PERGAMON

Computers & Operations Research 28 (2001) 271–287

computers & operations research

www.elsevier.com/locate/dsw

The irregular cutting-stock problem — a new procedure for deriving the no-fit polygon

Julia A. Bennell[a], Kathryn A. Dowsland[b,*], William B. Dowsland[b]

[a]*Department of Management, University of Southampton, UK*
[b]*OR Group, European Business Management School, University of Wales Swansea, Singleton Park, Swansea, Wales SA2 8PP, UK*

Received 1 August 1998; received in revised form 1 January 2000

Abstract

The nofit polygon is a powerful and effective tool for handling the geometry required for a range of solution approaches to two-dimensional irregular cutting-stock problems. However, unless all the pieces are convex, it is widely perceived as being difficult to implement, and its use has therefore been somewhat limited. The primary purpose of this paper is to correct this misconception by introducing a new method of calculating the nofit polygon. Although it is based on previous approaches which use the mathematical concept of Minkowski sums, this new method can be stated as a set of simple rules that can be implemented without an indepth understanding of the underlying mathematics. The result is an approach that is both very general and easy to use.

Scope and purpose

Cutting and packing problems involving irregular shapes are common in industries ranging from clothing and footware to engineering and shipbuilding. An important feature of any automated solution technique for such problems is the way in which the geometric calculations are handled. One of the most efficient approaches is to pre-process many of the calculations using a concept known as the nofit polygon (NFP). However, use of the NFP has been limited due to the difficulty in its calculation for arbitrary irregular pieces. This paper outlines the concept of the NFP in the context of the irregular cutting-stock problem and introduces a new simplified approach to its calculation. The aim is to make this powerful concept more easily accessible to those involved in the cutting and packing of irregular shapes.

Keywords: Irregular cutting stock problem; Nesting; Geometric algorithm

* Corresponding author. Tel.: + 44-1792-295601; fax: + 44-1792-295626.
E-mail address: k.a.dowsland@swan.ac.uk (K.A. Dowsland).

0305-0548/00/$ - see front matter
PII: S0305-0548(00)00021-6

1. Introduction

This paper is concerned with the geometric calculations needed to solve cutting and packing problems involving irregular shapes. Such problems occur whenever a manufacturing process requires irregular blanks or pieces to be cut from a sheet of stock material. Examples include dye-cutting in the engineering sector, parts nesting for shipbuilding, marker layout in the garment industry, and leather cutting for shoes, furniture and other goods. Even when all the components are rectangular the problem of finding layouts that minimise waste is known to be NP-hard. Where irregular components are involved an extra dimension of complexity is generated by the geometry. As sufficient accuracy can usually be obtained by representing the pieces as polygons we restrict our treatment to the packing of polygons. However, the methods employed can be modified to deal with curved edges as long as the tangents can be calculated at all points.

The precise requirements of a good layout will differ from industry to industry and this has lead to a variety of algorithmic approaches, ranging from single pass placement policies to sophisticated compaction algorithms based on mathematical programming techniques. A recent survey can be found in Dowsland and Dowsland [7]. In spite of their differences, all the methods have a common requirement in which they need to be able to answer questions concerning the geometric interaction of the pieces such as: *do these two pieces overlap*? *or how far can piece A be moved until it touches piece B*? While it is possible to base these calculations on standard trigonometry, this tends to be inefficient, and considerable gains in execution speed can be obtained by pre-processing some of the geometry using a concept known as the nofit polygon.

When all the pieces involved are convex the nofit polygon is very simple to calculate. If non-convex pieces are involved the calculations become more complex. Although implementations that make use of the NFP have increased since the wider dissemination of the work of Li and Milenkovic [13] via SICUP [18], these are still relatively scarce. Given its flexibility and obvious benefits this is somewhat surprising, but may be in part due to the perceived difficulty of calculation. This view is reinforced by a tendency for much of the relevant literature either to omit the necessary detail or to approach the problem from a theoretical point-of-view. Those who have made use of the NFP tend to use one of two approaches, both of which have disadvantages when dealing with complex pieces. In Ghosh [9] Ghosh describes an approach based on a form of vector addition that overcomes these problems. Although the theory holds good for any simple polygons (i.e. polygons without holes) it is difficult to implement when both pieces contain, possibly nested, concavities.

The primary purpose of this paper is to introduce a new way of calculating the nofit polygon. The method is based on Ghosh's approach and inherits all its positive features, but has the additional advantage that it can be stated as a set of relatively simple rules that do not get more difficult to implement as the pieces get more complex. Given the scarcity of publications dealing with the nofit polygon outside of the mathematics and computer science literature, and the apparent reluctance of some researchers to make full use of its properties, we have included sufficient detail in the earlier sections to act as a short tutorial on the subject. The next section deals with the nofit polygon and its significance to the computational requirements of efficient cutting and packing algorithms. We then go on to describe the two frequently cited calculation methods and point out their positive features and disadvantages. This is followed by an introduction to Ghosh's method, together with reference to the mathematics that underpins it. Finally, we show

how the method can be adapted to provide a new approach to finding nofit polygons, that is both relatively simple and computationally efficient.

2. The no-fit polygon: what it is and how to use it

The basic idea behind the nofit polygon (NFP) was originally proposed by Art [2] in his paper on the marker layout problem. He used the term envelope and it was not until a later paper by Adamowicz and Albano [1] that the term nofit polygon was introduced. Not all researchers have adopted this terminology and it is worth noting that the term hodograph is sometimes used to denote the NFP in the mathematical literature; the term configuration space obstacle or CSO is frequently found in the robotics and engineering literature for the equivalent of the NFP in two or more dimensions; and dilation, a term arising from the discipline of mathematical morphology, is commonly used in the field of computer vision. The NFP of two polygons A and B, denoted as NFP_{AB} is the polygon that results from a sliding operation in which A and B have specific roles. The first polygon in the subscript is defined as the fixed polygon whose origin is assumed to be at point (0, 0). The second polygon is called the tracing polygon and is moved to perform the sliding operation. A and B will retain these roles for the remainder of the paper unless otherwise stated. Given a physical representation of the two objects, NFP_{AB} can be obtained by placing B in a touching position with A and marking the locus of a reference point on B as it traces around the boundary of A. The tracing motion of B is performed in such a way that B does not rotate and A and B always touch, but never overlap. The locus of the reference point forms a closed path that is NFP_{AB}. Fig. 1 illustrates this tracing movement. If the roles are reversed then the resulting nofit polygon, NFP_{BA}, is NFP_{AB} rotated by 180°. The relevant property of NFP_{AB} with respect to the interaction between A and B is as follows: if A is placed at (0, 0) and B is positioned with its reference point inside NFP_{AB} then A and B intersect; and if B is positioned with its reference point on the boundary of NFP_{AB} then A and B touch. Thus, the interior of NFP_{AB} represents all intersecting positions of A and B and the boundary represents all touching positions.

When dealing with cutting or packing problems we obviously need to be able to place A at an arbitrary point. However, our calculations need not be based on the absolute positions of A and B as it is their position relative to each other that is of concern. We can therefore obtain equivalent information for the case when A is positioned at point (x_A, y_A) by transposing both pieces by $(-x_A, -y_A)$. This will position A at (0, 0) as required. If the reference point on B is at position (x_B, y_B) then A and B overlap if and only if the point $(x_B - x_A, y_B - y_A)$ lies inside NFP_{AB}. Thus if polygon A is placed at point u and B is placed at point v, the problem of determining whether A and B intersect is reduced to the point inclusion test of determining whether or not the point defined by $v - u$ lies inside NFP_{AB}. Algorithms for such a test can be found in any standard text on computational geometry.

Several researchers [15,19] have pointed out the connection between the nofit polygon and a form of vector addition known as the Minkowski sum. If A and B are two arbitrary sets of points in n-dimensional space then the Minkowski sum of A and B is given by

$$A \oplus B = \{a + b\colon a \in A, b \in B\}.$$

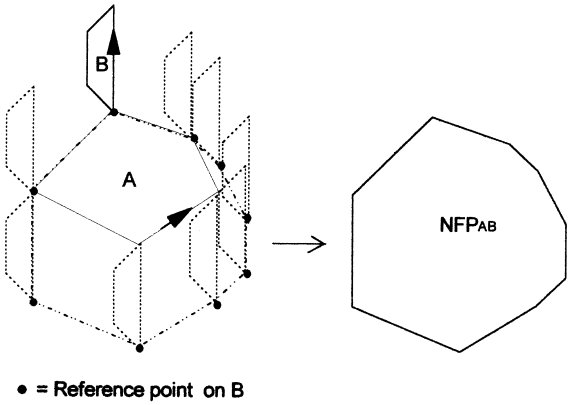

Fig. 1. The locus of the reference point on B maps out the nofit polygon as B traces around A.

Simple vector algebra can be used to show that $A \oplus -B$, known as the Minkowski difference of A and B is equivalent to NFP_{AB}. Since we follow the convention that polygons have counter-clockwise orientation then $-B$ is simply B with clockwise orientation.

Properties of the NFP can be used to supply the geometric information required by many different solution approaches to the family of irregular packing problems. Constructive approaches require information about the point where a new piece touches one of the existing pieces in the layout. The boundaries of the nofit polygons contain all the information that is necessary for such calculations. Many local search approaches require an efficient test for overlap [11,20]. Knowledge of the NFPs reduces this to a simple point inclusion test. Yet another class of packing algorithms [13,15] rely on compaction techniques based on linear programming, in which the constraints limit the relative movements between two pieces so that they do not overlap. Here the formulation of the linear programs is made significantly easier when the NFPs are known as their edges correspond directly to the constraints.

3. Calculating NFPs

When both shapes are convex the NFP is simple to calculate and the following algorithm can be found in a variety of sources (for example Cuninghame-Green [5]). Assume that the nofit polygon represents the motion of a reference point on B moving around A in a counter-clockwise direction. Its edges will be copies of the edges of A oriented in a counter-clockwise direction, and copies of the edges of B oriented in a clockwise direction. This gives an intuitive explanation of the algorithm. It starts by orientating A counter-clockwise and B clockwise, as shown in Fig. 2a. All the edges are then placed with their origin at (0, 0) retaining their original direction and magnitude, as shown in Fig. 2b. Then, starting at an arbitrary point, the vectors are ordered according to their counter-clockwise order in the diagram and are joined end to end resulting in Fig. 2c. Note that the edges from each individual polygon maintain their ordering within the diagram. This is a property of the convexity of the polygons, and ensures that each edge of the NFP is either the next edge encountered on A or the next edge on B as the sliding operation takes place.

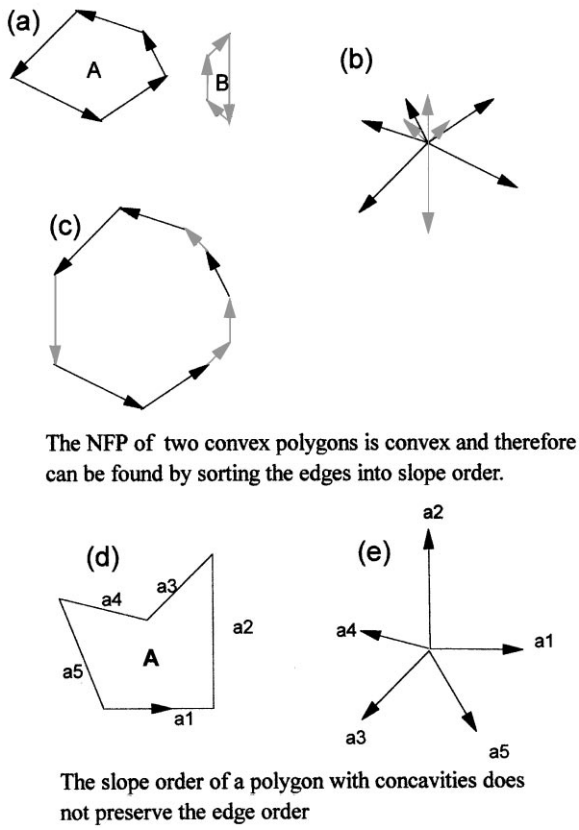

Fig. 2.

When one or both pieces are non-convex the method will not work as the edges within a single polygon will not be ordered in sequence after sorting, as shown in Figs. 2d and e. Yet the sliding analogy shows that the edges within a given polygon must be visited in the correct order. Thus a different approach must be taken. When dealing with non-convex shapes the following two approaches are commonly used.

4. Approaches for non-convex pieces

Perhaps the most obvious way of dealing with non-convex shapes is to decompose them into a set of convex sub-pieces. Piece B overlaps piece A if a sub-piece of B overlaps a sub-piece of A. Thus, the union of the NFPs of the sub-pieces placed in the correct positions will be NFP_{AB}. This has the advantage that the sub-NFPs can be found quickly and easily using the algorithm for convex pieces. However, there are a number of drawbacks. First, the decomposition usually involves a computationally efficient heuristic. This may lead to a larger number of sub-pieces than necessary, adding to the computational burden of calculating all the sub-NFPs. Second, when two sub-NFPs touch but do not overlap the touching edges lie on the boundary of the final NFP and care is needed to ensure that they are recorded appropriately. The number of sub-pieces involved

may be reduced by decomposing into other classes of shapes for which the NFP calculations can be simplified. For example Li and Milenkovic [13] base their decomposition approach on star-shaped[1] pieces.

An alternative to decomposition is to approach the problem from first principles and to calculate the NFP as the locus of the reference point as B slides around the boundary of piece A. Mahadevin [14] presents an algorithm for this. The decision as to whether the next edge of the NFP should be derived from A or B is made by comparing their slopes using a concept known as a D-function or determinant function [12]. However, due to the concavities it may not be possible to slide along the whole edge, and standard trigonometry must be employed to determine the point at which the edge should be clipped. Although such calculations are simple to implement they effect the efficiency of the method. A more serious drawback of this approach is that situations where piece B can be feasibly placed inside A's concavity, but is too wide to slide in from the outside, will not be detected. However, as long as the problem is recognised it can be dealt with in a separate phase. For example Oliviera et al. [16] have incorporated such a phase into their NFP calculation routine.

The above problems can be avoided using an elegant method for the calculation of Minkowski sums as suggested by Ghosh [9]. As described in the previous section the NFP and Minkowski difference are equivalent, thus any method for calculating Minkowski sums can also be applied in the calculation of NFPs. Ghosh shows that unless the two pieces have concavities that interfere with one another, in the sense that there may be some degree of interlocking in the sliding process, then a simple extension to the algorithm for convex pieces can be applied. Where this is not the case he outlines a modification. However, implementation details are only given in outline and lead to a cumbersome tangle of intersecting edges for complex pieces. It is therefore not surprising that the simpler methods outlined above tend to be used by those implementing a nofit polygon routine. Here we introduce a modification to Ghosh's method that overcomes these implementation difficulties. First we describe the underlying method in more detail.

5. Ghosh's Minkowski sum approach

Ghosh's method is based on the result that the Minkowski sum of any two polygons can be obtained as a function of their boundary edges. In the convex case this function leads to the same algorithm as that of Cuninghame-Green, in which the function simply requires that the edges be sorted into slope order. However, as will be illustrated, for the non-convex case some edges will appear several times in both their positive and negative orientations. The underlying mathematics of the algorithm is an extension of a result for convex shapes in n-dimensions due to Grunbaum [10]. It uses the concept of supporting lines to the points on the boundaries of the pieces. These are defined as lines passing through the point that, in the vicinity of the point, do not fall inside the piece. Vertices have an infinite set of supporting lines while edges have just one. Further details can be found in Bennell [3], Ghosh [8,9].

The first stage in Ghosh's approach is to extract the relevant information by summarising the polygon diagrammatically in what he calls a slope diagram. The edges are represented as points on the circumference of a circle, so that the slope of a line from the centre of the circle to the point is

[1] 1 A star-shaped polygon is one for which there is at least one internal point such that any line drawn from the point to the boundary stays wholly within the piece.

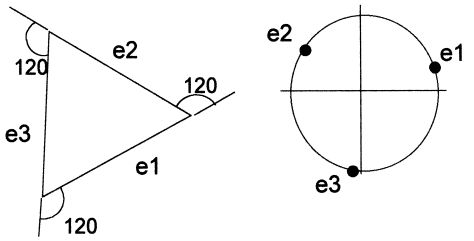

Fig. 3. An equilateral triangle and its slope diagram where points represent edges and arcs represent the turn of the vertex.

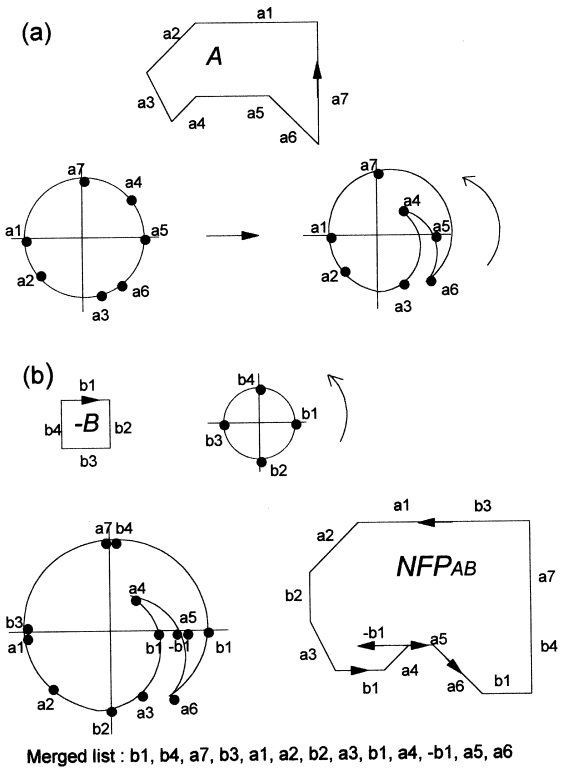

Fig. 4. Minkowski difference of a convex and non-convex polygon using boundary addition theorem.

equal to the slope of the edge in question. Such a single point represents the tangent of the supporting line of the edge. These edge points are then labelled according to their counter clockwise ordering in the polygon as in Fig. 3. As the edges are represented by points, the arcs between consecutively labelled points represent vertices, and the angle subtended by such an arc is simply the range of tangents of the set of supporting lines that represent that vertex. Thus, the slope diagram for this figure consists of three points separating three arcs each turning through 120°. In order to traverse the slope diagram of a non-convex polygon so that the edges are visited in the correct order it is necessary to make several passes over some regions of the diagram, changing direction between each pass. This is illustrated in Fig. 4a, where to maintain the ordering of a3 to a4

it is necessary to pass a5 and a6. To move on to a5 it is necessary to turn at a4 and move in a clockwise direction as far as a6 when another turn is necessary in order to reach a7. As all the angles between successive edge-points must be less than 180° there is never any ambiguity as to the correct direction of travel between edge-points. Note also that the clockwise traversal represents the concavity.

Ghosh shows that the Minkowski sum of two polygons can be obtained by merging their slope diagrams. (*Where edge points from different polygons occur in the same place we will adopt the convention of letting those from B precede those from A in the subsequent ordered list.*) For the convex case this leads to exactly the same process as that given in the previous section. In the case where one polygon, say A, is non-convex the algorithm is still applicable as long as the slope diagram is traversed so that the edges of A are visited in the correct order, and those edges of B are included every time they are passed in the traversal. A counter-clockwise pass is interpreted as adding the edge in its given direction, and a clockwise pass as traversing the edge in the opposite direction. Thus for the example in Fig. 4b edge b1 is traversed 3 times, twice in the positive and once in the negative direction. The Minkowski difference $A \oplus -B$ is given by the sequence of edges listed in the figure. We will refer to such a list as the edge list representation of the NFP. Note that at this stage the NFP may be represented as a self-crossing polygon such as the one in the figure. We assume for now that this is a suitable representation and address the problem of simplifying such a polygon in a later section.

The above idea can be extended to the case where neither polygon is convex, as long as the concavities do not interfere with one another so that in any area of the slope diagram only one set of edges are encountered in the wrong order. If, however, two concavities interact then there will be areas of the slope diagram where both sets of edges are wrongly ordered. This can be illustrated by considering polygons A and B in Fig. 5. Now B's concavity will interact with that of A, as is apparent from the slope diagrams in which traversal of the concavities lie in the same area. If we try to traverse the merged slope diagram starting at a1 we start with a1, b2, a2. At this point keeping A in order would require us to go to a3, thus passing b4 before b3, while keeping B in order would imply going to b3 and thus passing a4 before a3. The solution lies in the fact that the traversal of the slope diagram in this area must be carried out in two parallel paths, one obeying the order of A and one the order of B. When obeying the order of A then any mis-ordered parts of B (here b3/b4) are represented as a single point on the diagram located between the appropriate out of sequence points from A (here b3/b4 located at point b3 on the original diagram).

Thus we have for A:

a1, b2, a2, ($b3$, $b4$), a3, $-$($b3$, $b4$), a4, ($b3$, $b4$), b5, a5

and for B:

a1, b2, a2, ($a3$, $a4$), b3, $-$($a3$, $a4$), b4, ($a3$, $a4$), b5, a5.

The nofit polygon is interpreted as taking the outer face of the two paths, or equivalently the union of the two nofit polygons obtained by using first one path and then the other. Although the theory of traversal by parallel paths holds true for more complex cases there are considerable implementation problems in sorting out the paths. Consider for example Fig. 6. Here it is difficult

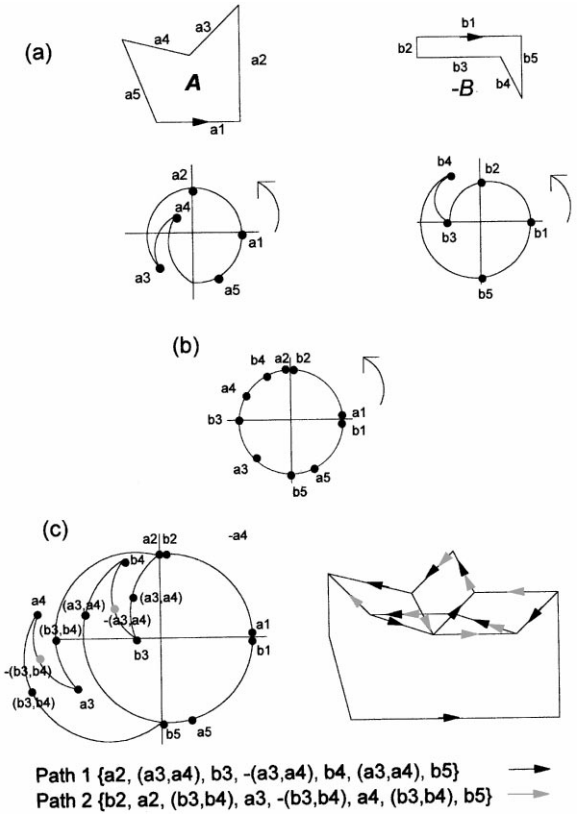

Fig. 5. Minkowski difference using parallel paths when concavities interfere with one another.

to determine which edge-points are interfering with which. This led us to seek an approach that would be easier to implement.

6. A new approach to finding the nofit polygon

Any algorithm based on the slope diagram requires a means of untangling the conflicting areas of the diagram caused by two interfering concavities. This can be achieved if the effects of the concavity on one piece, say B, could be suppressed while a suitable traversal of A is executed, and then expanded back again within the correct areas of the traversal. Intuitively, the first stage of such an approach can be achieved by removing all B's concavities and replacing B by its convex hull. We will denote this conv(B). Recall that to find the Minkowski difference, conv(B) must have clockwise orientation and we will denote this as conv($-B$). As conv(B) is convex Ghosh's method can be applied directly to find $\mathrm{NFP}_{A\mathrm{conv}(B)}$ no matter what the shape of A. The result will be a super-set of the required NFP in that feasible positions where A may nest inside B's concavities will now lie inside the nofit polygon. To obtain NFP_{AB} we need to remove these areas. Conv(B) can be regarded as a copy of B with its concavities replaced by dummy edges. The edge list representation

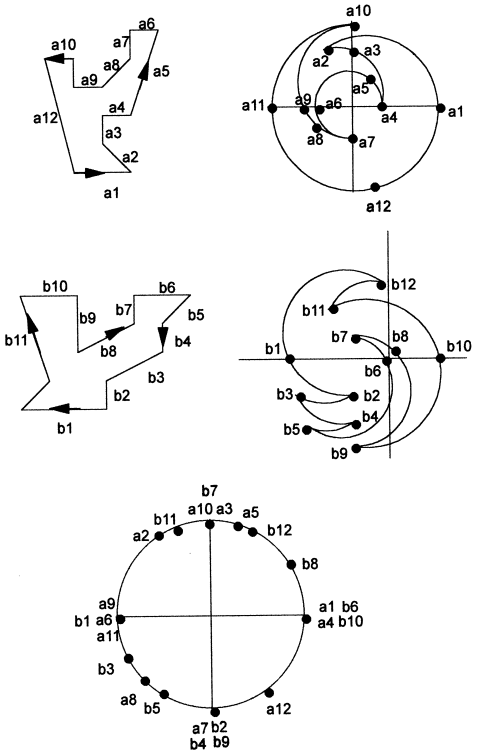

Fig. 6. The merged slope diagrams of two irregular shapes containing many overlapping interactions between concavities.

of $\mathrm{NFP}_{\mathrm{Aconv}(B)}$ will include both positive and negative copies of these edges, each copy lying somewhere between turning points in the traversal of the edges from A. Thus in order to replace dummy edge, b', with the concavity from which it was derived, we need to take account of the region of traversal in which b' appears in the edge list representation. In order to determine the appropriate region of the traversal we first determine the scope of the concavity. This is achieved by considering the positions of all the edge points corresponding to edges in the concavity. The scope of the concavity is the segment lying between the furthest such point from b' in the counter-clockwise direction to the furthest such point in the clockwise direction. We denote these points as b_{R} and b_{L}, respectively. We then take the first occurrence of b' in the edge list, and travel backwards around the slope diagram to the furthest occurrence of b_{R}. The area of transversal we need to consider is determined by travelling from this point in a forward direction to the last occurrence of b_{L}. This region of traversal will pass through the positive and negative occurrences of b', and may also turn backwards and forwards several times as it passes through turning points of A. If there is only one occurrence of b' then this region defines that part of the traversal that must be considered when replacing this single occurrence. If not then the region of traversal is partitioned into k segments, where k is the number of occurrences of b', such that the breakpoints between the segments are turning points of A, and each occurrence of b' occurs in exactly one segment.

Let a_p and a_q represent the limits of the region of traversal containing an occurrence of b'. The relevant part of the list will have form $a_p, a_{p+1}, a_{p+2}, a_{p+3}, \ldots, b', \ldots, a_{q-3}, a_{q-2}, a_{q-1}, a_q$, possibly including some additional B edges as well. b' can now be expanded in this traversal by traversing the slope diagram from b' around the edges of the concavity, *to the furthest occurrence of the relevant edge-point within the region of traversal*, and back to b'. This will involve a forward or anticlockwise section on entering the concavity, at least one turn and a traversal back, passing b', followed by at least one other turn before the return to b'. When following these traversals it is necessary to include any A edges between a_p and a_q that *are passed in the order they are encountered in the edge list*. Those passed in a forward direction will occur with the same sign as in the original list and those passed in the backwards direction will occur with the opposite sign. When moving towards a concavity edge-point b_i other edges of the concavity may be encountered, as may several occurrences of b_i in both negative and positive directions. All occurrences of b_i should be added to the edge list, but all $b_j, j \neq i$, should be ignored. Note that if b' occurs with a minus sign then the concavity is traversed in the reverse direction, but this still involves moving forward through the edge list before turning and following the list in the reverse direction.

The first stage of this process can now be illustrated with reference to the example in Fig. 6 (A full description is given in the appendix). Following this the algorithm is detailed as a series of formal steps that can be executed without the aid of the slope diagram.

First note, that in travelling from a11 to a12 in Fig. 6 we pass vertices of B in the wrong order (b5/b4). In addition, if we attempt to go to b6 from b5 we pass a12 immediately after a7. Therefore we have conflict and will need apply the two phase process.

Conv($-B$) is shown in Fig. 7 with the edges labelled bdx representing the dummy edges. Conv($-B$)'s edges occur in order b1, bd1, bd2, b6, bd3, b10, bd4.

Referring back to Fig. 6 we see that the turning points on A's slope diagram are a2, a4, a7, and a10 and the sorted merged list, as shown in Fig. 7a, is

a1, a4, a5, bd4, a3, a10, a2, b1, a6, a9, a11, a8, bd1, bd2, a7, a12, b6, bd3, b10.

The edge list representation of the nofit polygon is therefore

a1, bd4, a2, a3, $-$ bd4, a4, a5, bd4, b1, a6, bd1, bd2, a7, $-$ bd2, $-$ bd1, a8, a9, $-$ b1, a10, b1, a11, bd1, bd2, a12, b6, bd3, b10.

We now need to consider each occurrence of the dummy edges in turn and replace them by their concavities as implied by the region of traversal. The first dummy edge is bd4, appearing 3 times in total. The right and left extremities of the edge points of the relevant concavity on the slope diagram are b11 and b12. Thus the traversal will start at point S, between a1 and bd4 and continue to point T, between a10 and b1. A suitable way of dividing up this region of traversal is to break it at a2 and a4. The first occurrence of a dummy edge is bd4 in the area of traversal (arc) from a10 to a2 (i.e. in the sequence a10, b1, a11, bd1, bd2, a12, b6, bd3, b10, a1, bd4, a2). We need to replace bd4 by the concavity given by b11, b12 (as shown for Conv($-B$) in Fig. 7a). Merging these edge-points into the relevant sequence we find that b11 occurs between bd4 and a2. Therefore, we do not pass any A edge points on the way to b11. b11 is a turning point of B so we reverse direction and travel back to b12. The first A edge point in this direction is a1, which is further round the slope diagram than b12. Thus we reach b12 and turn again, returning to bd4. Thus bd4 is simply replaced by b11, b12.

We then need to consider in turn other occurrences of bd4, and those of bd1, bd2 and bd3. In some of these cases we will need to insert 'a' edges as described in the appendix. However, as

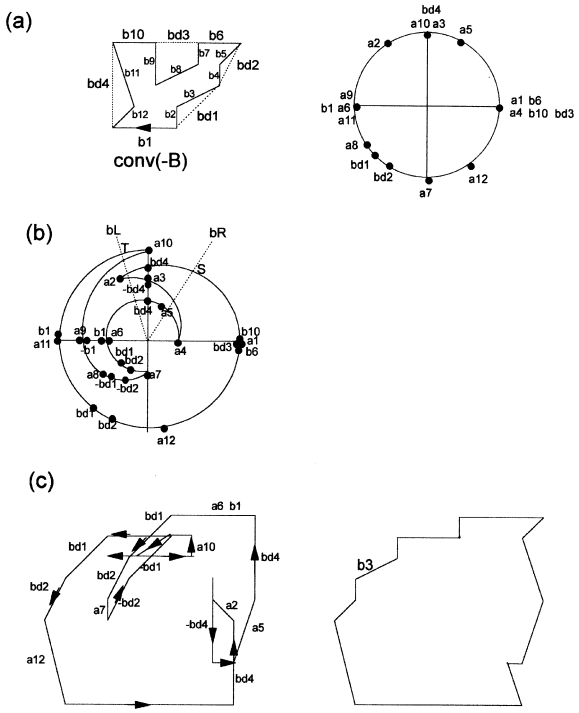

Fig. 7. (a) conv($-B$), and the merged slope diagram, (b) full slope diagram with the scope of bd4 shown by the segment between bR and bL, (c) full NFP for A and conv($-B$) and the boundary of NPF of A and B with internal loops removed.

emphasised earlier, the formal steps of the algorithms can be executed without the visual aid of the slope diagram.

Before detailing the algorithm we highlight the following points. First the algorithm uses a series of sorted lists representing the slope diagrams. These are regarded as circular lists. Thus if such a list is of length k then item $k+1$ is defined as the first item of the list. Second, although the circular nature of the lists means we can start anywhere, for simplicity of implementation we will guarantee to start at a point that is not part of a nested concavity. This will be achieved if the first edge is selected so as to start from the lowest point on polygon A.

7. Algorithm to find the nofit polygon of two polygons A and B

Step 1: Replace B by $-B$, i.e. replace all co-ordinates (x_B, y_B) of B by $(-x_B, -y_B)$

Step 2: Starting at the lowest point on each polygon label the edges in counter clockwise order. Calculate the angle $\theta(i)$ of each edge, i, from the horizontal in a counter clockwise direction. For each edge a let $\alpha(i) = \theta(i) - \theta(i-1)$.
If $\alpha(i) > 180^\circ$ then $\alpha(i) = \alpha(i) - 360^\circ$.

If $\text{sign}(\alpha(i)) \neq \text{sign}(\alpha(i-1))$ mark i as a turning point.
If any turning points have been detected then polygon is non-convex.
Sort the edges into angle order to form sort_list(P), where $P = A$ or $-B$.

Step 3: If $-B$ is convex call Mink($A, -B$), i.e. A plays the role of Q and $-B$ in the role of R. The result is the edge-list for NFP_{AB}
else
if A is convex call Mink($-B, A$). The result is the edge-list for NFP_{AB}
else
go to step 4.

Step 4: Merge sort_list(A) with sort_list($-B$) to form merge_list($A, -B$)

Step 5. Find conv($-B$).
For each dummy edge, bd(i) in conv($-B$), let cavity_list(i) be the list of edges it replaces.

Step 6: Call Mink(A, conv($-B$)).

Step 7: Determine the scope of the relevant concavity and the region of traversal for each dummy edge bd(i). Partition the region into sub-regions, one for each occurrence of bd(i) in nfp_list(A, conv($-B$)). Place a pointer at bd(i). Then starting at bd(i) in merge_list($A, -B$) find the edges in cavity_list(i) in order, finally returning to bd(i).
At the same time scan nfp_list(A, conv(B)) starting by moving forwards from the pointer and thereafter obeying all turns carried out with respect to the traversal of merge_list. Form a sequence incorporating the edges in cavity_list(i) and including an A edge from merge_list($A, -B$) if it is the next A edge in nfp_list(A, conv(B)). Replace bd(i) by the sequence found in this step.

Mink(Q, R).

Step 1: Merge sort_list(Q) with sort_list(R) to form merge_list(Q, R).

Step 2: Set $i = 1$, let $s_1 = q_1$.
Set *direction* $= 1$

Step 3: Set $i = i + 1$.
Search merge_list(Q, R) for q_i moving forward if *direction* $= 1$ and backwards if *direction* $= -1$. If R edge r_j is encountered set $i = i + 1, s_i = r_j$.
When q_i is encountered if $i = 1$ go to step 4.
Otherwise set $k = k + 1$, $s_k = s_i$.
If q_i is a turning point in Q set *direction* $= -$ *direction*.
Repeat Step 3.

Step 4: Return s_1 to s_k as nfp_list(Q, R)

8. Additional factors

The above algorithm requires additional computational procedures before it can be implemented. First, we need to be able to determine the convex hull of piece B. Second, as mentioned earlier, if either piece has concavities the result of the edge-list may be a self-crossing polygon. Although it is possible to use such a polygon within a packing algorithm, it complicates the tests for

point inclusion and will be inefficient as it will normally contain more edges than are necessary to represent its boundary. Thus, we need to consider the removal of internal loops. Finally, in order to use the NFP it is necessary to specify its corners in relation to the origin of the co-ordinate system. In this section, we deal with each of these problems in turn.

8.1. The convex hull of B

There are a number of algorithms for determining a convex hull of an arbitrary set of points. However, our case is simple as we know that our points form a self-crossing polygon. The approach we have used is to follow the vertices of the polygon around in sequence, marking those that give rise to internal angles in excess of 180°. The process is then repeated without the marked vertices, until a pass with no concave vertices is made. The resulting vertices, and the edges defined by visiting them in sequence, form the convex hull.

8.2. Internal loops

The problem of eliminating internal loops from the polygon defined by the edge list requires that we differentiate between those loops that truly lie inside the NFP and those that define the boundary of holes within the NFP. Loops that are oriented in a counter-clockwise direction belong to the former class and those in the clockwise direction belong to the latter. A simple process to remove these loops can be found in Ramkumar [17], although it should be noted that loops are only defined where the path crosses (as opposed to touches) itself, and degenerate loops in the form of an edge that doubles back on itself may need special treatment. It is also worth noting that if we know that there are no inner loops in the NFP (as in the case of star shaped polygons), then the sweep algorithm for finding the outer envelope for a set of line segments, as used by Li and Milenkovic [13], could be used.

8.3. Placing the NFP at the origin

Finally, it is necessary to ensure that the nofit polygon is defined so that its boundary truly represents the set of feasible positions for the reference point on B when A is at the origin. We can establish this by returning to the definition of $A \oplus -B$ as $\{a - b\colon a \in A,\ b \in B\}$. Let a_l and a_b be the vectors defining the leftmost and bottom-most points on A and b_r and b_t be the vectors defining the rightmost and top-most points on B, with A and B assumed to be located at the origin. Then the leftmost point on NFP_{AB} is given by $a_l - b_r$ and the bottom-most point is given by $a_b - b_t$. As we know the co-ordinates of the points represented by these vectors, NFP_{AB} can be positioned correctly.

The full algorithm as described above has been programmed in both Fortran and C and has been used as the basis for a number of irregular packing algorithms. Table 1 shows the results of running a multi-pass sequential placement algorithm known as 'Jostle' on three sets of convex data and three sets of non-convex data from the literature. This shows the benefits that can be derived by using a NFP approach over that of direct trigonometric evaluation. Table 2 shows the times required to calculate the NFPs for five sets of data widely described in the literature. These datasets

Table 1
Ratio of time taken using direct trigonometry to that of using a non-fit polygon approach [6]

	Convex datasets			Non-convex datasets		
Data set	C1	C2	C3	NC1	NC2	NC3
Ratio	1.71	3.63	4.90	1.24	1.82	1.56

Table 2
Time required to calculate NFP's for five typical industrial datasets (Seconds – DEC Alpha 3000/400). See [4] for details of shapes & quantities

Data set	DS1	DS2	DS3	DS4	DS5
Time (s)	0.32	0.22	0.23	0.38	0.36

involve up to 99 pieces of up to eight piece types and are detailed in Benell and Dowsland [4]. The results show the very small computation times involved.

9. Conclusions

This paper has introduced a new method for finding the nofit polygon of two polygons. The benefit of this method over that described in Ghosh [8] is that, although it is based on the mathematics of vector addition it is simple to implement. This is because it only relies on routines to sort and merge lists; to calculate the slopes of the edges; and to determine the crossing points of pairs of line segments. It is also very effective as it can deal directly with any pair of simple polygons (i.e. polygons without internal holes) regardless of their complexity. This avoids the necessity for routines to decompose the pieces into a suitable form and the subsequent assembly of the required NFP. As it initially produces a self-crossing polygon it is also able to deal with NFPs with holes and can therefore cater for nesting pieces that cannot slide into position, without specialist treatment. Although additional calculations to determine the points of intersection between pairs of edges are required in order to move from a self-crossing polygon to a non-crossing boundary, similar calculations are inherent in all other solution approaches. Our experiences with irregular packing algorithms have convinced us of the benefits of pre-processing the NFPs (see for example Dowsland et al [6]). By combining Ghosh's idea of traversals of the slope diagram with the simple concept of using the convex hull of one of the pieces, we have produced a method that is both mathematically elegant and simple to implement. We hope that this will lead to a more widespread use of this powerful concept by other researchers in this field.

Appendix

As has been illustrated in the paper, the algorithm can be described in a series of formal steps that can be executed without the visual aid of the slope diagram. For those who may wish to follow

through the complete procedure visually, the description below works through the example in Fig. 7.

In Fig. 7b we need to consider each occurrence of the dummy edges in turn and replace them by their concavities as implied by the region of traversal. The first occurrence of a dummy edge is bd4 in the area of traversal (arc) from a10 to a2. As described in the main text bd4 is simply replaced by b11, b12.

The next occurrence of bd4 is in the sequence a2, a3, − bd4, a4. bd4 has negative sign so will be replaced by − b12, − b11 together with any necessary *A* edges and any further positive or negative passes over b11 or b12 as necessary. We begin by moving *forwards* with respect to the relevant section of the edge list. In travelling to b12 we do not pass a4, however, on the backward journey to b11 we pass a3. a3 has positive sign in the original sequence. Therefore we insert it here with negative sign. At b11 we turn, passing a3 in a forward direction before reaching bd4. Thus we replace − bd4 by − b12, − a3, − b11, a3.

Next we consider bd4 on the traversal a4, a5, bd4, b1, a6, bd1, bd2, a7 … a10, b1 …. On this occasion we do not pass any *A* edges on the forward pass to the initial occurrence of b11. However, we do pass a6, a7, a8, a9, − b11, and a10 on the way to the outermost occurrence of b11. We then reverse the sign of the a edges on the way back towards bd4. We pass a5 in the backward pass from bd4 to b12. As this is backwards traversal we insert − a5 here and + a5 in the return journey to bd4. Therefore, this occurrence of bd4 is replaced by b11, a6, a7, a8, a9, − b11, a10, b11, − a10, − a9, − a8, − a7, − a6, − a5, b12, a5.

Next we come to bd1. Using break points at a7 and a10 will partition the scope into 3 single arcs, each containing one occurrence of bd1. In the first arc between a4 and a7 there is no interference from *A* edges and so we replace it by b2, b3. On the second arc the traversal of − bd1 passes a8 on the way to b3 and again on the way back. Thus the traversal is a8, − b3, − a8, − b2. On the final pass no *A* edges are encountered and bd1 is replaced as above.

For bd2 the same three arcs cover the scope, but no edges are passed on any traversals. Therefore bd2 is replaced by b4, b5 and − bd2 by − b5, − b4 throughout.

Finally, we consider bd3 with scope define by the half circle from b7 to b9. In travelling from bd3 to b7 we pass b7, a2, a3, − b7, a4, a5, b7, and then return again with the a edges given negative sign and b8 inserted as appropriate. Finally, we travel from b8 to b9 and back again This gives: a1, b7, a2, a3, − b7, a4, a5, b7, − a5, b8, − a4, − b8, − a3, − a2, b8, − a1, − a12, b9, a12.

Thus we have our final sequence of edges:

a1, b11, b12, a2, a3, − b12, − a3, − b11, a3, a4, a5, b11, a6, a7, a8, a9, − b11, a10, b11, − a10, − a9, − a8, − a7, − a6, − a5, b12, a5, b1, a6, b2, b3, b4, b5, a7, − b5, − b4, a8, − b3, − a8, − b2, a8, a9, − b1, a10, b1, a11, b2, b3, b4, b5, a12, b6, a1, b7, a2, a3, − b7, a4, a5, b7, − a5, b8, − a4, − b8, a3, − a2, b8, − a1, − a12, b9, a12, b10.

When the inner loops are removed the result is the polygon shown in Fig. 7.

References

[1] Adamowicz M, Albano A. Nesting two dimensional shapes in rectangular modules. Computer Aided Design 1976;8(1):27–33.

[2] Art RC. An approach to the two dimensional irregular cutting stock problem. IBM Cambridge Scientific Centre, Report 36-Y08, 1966.

[3] Bennell JA. Incorporating problem specific knowledge into a local search framework for the irregular shape packing problem. Ph.D. thesis, University of Wales, Swansea, 1998.
[4] Bennell JA, Dowsland KA. A tabu thresholding implementation for the irregular stock cutting problem. International Journal of Production Research 1999;37(18):4259–75.
[5] Cuninghame-Green R. Geometry, shoemaking and the milk tray problem. New Scientist 12 August 1989; 1677 :50–3.
[6] Dowsland KA, Dowsland WB, Bennell JA. Jostling for position: local improvement for irregular cutting patterns. Journal of the Operational Research Society 1998;49(6):647–58.
[7] Dowsland KA, Dowsland WB. Solution approaches to irregular nesting problems. European Journal of Operational Research 1995;84:506–21.
[8] Ghosh PK. A unified computational framework for minkowski operations. Computers and Graphics 1993;17(4):357–78.
[9] Ghosh PK. An algebra of polygons through the notion of negative shapes. CVGIP: Image Understanding 1991;54(1):119–44.
[10] Grünbaum B. Convex polytopes. London: Interscience, 1967.
[11] Heckmann R, Lengauer T. A simulated annealing approach to the nesting problem in the textile manufacturing industry. Annals of Operational Research 1995;57:103–33.
[12] Konopasek M. Mathematical treatment of some apparel marking and cutting problems. US Department of Commerce Report, 99-26-90857-10, 1981.
[13] Li Z, Milenkovic V. Compaction and separation algorithms for non-convex polygons and their application. European Journal of Operations Research 1995;84:539–61.
[14] Mahadevan A. Optimisation in computer aided pattern packing. Ph.D. thesis, North Carolina State University, 1984.
[15] Milenkovic V, Daniels K, Li Z. Placement and compaction of non convex polygons for clothing manufacture. Fourth Canadian Conference on Computational Geometry, St John's, Newfoundland, 1992.
[16] Oliveira JF, Gomes AM, Ferreira JS. A new constructive algorithm for nesting problems. Conference Paper, IFORS, Vancouver, BC, Canada, 1996.
[17] Ramkumar GD. An algorithm to compute the minkowski sum outer-face of two simple polygons. Proceedings of the 12th Annual Symposium on Computational Geometry FCRC 96, 1996.
[18] Special interest group on cutting and packing (SICUP), http://prodlog.wiwi.uni-halle.de/sicup/.
[19] Stoyan YG, Ponomarenko LD. Minkowski sum and hodograph of the dense placement vector function. Reports of the SSR Academy of Science, SER.A 10, 1977.
[20] Theodoracatos VE, Grimsley JL. The optimal packing of arbitrarily-shaped polygons using simulated annealing and polynomial-time cooling schedules. Computer Methods in Applied Mechanics and Engineering 1995;125:53–70.

Julia Bennell is a lecturer in the Department of Management at the University of Southampton, UK. She has a B.Sc. in Pure Mathematics and Management Science and a Ph.D. in Operational Research, both from the University of Wales Swansea. Her research interests include the use of sequential search heuristics and optimisation procedures in the solution to the irregular shaped stock cutting problem.

Kathryn Dowsland has a B.Sc. in Pure Mathematics, and the degrees of M.Sc. and Ph.D. in Operational Research from the University of Wales. She worked in the steel industry before joining the University of Wales Swansea, where she is a Reader in Operational Research in the European Business Management School. Her research interests include graph-theoretic approaches to problem solving and the use of tabu search, simulated annealing and genetic algorithms, particularly in the application areas of packing, scheduling and timetabling.

Bill Dowsland is a graduate of Industrial Engineering and has a Ph.D. in Information Systems. He is a senior lecturer within the European Business Management School at Swansea and has particular teaching interests in the business applications of information technology. His research focuses on the solution of problems in the fields of cutting and packing using a variety of heuristic techniques.