Journal of the Operational Research Society (1998) 49, 647–658 0160-5682/98 $12.00
http://www.stockton-press.co.uk/jor

Jostling for position: local improvement for irregular cutting patterns

KA Dowsland, WB Dowsland and JA Bennell
University of Wales, Swansea

This paper introduces a new improvement heuristic for irregular cutting and packing problems. The method is based on a small number of repetitions of any leftmost placement policy and is particularly effective in situations where computation time is strictly limited but exceeds that required for a single pass approach. Both the algorithm and the geometry required for implementation are described in full and the results of computational experiments on a variety of data are presented. These results show that the algorithm is an effective technique for producing good packings.

Keywords: heuristic; irregular cutting/packing

Introduction

The problem of cutting irregular shapes from a rectangular stock-sheet arises across a broad spectrum of industries. It has been tackled in a variety of ways and some of the most effective solution techniques combine problem specific rules with compaction techniques based on linear programming.[1,2] However, the placement rules often involve trying many different initial arrangements, and each arrangement may need several calls to the LP routines. Therefore this approach can be very time-consuming and tends to be confined to industries such as garment manufacture where a single pattern will be used many times. At the other end of the scale are single pass routines which combine a piece ordering with a placement policy.[3] These approaches are geared towards situations where patterns must be generated on-line and there is insufficient time for more sophisticated methods. This paper is concerned with situations between these two extremes, where the time available is strictly limited but there is scope to improve upon a single pass solution.

Our approach is simple in that it involves several applications of a single pass heuristic. As with all irregular cutting problems the efficiency of the algorithm is dependent on the handling of the geometry. The most efficient methods require a considerable overhead in assimilating and implementing the necessary mathematics, particularly when the pieces involved are not convex. We therefore present two versions of our algorithm, the first uses straightforward calculations based on standard trigonometry, while the second uses a concept known as the no-fit polygon to pre-process many of the geometric calculations.

Correspondence: Dr KA Dowsland, European Business Management School, University of Wales, Swansea, SA2 8PP, UK.
E-mail: k.a.dowsland@swansea.ac.uk

We begin by outlining our general approach which we have called the *jostle* approach. This is followed by a detailed description of the underlying single-pass heuristic, which is an improvement over more widely used policies in that it allows new pieces to be placed behind the current packing front. The simple geometric calculations needed to implement the method are detailed in full, starting with convex shapes and then extending the arguments to allow for concavities. This is followed by a section on the no-fit polygon. Here we outline the concept and quote computation times to illustrate the sort of efficiency gains that can be expected. We then go on to describe the computational experiments and results.

The jostle approach

The problem tackled in this paper is that of minimising the length required to pack a given set of irregular pieces, without rotation, onto a rectangular stock sheet of fixed width. Most simple single pass algorithms for this problem borrow ideas from two-dimensional bin-packing[4] and use an ordering of the pieces combined with a placement policy that places pieces towards the left of the stock-sheet according to a set of predetermined rules.[3,5,6] The resulting layouts frequently have a jagged right hand end as shown in Figure 1a, but intuitively less length will be required for a layout that has a flatter right hand profile as in Figure 1b. The jostle algorithm is inspired by this observation and is based on the fact that when granular products are stored in a container any unevenness in the surface can be removed by shaking the container up and down.

It starts with an ordering of the pieces (which may be arbitrary or defined by a set of heuristic rules) and a left-most placement policy. Once the first pass is complete and all the pieces have been placed they are re-ordered in decreasing order of the x-co-ordinates of their *right*-most

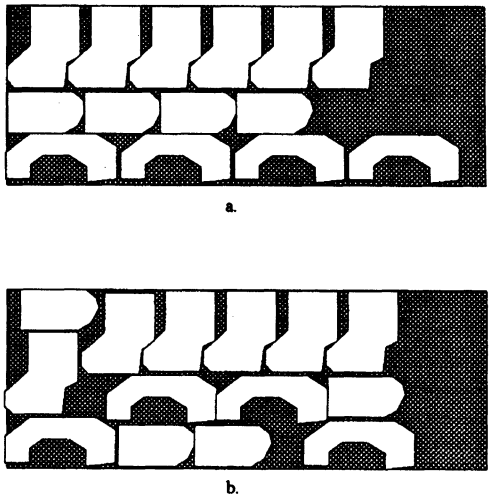


Figure 1 Layouts with jagged and flat left-hand sides.

points and packed according to a right-most placement policy. Once this packing is complete the pieces are re-ordered in increasing order of their left-most points and packed again using the left-most policy. The process is allowed to continue for a fixed number of iterations and can be applied to any initial ordering and left-most placement policy. Our implementation uses an initial random ordering and the placement policy defined below.

The best-known single pass placement policy is bottom-left, in which each piece is allowed to slide into position at the left-most feasible co-ordinate, with ties being broken by selecting the position nearest the bottom. This does not allow a new piece to 'jump over' pieces already placed, even if there is a big enough hole further back in the layout. Therefore this type of policy relies heavily on the sequence of placements to ensure that the smaller pieces are placed in the available gaps. For rectangle packing this may not matter too much as gaps left between pieces tend to be small. For irregular pieces, especially if they are not convex, it is vital to use gaps effectively in order to produce a dense packing. We therefore extend this type of policy to allow pieces to fill holes behind the current packing front. We also abandon the bias towards the bottom of the stock-sheet and give priority to positions closest to the top or bottom edges. This is achieved as follows.

For each piece the feasible positions are reduced to a lattice of positions, with x-co-ordinates at unit intervals from zero onwards and y-co-ordinates ranging from zero to W–w, where W is the width of the stock sheet and w the overall width of the piece. These are then ordered as shown in Figure 2. When a new piece is to be placed it is implicitly tested at each position in turn, until a feasible position with no overlap with previously placed pieces is found. However, the following two observations mean that only a subset of positions need to be tested explicitly.

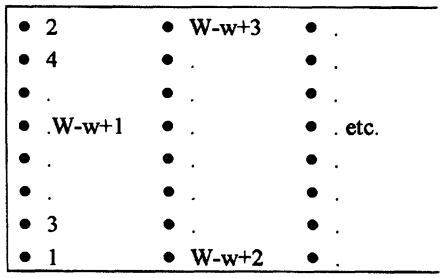


Figure 2 Ordering of placement positions.

Firstly consider the situation where we attempt to place piece A at a point with co-ordinates (x, y) and discover overlap with piece B. If no feasible position is found in the meantime piece A will eventually be tested at co-ordinates $(x+1, y), (x+2, y)$ etc. However, if the original test showed that the right-wards horizontal movement needed for piece A to clear piece B is at least z units then there is no point in considering positions $(x+\varepsilon, y)$ for any $\varepsilon < z$. Therefore for each y-co-ordinate we can store the next feasible x-co-ordinate and select from those y with the minimum such value according to their indices.

Secondly, when considering any other than the first piece of a given type we can eliminate many positions near the beginning. If the previous piece of type i was placed in position m, then positions 1 to $m-1$ are not available for a piece of type i and the search for a feasible placement for the next type i piece can start at position $m+1$.

In order to implement the leftmost placement policy we simply require a means of testing whether or not two pieces at given positions overlap. However, to take advantage of the first observation we also need to know the minimum horizontal movement required for piece A to lie entirely to the right of piece B, that is the quantity z defined above. As the corresponding rightmost policy is simply the mirror image of the leftmost policy the calculations can be adjusted as necessary. Alternatively the co-ordinate system can be changed to reflect this.

Implementing the algorithm using simple trigonometry

Convex pieces

If all the pieces are convex the geometry required to determine whether or not two pieces overlap and the distance that the new piece A needs to move to the right in order to avoid a fixed piece B is relatively straightforward. We define a piece, A, together with its position on the stock-sheet, as a set of vertices or corners $V(A)$ of known co-ordinates, joined by a set of edges or line segments $E(A)$. We will also use the notation $x_{A(\max)}, x_{A(\min)}, y_{A(\max)}, y_{A(\min)}$, to denote the maximum and minimum x- and y-co-ordinates of piece A. Two pieces overlap if there exists $v \in V(A)$ and e_1 and $e_2 \in E(B)$ or $v \in V(B)$ and e_1 and $e_2 \in E(A)$ such that

a horizontal line passing through v crosses e_1 to the right of v and e_2 to the left; or there exist $(v_1, v_2) \in E(A)$ and $e \in E(B)$ or $(v_1, v_2) \in E(B)$ and $e \in E(A)$ such that a horizontal line through v_1 crosses e to the right and a similar line through v_2 passes through e to the left. Konopasek[7] showed that the relationship between lines and points can be easily determined using the concept of D functions. The D function for a line (i, j) and a point p is given by: $D(i, j, p) = \text{sign}((x_i - x_j)(y_i - y_p) - (y_i - y_j)(x_i - x_p))$ and derives its name from the fact that it is the sign of the determinant of the matrix

$$\begin{pmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_p & y_p & 1 \end{pmatrix}.$$

Its value can be regarded as a measure of the difference in slope of the lines (i, j) and (i, p). If $D(i, j, p)$ is equal to sign $(y_i - y_j)$ then p lies to the right of the infinite line defined by segment (i, j) and if $D(i, j, p)$ and sign$(y_i - y_j)$ are opposite then p lies to the left. If $D(i, j, p) = 0$ then the point is on the line. Therefore we can use D functions to check for overlap by taking each vertex, v, from one polygon and checking its position relative to each edge (i, j) of the other. If for one polygon we order each edge (i, j) so that $y_i \geqslant y_j$ and let $F(v)$ be the set of vertices on the other polygon satisfying $y_i \geqslant y_v \geqslant y_j$ then a necessary and sufficient condition for overlap is that there exist v_1 and v_2 (possibly equal) and two edges $(q, r) \in F(v_l)$ and $(s, t) \in F(v_2)$ such that $D(v_1, q, r)$ and $D(v_2, s, t)$ are opposite.

The search for suitable vertices can be made more efficient by the following observations.

1. As A and B are convex then unless $y_v = y_i$ or $y_v = y_j$ there will be at most 2 edges in $F(v)$. Thus once these have been identified, either overlap is detected by opposite signs, or the search can move to the next vertex.
2. If $y_v = y_i$ then either $y_i = y_{A(\max)}$ or there is a corresponding edge (k, i) with $y_k \geqslant y_i$ such that we need only test one of these two edges. A corresponding argument applies when $y_v = y_j$.
3. If $y_v = y_{\max(A)}$ or $y_v = y_{\min(A)}$ then v is not inside A.

We can combine these facts to obtain a new checking procedure:

Check for overlap between 2 pieces

For each v such that $v \in V(A)$
  If $y_{\max(A)} > y_v > y_{\min(A)}$,
  find (q, r) and $(s, t) \in E(B)$ such that if $y_q > y_v \geqslant y_r$ and $y_s > y_v \geqslant y_t$
  if $D(q, r, v)$ and $D(s, t, v)$ are opposite then v is inside B and the polygons overlap.Stop
  if $D(q, r, v)$ or $D(s, t, v)$ is opposite to the D function for the previous vertex then edges cross and the polygons overlap. Stop.
Next v
  If overlap is not detected repeat with the roles of A and B reversed.

These checks must be carried out for piece A with all previously placed pieces. However, if

$$\max\{x_{\min(A)}, x_{\min(B)}\} \geqslant \min\{x_{\max(A)}, x_{\max(B)}\}$$

or

$$\max\{y_{\min(A)}, y_{\min(B)}\} \geqslant \min\{y_{\max(A)}, y_{\max(B)}\}$$

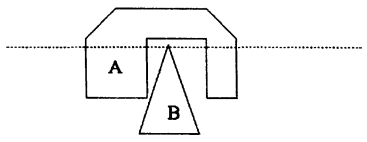
the pieces cannot overlap. Therefore this simpler test is used first and the full checking procedure is executed only if neither of these conditions are true.

Although the D function test detects overlap it gives no information as to the quantity z, however, a small modification enables us to rectify this. Standard trigonometry shows that $z = (x_i - x_j)(y_i - y_p)/(y_i - y_j) - (x_i - x_p)$ and that as long as $y_i > y_j$ then sign$(z) = D(i, j, p)$. Thus we can use sign(z) to detect overlap in place of the D function, thus calculating z as a by-product.

Dealing with concavities

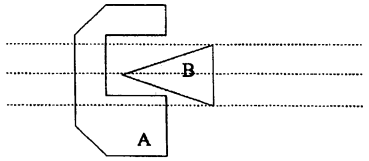
If any of the pieces have concavities the problem is not so straightforward. If piece A lies inside a concavity of piece B a horizontal line drawn through a vertex of B may pass through edges of A on either side, even though there is no overlap, as shown in Figure 3a. However, there will be no problem with concavities such as those displayed in Figure 3b.

In order to make this distinction we define two different types of convexity. The standard definition of a convex regions is that any line segment drawn between two points of the region does not pass outside. Here we define a piece



a.

A horizontal line through a vertex on B passes through E(A) on both sides



b.

All horizontal lines through vertices on B pass through E(A) on one side

Figure 3 x- and y-concavities.

to be x-convex if all *horizontal* lines between two points inside the piece stay within the piece. Any such line passing outside implies a concavity, and any concavity detected in this way will be referred to as an x-concavity. By definition if a piece is x-convex then the overlap detection routine outlined above will still work, with the exception of some horizontal edges that need no longer lie at the maximum or minimum y-co-ordinates. In these cases if $y_v = y_i = y_j$ then it is necessary to check whether or not v lies on edge (i, j). If it does then overlap is not detected. If it does not then the edge can be ignored. It is therefore only necessary to make major changes to the calculations for pieces that are not x-convex.

In order to do this we define a decomposition sequence that decomposes the polygon into a series of (possibly overlapping) pieces, each of which is x-convex. This is achieved by finding the deepest point, p, of an x-concavity and drawing a horizontal line to the left and right until a boundary edge is reached. If the point of contact of either line segment is a vertex then the piece is cut along this line, if not, the segment is pivoted about p until it reaches the next vertex, moving in an upwards direction for a concavity 'underneath' the piece and downwards for a concavity 'above'. The piece is then cut along this line. Doing this for both the left and right line segments creates two new pieces. If either still contains x-concavities the process is repeated using a binary search structure until all the pieces are x-convex. Figure 4 illustrates an example. The left-hand piece at level two is formed by taking the dotted line and pivoting it anti-clockwise until it hits the top right-hand corner. The piece is then cut along this line and left hand portion reserved. The right-hand piece is formed by the symmetric action of taking the dashed line and pivoting clockwise to the next vertex, cutting along the line and taking the right-hand portion etc. Note how the six shaded sub-pieces defined by the terminal nodes fit together, with some overlap, to re-form the original piece.

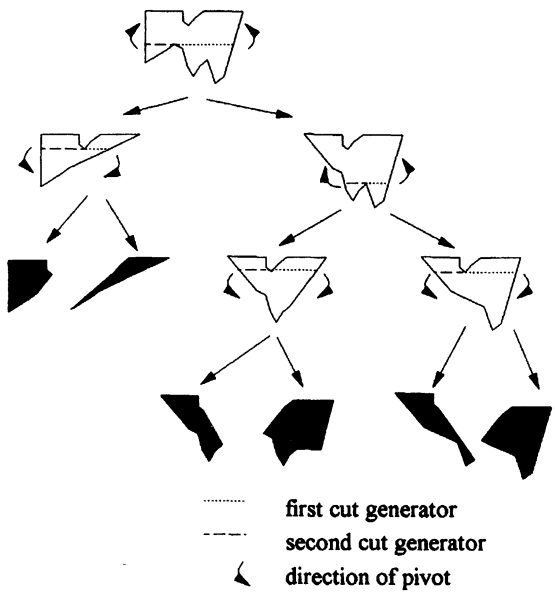


Figure 4 Tree of sub-pieces.

The x-concavities are recognised by searching the corners in order starting from the one with the minimum x-co-ordinate and moving clockwise. This corner is chosen as it cannot lie in an x-concavity. The point where the y-co-ordinates start decreasing is noted. If the y-co-ordinate turns again an x-concavity has been detected and the turning point is the deepest point from which the cut must be made.

Having performed the decomposition, overlap between pieces A and B can be detected if any of the sub-pieces formed from piece A overlap any of those formed from piece B. As the vertex sets of the sub-pieces are subsets of the original, and each branch in the tree adds just one new edge, the additional work in calculating the modified D functions is minimal. The process is therefore reasonably efficient.

The no-fit polygon

The above calculations have the advantage that they are simple to implement and rely on basic trigonometry. However the full set of calculations, requiring up to $2mn$ tests for two pieces with n and m corners respectively, must be executed each time a piece is considered for placement in a new position. A more efficient approach is to pre-process some of these calculations using the concept of the no-fit polygon. This was first introduced by Art,[5] but the name no-fit polygon was coined later by Adomowicz and Albano.[8] The interior of the no-fit polygon of two pieces A and B defines the region where, if A is assumed to lie at the origin, B cannot be placed without overlapping A. The boundary of the polygon represents the points where A and B would be touching and the area outside the polygon represents the positions where B can be placed without interfering with A in any way. If B is placed at point v, and A at u, the problem of detecting overlap reduces to that of determining whether or not the point $(v - u)$ is contained within the no-fit polygon—a procedure which is linear in the number of edges in the polygon. If A and B are convex the no-fit polygon will have $n + m$ edges and the complexity of overlap detection is reduced from $O(nm)$ to $O(m + n)$. Given the no-fit polygon it is then also a simple matter to calculate the required values for z. Details of such calculations can be found in any standard text on computational geometry.[9]

When all pieces are convex the no-fit polygons can be obtained by orienting the edges of one piece in a clockwise direction and the other in an anti-clockwise direction; ordering the edges according to their slopes; and then combining them to form a single polygon.[10] However, concavities pose a greater problem, particularly if several concavities in the two pieces interact. A number of

suggestions have been made to overcome this problem. Li and Milenkovic[11] decompose the polygons into pieces that are either convex or star shaped and then use a method designed to deal with these, while Mahadevan[12] goes back to first principles and calculates the effects of sliding a reference point on piece B around the circumference of piece A. For our experiments we modify a method based on slope diagrams due to Ghosh,[13] that has the advantage that non-interfering concavities do not need special treatment. Full details can be found in Bennell and Dowsland.[14]

In order to determine the efficiency gains that can be made by using the no-fit polygon the jostle algorithm was run on a variety of data sets covering both convex and non-convex pieces and with differing numbers of vertices. The results in Table 1 are based on the times required for an initial placement followed by 19 jostle iterations. They are presented in the form of ratios as the various data sets have been run on different processors rendering absolute times meaningless.

The three convex data sets have pieces with differing numbers of vertices (3–6, 6–10 and 10–15 for C1–C3 respectively). As expected the performance gains are most evident for the larger numbers of vertices, but even C1 shows significant improvement. The non-convex data sets are Data sets 1, 3 and 4 in Figure 5. Note that in spite of the fact that all data sets have pieces with large numbers of vertices the performance gains are relatively modest. Again this is to be expected as the number of vertices in the no-fit polygons will be in excess of $n+m$ if either piece has concavities. Nevertheless, computation times on all data sets are reduced significantly, suggesting that as long as the pre-processing is carried out efficiently the use of the no-fit polygon will result in improved computational efficiency.

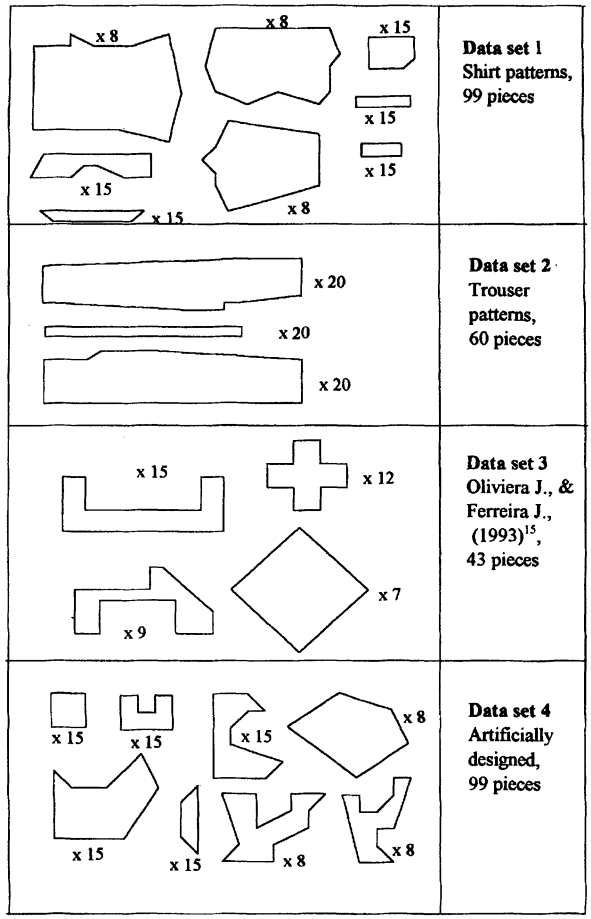


Figure 5 Data sets.

Computational experiments

Our computational experiments were designed to answer a number of questions concerning the jostle approach and the underlying 'hole filling' placement policy:

1. What degree of improvement does hole-filling give over a more standard sliding placement policy?
2. Does the jostle approach have any benefits over the same amount of work spent on a series of single pass placements using random orderings?
3. To what extent does the performance of jostle depend on the 'hole-filling' placement policy?
4. Does the performance depend on the data characteristics?

Table 1 Ratio of time taken (direct trigonometry/no-fit polygon)

	Convex			*Non-convex*		
Data set	C1	C2	C3	NC1	NC3	NC4
Ratio	1.71	3.63	4.90	1.24	1.82	1.56

Experiments were carried out on the four data sets illustrated in Figure 5. These are:

Data set 1. Data originally derived from real shirt patterns displaying a good mix of shapes and sizes

Data set 2. Real trouser pattern data consisting of two groups of similar long pieces

Data set 3. Data obtained from the literature consisting of a mix of geometric shapes with some nesting possibilities[15]

Data set 4. An artificial data set designed to combine a mix of shapes and sizes with possibilities for nesting.

These four data sets were packed onto stock sheets of three different widths; narrow, medium and wide. These were calculated by taking the total area of the enclosing rectangles of the entire piece set and then transforming this into a stock sheet of aspect ratio (namely length:width ratio), r, where $r = 6.825$ for narrow, $r = 3.675$ for medium and $r = 1.575$ for wide. These ratios were chosen for

historical reasons to match previous experiments on some of the data sets, and to cover a range of sheet dimensions. Each data set was packed into each sheet width using the following treatments

HL: 2000 random starts using a left-most placement policy with hole-filling.
HR: 2000 random starts using a right-most placement policy with hole-filling.
HJ: 100 random starts each jostled 19 times with hole-filling.
SL: 2000 random starts using a bottom left placement policy without hole-filling.
SR: 2000 random starts using a bottom right placement policy without hole-filling.
SJ: 100 random starts each jostled 19 times without hole-filling.

In order to compare random starts with the results of jostling, all four sets of random results were divided into 100 groups of 20 and analysis was based on items 2–20 in each set. Table 2 shows the mean lengths taken over all 1900 observations for each data set.

Not surprisingly the results clearly indicate the benefits that can be gleaned by incorporating hole filling, with the mean lengths achieved being significantly shorter across the full range of experiments. They also appear to indicate that for the H-treatments (that is those with hole-filling), that apart from on the wide stocksheet with Data set 2, jostle appears to give better results than the same number of random starts. In order to investigate this further the full distributions of length taken are shown in Figure 6. As the x-axis corresponds to length taken, the further the distribution lies to the left, the better the results. All but the plots for Data set 2 in Figure 6b show a clear advantage in using jostle. Even for this data set the minimum length achieved by jostle is several units lower than that achieved by the random orderings. It is also interesting to note the different shapes of the distributions, and the fact that they are dependent on the shapes of the pieces, rather than the width of the stock sheet. For Data set 1, not only is the jostle distribution further to the left in terms of position, it is also distinctly skewed in this direction. For Data set 2 the jostle distribution tends to be rather flat with a long tail while the random orderings display a few tall peaks, again followed by a long tail. Data sets 3 and 4 display similar behaviour to each other, although Data set 4 produces a smaller set of different lengths. Finally, there is no significant difference between the results from the right and left-most random placement policies.

Although the above analysis shows that jostle with hole-filling appears to produce good results it does not reflect the way the approach would be used in practice. It is likely that there will only be sufficient time to jostle a few random starts. Therefore it is worth comparing the best result from each set of 19 jostles with the best from each group of random starts. Figure 7 shows the cumulative frequency distributions for the best length reached over the 100 random starts for HJ and the 100 separate groups for HL and HR. All the graphs indicate the superiority of the jostle approach. This is true, even for Data set 2, although it should be noted that the three plots tend to converge earlier on this data set, particularly for the medium width stock sheet.

As jostle is less impressive on Data set 2 it is worth considering the differences between this and the other data sets. Firstly the pieces can be partitioned into two groups of similar types, while the other sets all have at least four significantly different shapes and sizes. Secondly all the pieces are around the same length, and thirdly they do not have many deep concavities. In order to investigate these ideas further two more data sets were produced. Our rationale was to transfer the lack of concavities and the similar lengths onto the basic features of Data set 1 where jostle performed very well. Data set 5 is made up of the enclosing rectangles of the pieces in Data set 1, that is rectangles of dimension $x_{\max(A)}$ by $y_{\max(A)}$ for each piece A, and Data set 6 consists of the pieces in Data set 1 stretched in the y direction so that they are all of the same length as the longest piece.

The results are shown in Figures 8 and 9. On Data set 5, the rectangular data set, the distributions are very similar to those of the original Data set 1. This suggests that the

Table 2 Comparison of mean performance over 1900 observations

	Narrow				*Medium*				*Wide*			
	1	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
HJ	141.2	211.0	135.1	98.1	104.2	158.8	100.2	73.1	77.9	118.7	73.4	54.1
HL	145.8	211.9	137.9	100.1	107.4	159.0	102.8	74.9	81.3	117.2	76.2	55.8
HR	145.4	212.7	137.5	100.4	107.3	159.2	102.5	74.8	81.3	117.3	76.1	55.8
SJ	164.8	223.0	157.1	112.8	119.2	168.9	115.3	83.7	88.6	125.1	85.0	62.1
SL	159.6	220.1	157.4	112.1	115.3	164.6	115.3	82.9	85.7	120.5	85.0	60.9
SR	159.2	220.3	157.0	111.8	115.1	163.8	114.9	82.7	85.4	120.8	85.1	60.2

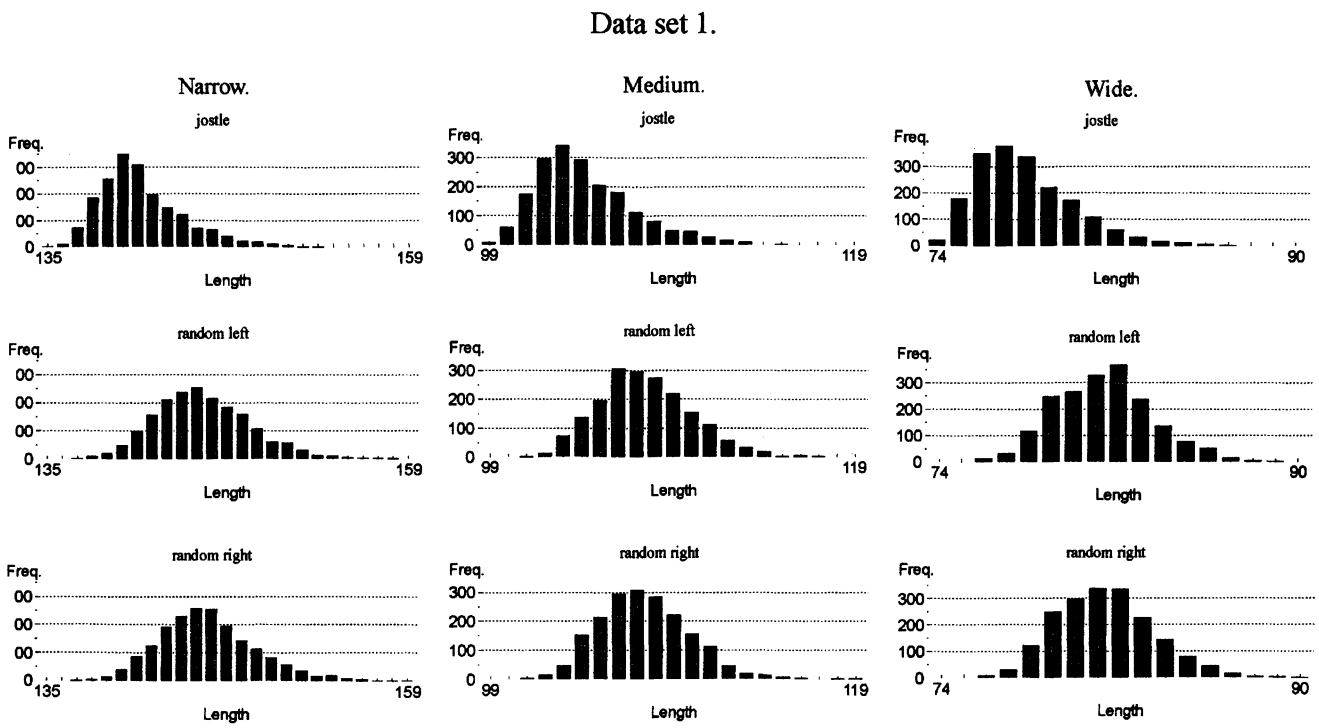


Figure 6a Distribution of lengths for HJ, HL and HR on data set 1.

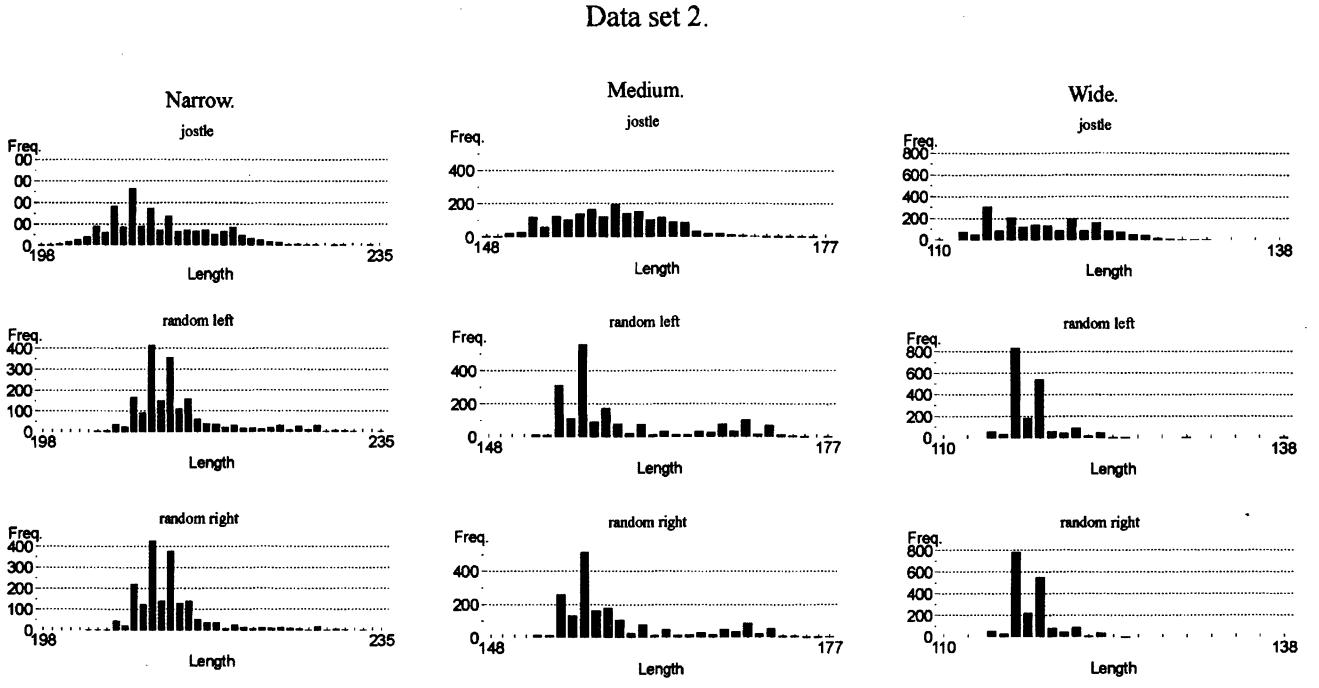


Figure 6b Distribution of lengths for HJ, HL and HR on data set 2.

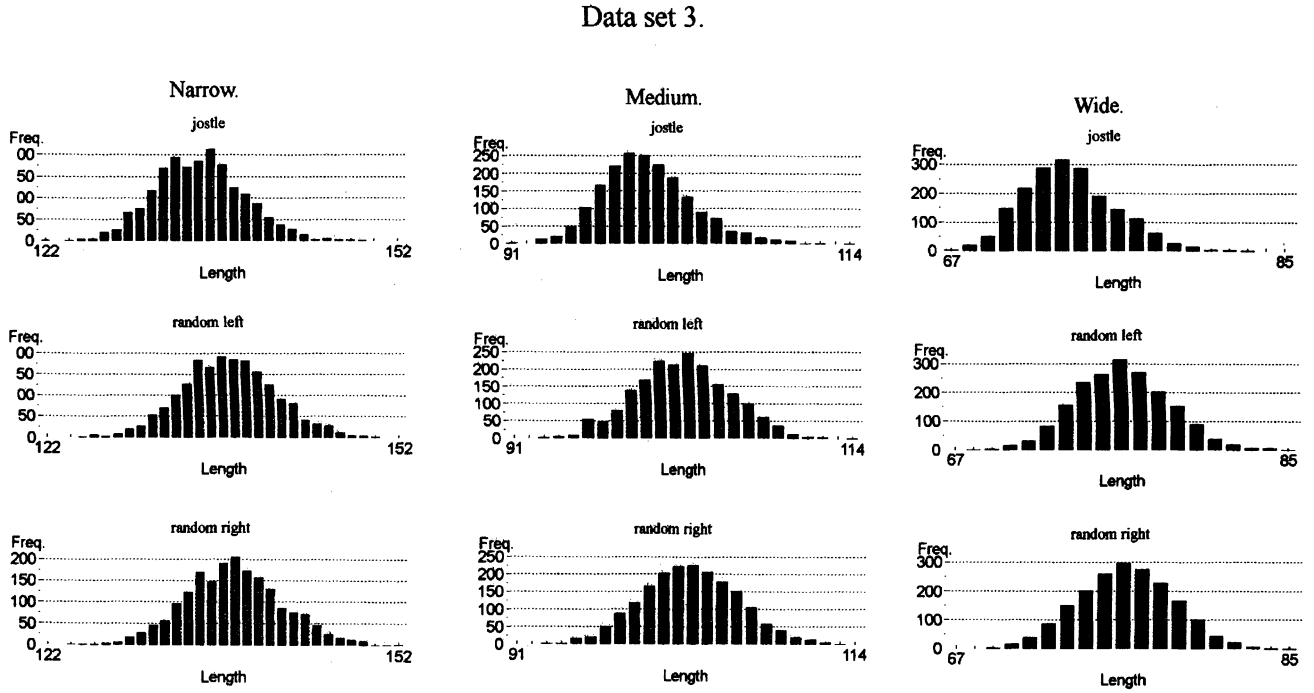


Figure 6c Distribution of lengths for HJ, HL and HR on data set 3.

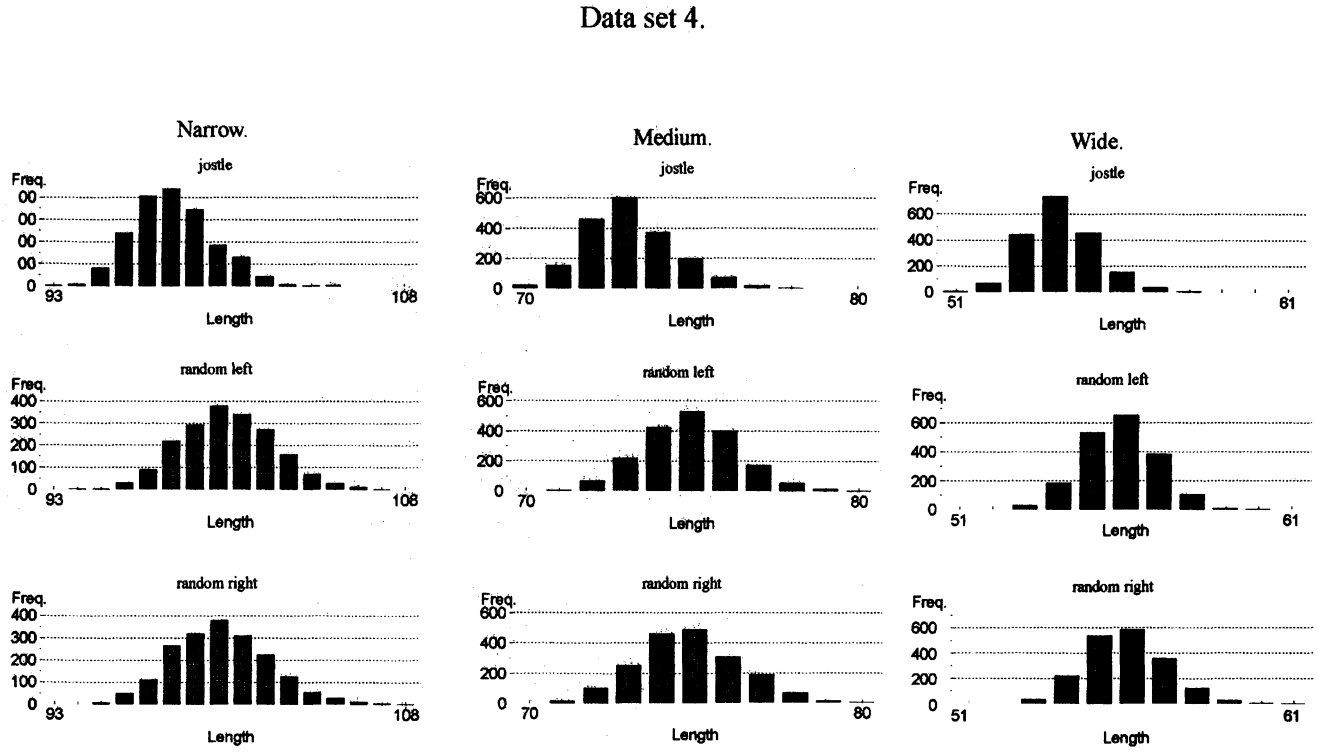


Figure 6d Distribution of lengths for HJ, HL and HR on data set 4.

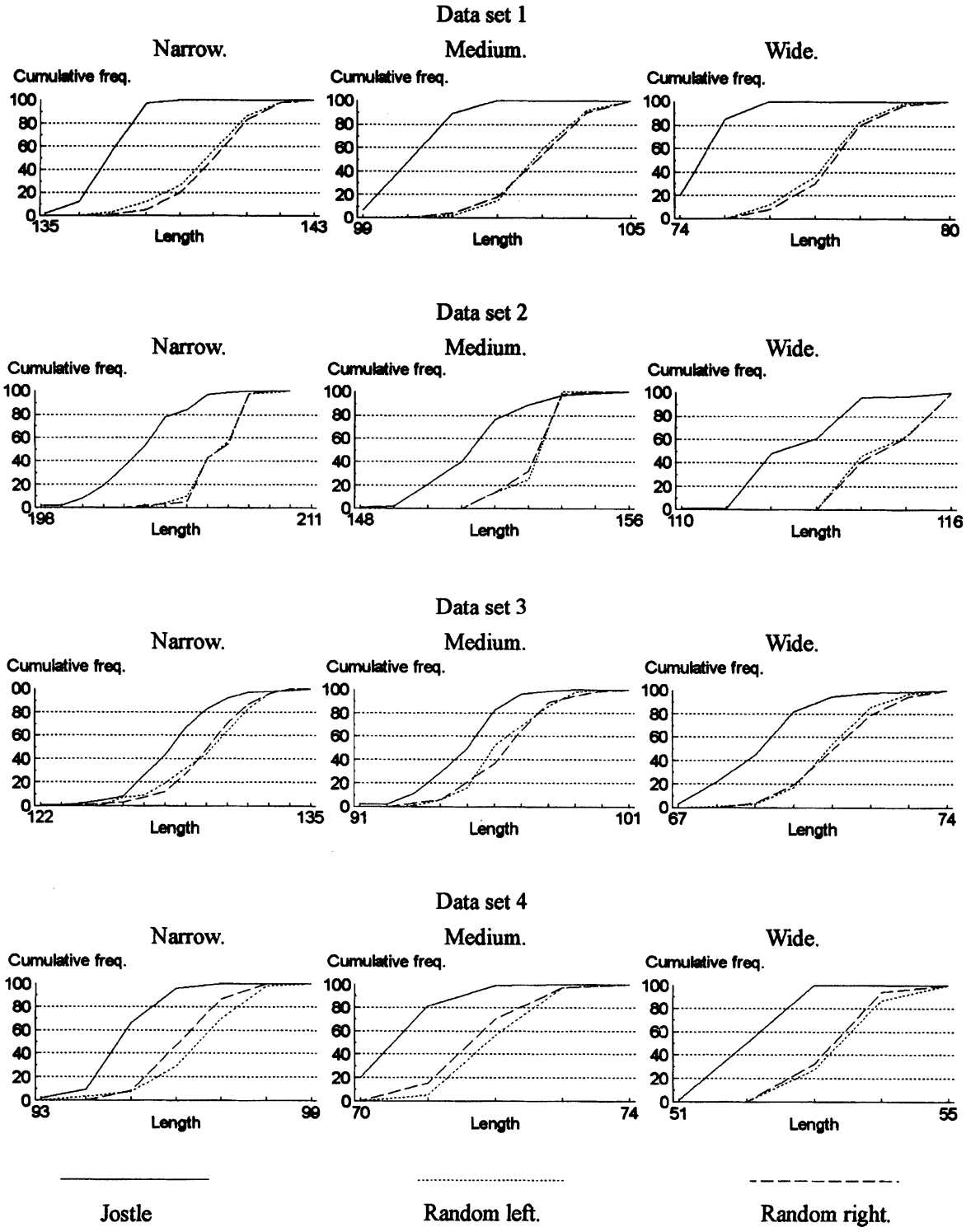


Figure 7 Cumulative frequencies over 100 sets of 19 jostles or random orderings.

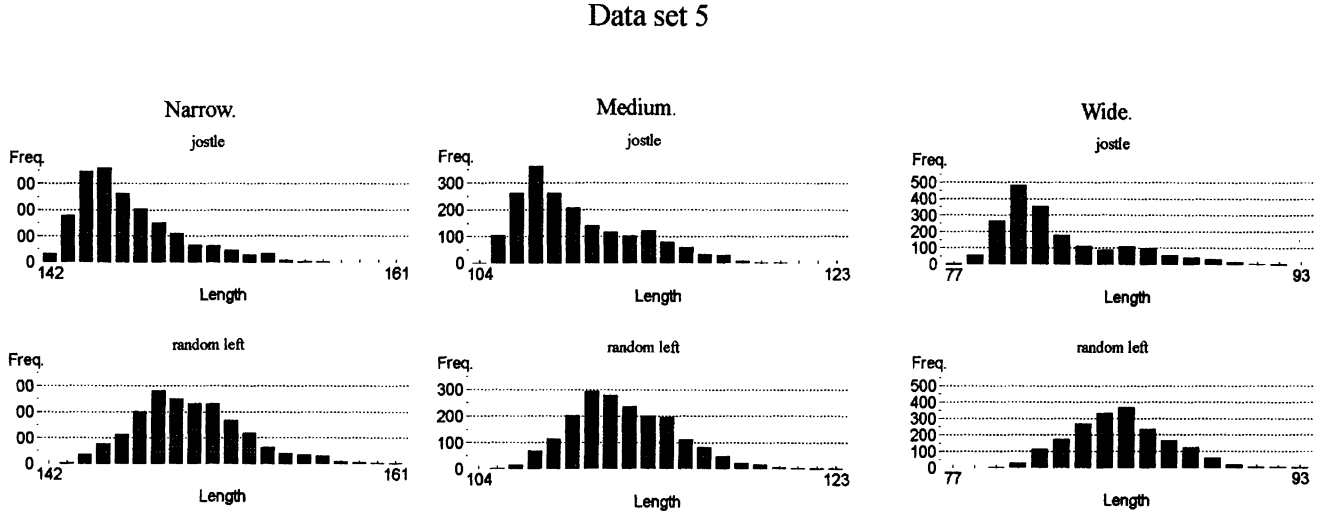


Figure 8a Distribution of lengths for HJ, HL and HR on data set 5.

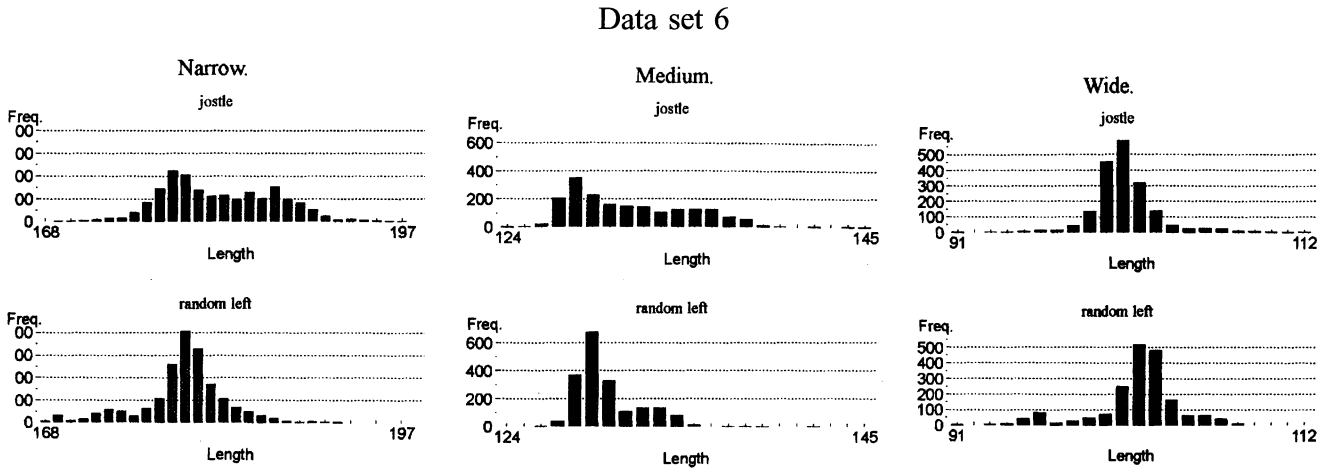


Figure 8b Distribution of lengths for HJ, HL and HR on data set 6.

overall size and approximate shape of the pieces has more effect than the precise details. These results also show that, although designed for irregular packing problems, jostle can be very effective for rectangle packing. The results for Data set 6 are more interesting in that jostle does not now perform as well and for the narrow and wide stock sheets the cumulative performance plot suggests that jostling is a bad idea. Intuitively this result is as expected as pieces of the same length have less chance of hole-filling, and there will be little change in the ordering of the pieces from one iteration to the next—especially on the narrow stock sheet. However, this does not explain the difference observed for the medium width stock sheet. Observation of the algorithm as it placed each piece in turn suggested that the width may be critical here, in that the large pieces tend to fit across this width in such a way as to leave gaps for the narrower pieces. Therefore there is more opportunity for hole-filling than was apparent with the other two widths.

One final observation of the data is that Data sets 3 and 4 have more concavities and opportunity for nesting than Data set 1. Given that the jostle distribution takes on a different shape for this data set we performed one more experiment to see whether or not there was any evidence that the skewed distribution was more likely in convex data. For this experiment we produced Data set 7 by replacing the pieces in Data set 3 by their enclosing rectangles. The results, not duplicated here, were inconclusive in that there was some evidence of skewness, but his was not as pronounced as for Data set 1.

Overall the results show that spending time jostling from a few random starts is likely to yield better solutions than packing the same number of random orderings, particularly if the piece dimensions include a variety of widths. Indeed

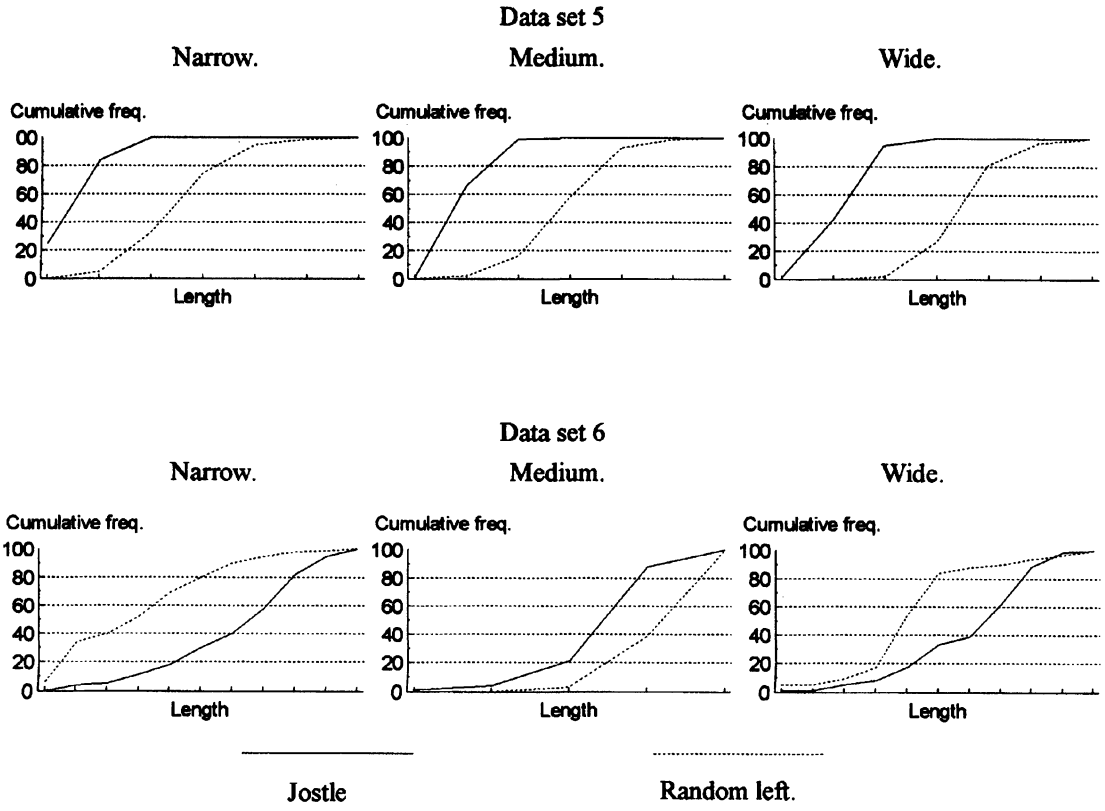


Figure 9 Cumulative frequency distributions for Data sets 5 and 6.

one of our earliest experiments carried out in 1993,[16] produced a solution for Data set 3 that has not been matched by any other method to date.[17] This layout is reproduced in Figure 10. Typical computing times for this data set are around 30 seconds for 19 jostles on a Pentium 166 pc. Given that the computational complexity of checking each piece given a fixed number of grid positions for a problem with n pieces is $O(n_2)$, and that the number of positions will increase with the number of pieces, computation times do increase significantly with problem size. For example the corresponding times for the 99 pieces in Data set 4 is around 180 seconds.

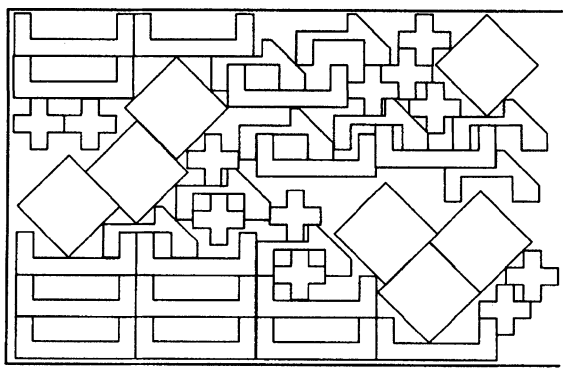


Figure 10 A jostle produced best known result for Data set 3.

It should also be noted that jostle is only effective with a hole-filling placement policy, which will be more computationally expensive than a method that simply builds forwards from the packing front. Efficiencies can be achieved by implementing effective strategies based on the no-fit polygon for dealing with the computational geometry. However the overheads, in terms of development effort, for achieving this can be considerable and the savings gained depend on the complexity of the pieces and the accuracy with which they are represented.

Conclusions

This paper has introduced a new improvement heuristic for irregular cutting and packing problems. The improvement phase is easy to implement in that it simply requires a sorting facility to be added to any left-most placement policy that allows pieces to be placed in holes behind the current packing front. The implementation of the initial packing algorithm is also relatively straight-forward and uses modified D-functions based on traditional trigonometry. However, significant savings in computation time can be made by adopting an approach based on the no-fit polygon. Preliminary results indicate that although the behaviour of the method depends on the characteristics of

the data, it is very effective for a variety of different data characteristics, with the exception of problems where all the pieces have the same length.

The work reported here is just the first stage in a fuller investigation. Further analysis is needed to ascertain why the method works, and to identify more clearly the types of data for which it is likely to be particularly effective. We hope to report these results in a later publication. However, the evidence presented here shows that the jostle approach is certainly worth considering in environments where the computation time available is more than that required for a single pass placement policy, but is insufficient for one of the more sophisticated approaches.

References

1 Milenkovic V and Daniels K (1992). Placement and compaction of non convex polygons for clothing manufacture. In: Wang CA (ed) *Proceedings of the Fourth Canadian Conference on Computational Geometry*. Memorial University of Newfoundland, St. John's, Newfoundland, August 1992, pp 236–243.

2 Stoyan YG, Novozhilova MC and Kartashov AV (1993). Mathematical model and method of searching for a local extremum for the non-convex oriented polygons allocation problem. Working paper—Institute for Problems in Machinery, Ukranian Academy of Sciences.

3 Amaral C, Bernardo J and Jorge J (1990). Marker making using automatic placement of irregular shapes for the garment industry. *Comp Graph* **14**: 41–46.

4 Coffman EG, Garey MR and Johnson DS (1984). Approximation algorithms for bin packing—an updated survey. In: Ausiello G *et al (eds)*. Algorithm Design for Computer Systems Design. Springer Verlag: Vienna, pp 49–106.

5 Art RC (1996). An approach to the two-dimensional cutting stock problem. IBM Cambridge Scientific Center Report 36-Y08.

6 Böhme D and Graham A (1979). Practical experiences with semi-automatic and automatic partnesting methods. In: Kuo C *et al* (eds). *Computer Applications in the Automation of Shipyard Operation and Ship Design III*. North Holland: Amsterdam, pp 213–218.

7 Konopasek M (1981). Mathematical treatments of some apparel marking and cutting problems. US Department of Commerce Report 99-26-90857-10.

8 Adamowicz M and Albano A (1976). Nesting two dimensional shapes in rectangular modules. *Comp Aided Design* **8**: 27–33.

9 Preparata FP and Shamos MI (1985). *Computational Geometry—An Introduction*. Springer Verlag: Berlin.

10 Cunninghame-Green R (1992). Cut out waste!. *O.R. Insight* **5**(3): 4–7.

11 Li Z and Milenkovic V (1995). Compaction and separation algorithms for non-convex polygons and their applications. *Eur J Opl Res* **84**: 539–561.

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

13 Ghosh PK (1991). An algebra of polygons through the notion of negative shapes. *CVGIP: Image Understanding* **54**: 119–144.

14 Bennell JA and Dowsland KA (1997). Packing polygons: dealing with the geometry. Working paper EMBS/1997/03, European Business Management School, University of Wales Swansea.

15 Oliveira JFC and Ferreira JAS (1993). Algorithms for nesting problems. In: Vidal RVV (ed). *Applied Simulated Annealing, Lecture Notes in Economic and Mathemaical Systems* **396**. Springer Verlag: Berlin, pp 255–273.

16 Dowsland KA and Dowsland W (1993). Heuristic approaches to irregular cutting problems. Working Paper EBMS/1993/13, European Business Management School, University of Wales Swansea.

17 Oliveira JFC and Ferreira JAS (1997). Private Communication.

Received October 1997;
accepted January 1998 after one revision