundary of NFP_{ij} such that $(x_i^*, y_i^*) \notin \mathrm{int}(\mathrm{NFP}_{kj})$ for any fixed piece k, where $k = 1$ to $i-1$, $k \neq j$. If no such point exists, then set $(x_i^*, y_i^*) = (\infty, \infty)$.
2. Set the reference point of the moving piece j as $(x_j, y_j) = \{(x_{imin}^*, y_{imin}^*) : x_{imin}^* \leqslant x_i^* \ \forall i = 1, j-1\}$. In case of a tie in values of x_i^*, choose the one with minimum y_i^*.

Note as the partial layout is finite, and the stock sheet is assumed to have infinite length there must be at least one $x_i^* \neq \infty$.

The problem of finding the leftmost feasible point on any NFP_{ij} i.e. the (x_i^*, y_i^*)s, can be regarded as the problem of finding the leftmost feasible point on each of the edges of NFP_{ij}, and then finding the leftmost of these. As the required point will be the leftmost of all these leftmost points, we do not need to consider each NFP separately. Instead we can simply consider each edge in turn, and select the leftmost feasible point over all the edges. We can therefore modify the above procedure into one where the edges of the NFPs are checked in turn and the bottom leftmost point extracted. Let E be the set of edges from the set of no-fit polygons NFP_{ij}. Let (x_m^*, y_m^*) be the leftmost point on edge-m, such that $(x_m^*, y_m^*) \notin \mathrm{int}(\mathrm{NFP}_{kj})$ $\forall k = 1, \ldots, j-1$. The required bottom-leftmost position for shape j can be found by inspecting the values of (x_m^*, y_m^*) for all edges in E i.e. $(x_j, y_j) = \{(x_{mmin}^*, y_{mmin}^*) : x_{mmin}^* \leqslant x_m^* \ \forall m \in E\}$. As above in case of a tie in values of x_m^*, choose the one with minimum y_m^*.

The above procedure does not take account of the boundaries of the stock sheet. This requires two slight modifications to deal with pieces resting against the left-hand edge of the stock-sheet, and to ensure that positions in which pieces protrude beyond the sheet are eliminated.

3.2. Including the sheet boundaries

The above procedure assumes that the moving piece will touch at least one fixed piece. There is also the possibility that the moving piece will be placed against the left-hand edge of the sheet. This possibility can be included into the above procedure simply by adding this edge to the set E. It is also possible that placements on any of the NFP edges may result in placements that are not wholly contained within the stock sheet. As the reference point is defined as the bottom-left corner of the containing rectangle, any placements for which $x_j < 0$ or $y_j < 0$ will extend to the left of or below the sheet. Similarly, if w_j is the width of the enclosing rectangle of piece j, any placements for which $y_j > W - w_j$ will extend over the top of the sheet. This is easily avoided by clipping all edges with respect to the rectangle defined by $(0, 0, M, W - w_j)$ where M is

considerably larger than the maximum packing length required.

The procedure can now be stated as follows:

Step 1. Place the reference point of piece 1 at $(0,0)$. Set $j=2$.
Step 2. Set $(x_j,y_j)=(\infty,\infty)$.
Step 3. Let $E=\{e_1,e_2,\ldots,e_k\}$ be the list of NFP edges from NFP_{ij} for $i=1,\ldots,j-1$.
Step 4. Let e_{k+1} be the left edge corresponding to the left edge of the sheet. Let $E=E\cup e_{k+1}$.
Step 5. Clip the edges in E against the region $(0,0,M,W-w_i)$.
Step 6. For $m=1,\ldots,k+1$ find (x_m^*,y_m^*); set $(x_j,y_j)=\{(x_{m\min}^*,y_{m\min}^*):x_{m\min}^*\leqslant x_m^*\ \forall e_m\in E\}$. (In case of a tie in values of x_m^*, choose the one with minimum y_m^*.)
Step 7. Set $j=j+1$. If $j\leqslant n$ goto step 2 else stop.

Step 6 requires further detailed discussion as no procedure for finding the points (x_m^*,y_m^*) has been defined. This is discussed next.

3.3. Calculating the best feasible position on an NFP edge

When a NFP edge is being checked for placement of a moving shape the goal is to find the leftmost point that does not result in overlap. If we start by checking the end point having the smaller x co-ordinate (or in the case of a vertical edge that with the smaller y co-ordinate) and find a feasible position there, then the problem is solved. However, if the first point is not feasible due to overlap then it may be possible to identify a feasible position at an intermediate point on the edge. Before formalising the procedure for finding such a point we illustrate the basic idea with reference to Fig. 4.

Assume we are considering edge P_1P_2. The overlap caused by piece j when placed at the lower left endpoint P_1 of clipped left edge P_1P_2 is shown. The overlap of piece j with piece i is due to the fact that the reference point (x_j,y_j) of j lies inside NFP_{ij}. To avoid this overlap we need to place (x_j,y_j) on the boundary of NFP_{ij} by travelling along edge P_1P_2, starting from P_1 and moving towards P_2. Such a traversal leads to the intersection point of P_1P_2 with the NFP edge e_7. This intersection point can

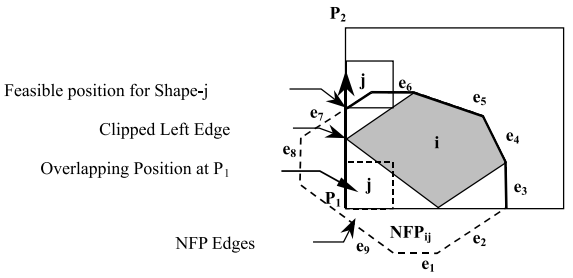

Fig. 4. Overcoming overlap on the clipped left edge P_1P_2.

be easily calculated from standard trigonometry. At this intersection point the overlap of piece i with piece j will be reduced to 0. If there are other pieces in the layout then piece j may now overlap with some other piece $k=1$ to $j-1$, where $k\neq i$. Thus if overlap is detected at the intersection of the current edge and some NFP, standard trigonometry is used to find the point where e_m leaves the offending NFP and this new point is then checked against all other NFP_{kj}. In this example if a new overlap had been found at the intersection point, then we would have kept travelling along P_1P_2 until a feasible position was found, or the other end of the edge (P_2) was reached.

In practice it is likely that a given point will lie inside more than one NFP. However, our procedure simply looks for the first offending NFP, and eliminates its overlap. If overlap with any of the other NFPs has not been eliminated this will be apparent when the new point is checked.

Thus the procedure for finding the leftmost feasible position (x_m^*,y_m^*) on an arbitrary edge e_m in E can be formally stated as

Step 0. Set $v=$ the first (i.e. leftmost) endpoint.
Step 1. Check v, using the point-in-polygon test for each NFP_{kj}, where $k=1,\ j-1$ until either an overlapping NFP_{kj} is found, or all NFPs have been considered. If no overlap is detected then a feasible bottom-left position for piece j has been found. Otherwise proceed to step 2.
Step 2. Use standard trigonometry to find all the intersection points between edge e_m and edges of NFP_{jk}. If no intersections exist there is no feasible position on edge m.

Otherwise let (x_p, y_p) be the closest such point to v. Set $v = (x_p, y_p)$ and return to step 1.

The above finalises the detail of the polygon packing algorithm. However, as it stands the algorithm is computationally demanding for all but small problem instances. The following section deals with ways of reducing computation time.

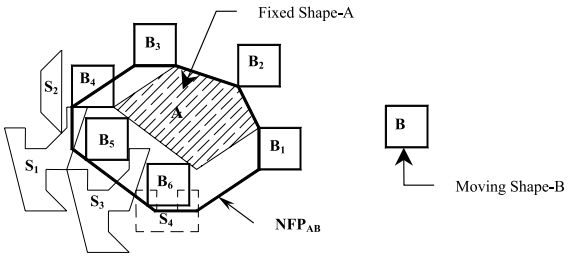

Fig. 5. Inspecting feasible positions on the edges of the NFP.

4. Improving algorithm efficiency

The set of feasible positions in the polygon packing algorithm is found by inspecting the edges of the NFPs of the moving shape and all fixed shapes already placed. It is obvious that the number of NFP edges increases as we move from the leftmost to the rightmost shape in the layout as there are more fixed shapes to consider. The number of such comparisons increases at rate that is quadratic in the number of pieces. This section considers three simple observations that can be used to reduce the number of comparisons required, and thus speed up solution time. Details of the improvements in execution time resulting from each of the modifications will be given in Section 5, and these illustrate that the implementation of all three together can typically reduce execution times by approximately 80%.

4.1. Differentiating between front and back edges

The complete set of feasible touching positions for a moving shape with respect to a fixed shape can be found by inspecting the edges of the corresponding NFP. However, these can be divided into two categories. One, when the moving shape is placed to the front of a fixed shape, and two, when it is placed to the back of a fixed shape. We illustrate this point with an example of a partial layout shown in Fig. 5. In this case let us assume that some of the possible touching positions for piece B are B_1 to B_6. It may be seen that out of these the non-overlapping positions are at B_1, B_2, B_3, B_4 and B_5. In position B_6 the placement of B on the edge of NFP_{AB} will cause overlap with the neighbouring shape S_4 (shown by dashed lines). Also it may be noted that if B was to be placed in positions B_1 to B_3, then it always touches front of A, while in positions B_4 to B_6 it touches the back of A. Of the two feasible positions at the back, if placed at B_4, B could be pushed backwards to touch S_2, i.e., B_4 cannot be a leftmost feasible position. At B_5, B cannot be pushed further back but it is possible to generate an equivalent or better position on the front edge of the NFP defined by S_1. This observation holds in the general case i.e., a position generated by a back edge of an NFP can either be improved, or will be generated on the front edge of another NFP. The logic also holds in the case of non-convex pieces and the resulting non-convex NFP's. In this case a concavity or hole within the NFP may have a combination of front and back edges. A front edge will result if a piece touching that edge could be moved further back within the concavity. Front and back edges are easily determined in all cases by the following definition.

Assume that the edges of the NFP are oriented in a counter-clockwise direction. A front edge is then defined as any edge for which the y coordinate is non-decreasing as one travels along the edge in the direction of orientation.

This motivates the following improvement. The boundary of each NFP can be considered to be made up of two sets of edges, the front and the back edges. The front edges correspond to active positions of the moving shape because in these positions the moving shape touches the front side of the fixed shape. It is sufficient to restrict the search to these edges because for all other edges the moving shape would be touching the back profile of the NFP and represent positions that are dominated by the front edges of other neighbouring shapes. In Fig. 6, we illustrate this with an

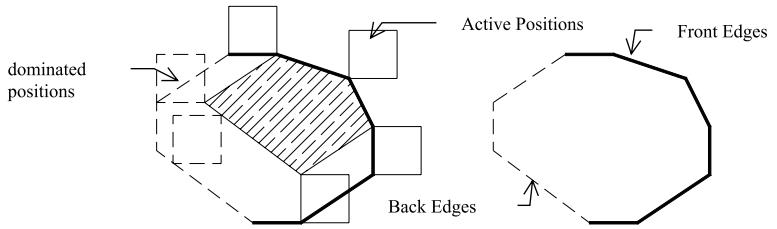

Fig. 6. Front and back edges of a NFP.

example of a NFP whose edges are classified as front and back. The front and back edges are respectively shown with solid and dashed lines. The active and dominated positions are also shown with solid lines and dashed lines respectively. A front edge can be formally defined as one that has the y co-ordinates of its end points in ascending order, when the NFP is traversed in anti-clockwise order. If an edge P_1P_2 has its two endpoints as $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, then P_1P_2 is a front edge if $y_2 \geqslant y_1$. From now on we assume that the set E consists only of front edges. This typically reduces the number of edges considered by approximately 50%.

4.2. Ordering of the edges

As the objective of the polygon packing algorithm is to find the bottom leftmost positions for the moving shape examination of each edge commences at its leftmost point. It therefore makes sense to store the edges in this way. Further efficiency gains can be made if the edges are sorted in increasing order of their x co-ordinates as follows. Let (x_m^*, y_m^*) be the leftmost feasible position for the moving shape found on edge-m in the sorted list. If the edges are sorted then an improved position is only possible until we reach an edge whose first end-point lies to the right of x_m^*. The remainder of edges in the list will have the x co-ordinates of their first endpoints equal to or greater than x_m^*. Thus the search for feasible positions can be terminated. This saves considerable computing effort in cases where small pieces are able to fit into holes well behind the packing front.

4.3. Defining a packing span for each piece type

Algorithm run time can further be reduced if some book keeping is done to record the minimum x co-ordinate positions at which shapes of a particular type are placed. When placing the qth shape of a particular type, we know that there is no need to consider positions to the left of the $(q-1)$th piece of that type, as they have already been considered and rejected. We define the region to the right of this as the packing span of the piece type. Using this information, all those front edges that lie outside the packing span can be ignored. For example in Fig. 7 when the diamond shaped piece D7 is considered it can lie no further left than the last placed shape of the same type, in this case D6. Thus the range of x co-ordinates that separates the leftmost x co-ordinate of D6 and XMax along the

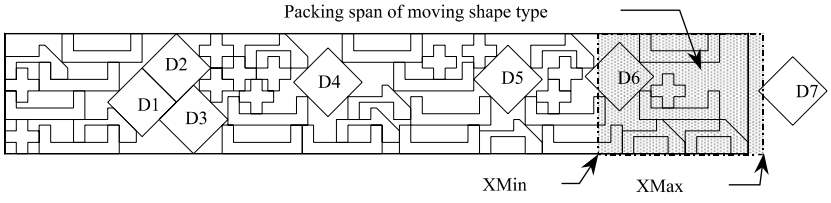

Fig. 7. Reducing search time by considering span of the packed shape type.

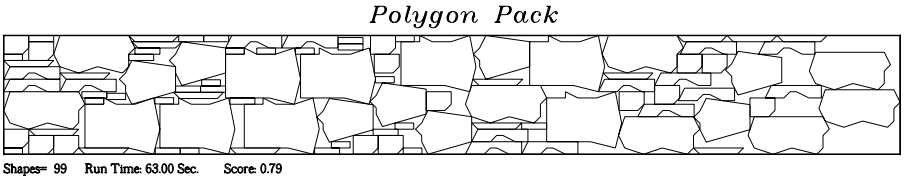

Fig. 8. A nested layout produced by the polygon packing algorithm.

sheet's bottom and top edges corresponds to the packing span for shape D7. This is shown as shaded in the figure.

All the above modifications have been incorporated into our polygon placement algorithm. A typical layout produced by the algorithm in just 68 seconds on a Pentium II-200 PC is shown in Fig. 8.

5. Computational experiments

The polygon packing algorithm has been used to compare the results of different static orderings on solution quality. The experiments are based on the four datasets used in Dowsland et al. (1998) together with the dataset used by Blazewicz and Walkowiak (1993). For each dataset three sheet widths were considered so as to gauge the effect (if any) of sheet aspect ratio on algorithm performance.

In each case 200 different problem instances were generated by randomly varying the quantity of pieces of each type, so that the total number of pieces lay in the range 20–120. For each dataset the algorithm was run using nine different orderings as given in Table 1. For each run the score was calculated as the ratio of the length required for a 100% utilisation over that recorded by the algorithm. For all but the random variant the score is averaged over the 1000 problems. In the case of the random variant 100 random orderings were tried for each of the 1000 problems.

In order to determine the improvements in computation time resulting from the three modifications described in the previous section, 50 of these experiments were selected at random. These were run four times starting with the basic algorithm and then adding the modifications in a cumulative fashion as follows:

BA – Basic algorithm.
OE – Basic algorithm with edges pre-sorted as described in Section 4.2.
PS – Basic algorithm with pre-sorting and packing span as described in Section 4.3.
FO – Basic algorithm using front edges only as described in Section 4.1, with pre-sorting and packing span.

The remaining problems were just run with the full version (i.e. FO).

The results on solution quality suggest that although the differences in mean performance are not large some orderings tend to give better solutions than others. This was backed up by a more detailed analysis in which the number of times each method out-performed each of the others was recorded. Both sets of analysis highlight PP-RU as a poor performer, and suggest that PP-PP and PP-L are the strongest performers, with PP-P and

Table 1
Different static orderings

Description	Code	Mean score (%)
Decreasing area of enclosing rectangle	PP-A	70.5
Decreasing length of enclosing rectangle	PP-L	71.3
Decreasing width of enclosing rectangle	PP-W	70.2
Decreasing perimeter of enclosing rectangle	PP-P	70.6
Decreasing aspect ratio of enclosing rectangle	PP-AR	69.2
Decreasing area of polygon	PP-PA	70.4
Decreasing perimeter of enclosing polygon	PP-PP	71.5
Decreasing utilisation ratio of enclosing rectangle	PP-RU	66.4
Random	PP-R	70.2

PP-A also doing well. One possible explanation for this is that all the high quality orderings encourage large pieces to be placed first, so that there is potential for smaller pieces later in the ordering to utilise any gaps in the layouts.

Those pieces for which the polygon perimeter is large also tend to be pieces with many concavities, thus increasing the potential for utilisable spaces within the layout. This view is reinforced by the poor performance of PP-RU, in which giving priority to pieces that fit well into their enclosing rectangles i.e. those without large concavities performed poorly.

The mean computation time for all variants using the final algorithm (FO) was under 30 seconds per run on a Pentium II-200 PC. The results of the 50 experiments used to measure the effectiveness of the three modifications intended to reduce computation times are shown in Table 2.

From this it can be seen that all the modifications are successful in reducing computation time. Although sorting the edges results in a significant reduction in mean computation time, there were significant variations between the various datasets in that datasets with mixed sizes of pieces showed the greatest improvement while one dataset in which all the pieces were of similar dimensions actually showed an increase in computation time. This is as expected as the search for a feasible position will only terminate early when a hole-filling position is discovered. Using the packing span (PS) resulted in the largest improvement and this was apparent across all datasets. The restriction to using front edges (FO) only reduced computation times by approximately 50% by those required by PS. Again this is as expected as the number of edges considered is reduced by approximately a half. Although computation times of up to 5 minutes as observed when using the basic algorithm might be acceptable for a single run of a single pass heuristic, the five fold improvement gained as a result of the modifications mean that the procedure is also suitable for use as a subroutine in more sophisticated packing heuristics.

Table 2
Mean computation times of different variants (seconds – PII-200)

Variant	BA	OE	PS	FO
Time (seconds)	131.54	118.06	42.14	26.44

6. Conclusions and suggestions for further research

This paper has described a fast and efficient implementation of a bottom-left placement policy for polygon packing that is capable of solving problems having around 100 pieces within one minute of computation time. An optimal bottom-left arrangement is produced (i.e. one in which each piece is placed in its leftmost, bottom-most position over the infinite set of feasible positions). Although the algorithm is stated for the case where pieces are not allowed to rotate, such a facility could easily be added by including the edges from the NFPs of the rotated pieces to the set M. The assumption that there are several copies of each piece type can also be relaxed. The basic algorithm is valid for any set of polygons, convex or concave, regardless of the different numbers of pieces or piece types. However, it is worth noting that in our experiments the greatest amount of speed up was obtained by the improvement limiting the search to the packing span for each piece type. In cases where there are fewer piece types and more pieces of each type the larger span will no doubt increase the computational burden.

The algorithm was used as a basis of computational experiments using a single pass placement policy and a static ordering. The results suggest that ordering rules based on the overall size of the piece, and the length of its perimeter are the most effective. However, it would be possible to use the placement algorithm in combination with more sophisticated dynamic selection rules, or as the basis of any of the improvement techniques cited in Section 2 of this paper.

Acknowledgements

Much of the work described in this paper was carried out whilst all the authors were staff members at University of Wales, Swansea.

References

Amaral, C., Bernardo, J., Jorge, J., 1990. Marker making using automatic placement of irregular shapes for the garment industry. Computers and Graphics 14 (1), 41–46.

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

Baker, B.S., Coffman Jr., E.G., Rivest, R.L., 1980. Orthogonal packings in two dimensions. SIAM Journal of Computing 9 (4), 846–855.

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 OR 28 (2001), 271–287.

Blazewicz, J., Walkowiak, R., 1993. An improved version of tabu search for irregular cutting problem. In: Karmann, A., Mosler, K., Schader, M., Uebe, G. (Eds.), Operations Research. Physica, Heidelberg, pp. 102–104.

Brown, D.J., 1980. An improved BL bound. Information Processing Letters 11 (1), 37–39.

Coffman Jr., E.G., Garey, M.R., Johnson, D.S., 1984. Approximation algorithms for bin-packing – An updated survey. In: Ausielo, G., Lucertini, N., Serafini, P. (Eds.), Algorithm Design for Computer System Design. Springer, Vienna, pp. 49–106.

Cuninghame-Green, R., 1989. Geometry, shoemaking and the milk tray problem. New Scientist 1677, 50–53.

Davis, L., 1985. Applying adaptive algorithms to epistatic domains. In: Michandani, P.B., Francis, R.L. (Eds.), Proceedings of the Ninth International Joint Conference on Artificial Intelligence, vol. 1. Morgan Kaufmann, Los Altos, CA, pp. 162–164.

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.

Gurel, O., 1968. Marker layout problem via graph theory: An attempt for optimal layout of irregular patterns. IBM Scientific Centre Report 320-2945, New York.

Ghosh, P.K., 1991. An algebra of polygons through the notion of negative shapes. CVGIP: Image Understanding 54 (1), 119–144.

Jacobs, S., 1996. On genetic algorithms for packing problems. European Journal of Operational Research 88 (1), 165–181.

Lamousin, H., Waggenspack Jr., W.N., 1997. Nesting of two-dimensional irregular parts using a shape reasoning heuristic. Computer-Aided Design 29 (3), 221–238.

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

Liu, D.Q., Teng, H.F., 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research 112 (2), 413–420.

Mahadevan, A., 1984. Optimisation in computer aided pattern packing. Ph.D. thesis, North Carolina State University.

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., Ferreira, J.S., 1993. Algorithms for nesting problems. In: Vidal, R.V.V. (Ed.), Applied Simulated Annealing. Lecture Notes in Economics and Maths Systems, vol. 396. Springer, Berlin, pp. 255–274.

Oliveira, J.F., Gomes, A.M., Ferreira, J.S., 1996. A new constructive algorithm for nesting problems. Conference Paper, IFORS 96, Vancouver, BC, Canada.