Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 184 (2008) 397–415
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Invited Tutorial

The geometry of nesting problems: A tutorial

Julia A. Bennell [a,*], Jose F. Oliveira [b,c]

[a] Centre for Operations Research, Management Science and Information Systems, University of Southampton, Highfield, Southampton SO17 1BJ, UK
[b] Faculdade de Engenharia da Universidade do Porto, Portugal
[c] INESC Porto – Instituto de Engenharia de Sistemas e Computadores do Porto, Portugal Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

Received 20 July 2006; accepted 15 November 2006
Available online 31 December 2006

Abstract

Cutting and packing problems involving irregular shapes is an important problem variant with a wide variety of industrial applications. Despite its relevance to industry, research publications are relatively low when compared to other cutting and packing problems. One explanation offered is the perceived difficulty and substantial time investment of developing a geometric tool box to assess computer generated solutions. In this paper we set out to provide a tutorial covering the core geometric methodologies currently employed by researchers in cutting and packing of irregular shapes. The paper is not designed to be an exhaustive survey of the literature but instead will draw on the literature to illustrate the theory and implementation of the approaches. We aim to provide a sufficiently instructive description to equip new and current researchers in the area to select the most appropriate methodology for their needs.

Keywords: Cutting; Packing; Geometry; Optimisation

1. Introduction

Cutting and packing problems involving irregular shapes arise in a wide variety of industries including; garment manufacturing, sheet metal cutting, furniture making and shoe manufacturing. An example of a layout from the garment manufacturing industry is provided in Fig. 1. As a result the range of problem variants involving irregular shapes, referred to here as nesting problems, is a core area of research in the field of cutting and packing. The problem is NP-complete and as a result solution methodologies predominantly utilise heuristics. A further defining characteristic of nesting problems is the requirement to develop powerful geometric tools to handle the wide variety and complexity of shapes that need to be packed. In this paper we propose to provide detailed explanations of the most popular techniques for handling the geometry when solving nesting problems and provide guidance on their implementation, strengths and weaknesses.

The remainder of this paper is organized in six sections. In the next section the nesting problem is defined more precisely and the key differences

* Corresponding author. Tel.: +23 80595671; fax: +23 805933844.
E-mail addresses: J.A.Bennell@soton.ac.uk (J.A. Bennell), jose.oliveira@fe.up.pt (J.F. Oliveira).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.11.038

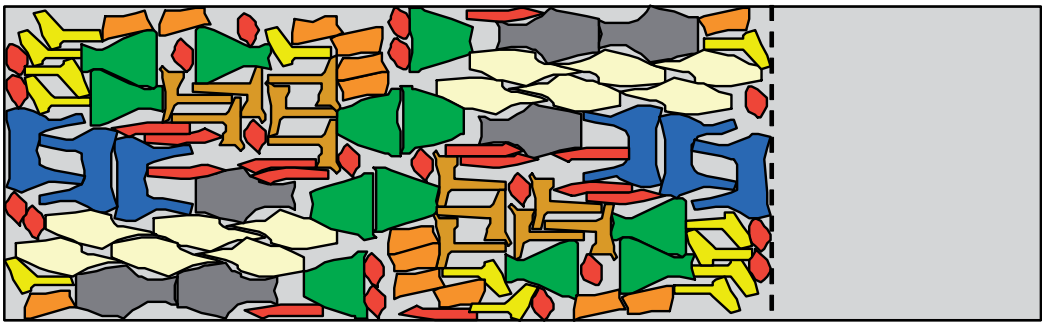


Fig. 1. An example layout from garment manufacturing.

between nesting problems and other cutting and packing problems are highlighted. In particular, the importance of the geometry is explained. In Section 3 geometric approaches based on pixel or raster representations are introduced. In Section 4 the concept of the *D* function is presented including algorithms to handle geometric tasks that use this function. In the following section, the nofit polygon is introduced and two different algorithms for its generation are described, one based on the sliding analogy and another using Minkowski sums. Polygon decomposition is also discussed in this section. In Section 6 the concept of Phi function is presented with illustrative examples.

2. Problem definition

Nesting problems are one of many terms used to identify the group of problems that will be discussed. A discussion of nomenclature is beyond the scope of this paper, however, it is helpful to identify the other names under which these problems are referenced. Some commonly used terms are irregular shape stock cutting, irregular packing, polygon placement, marker making, non-convex cutting stock, 2-D packing problems or some combination of any of these terms. The term *nesting problems* is adopted in this paper and can be defined as follows:

where more than one piece of irregular shape must be place in a configuration with the other piece(s) in order to optimise an objective

Irregular shapes are defined as simple polygons, and in some cases, polygons that may contain holes. When pieces include curved edges, it is common to approximate them by an enclosing polygon, where a series of tangents to the curve form the polygonal edges. Recently new research has emerged that permit the curved edges to be retained in their original form. It is appropriate at this point to consider whether shapes such as circles or triangles would be classified as nesting problems. There is no clear conclusion that can be drawn from the literature. However, we suggest that problems that involve non-trivial handling of the geometry would be classified as a nesting problem. Hence circles are not, since overlap can be detected simply by evaluating the distance between the centres of two candidate circles.

The above definition is clearly very general and within it many variants of the problem can be defined. These can be identified through a number of key characteristics; homogeneity/heterogeneity of the data; stock sheet size, shape, number and quality (with respect to the homogeneity of the stock sheet surface); and single or multi objective.

Dyckhoff (1990) proposed a useful classification of cutting and packing problems. His classification partitioned the problems by four fundamental criteria; dimensionality, objective of the assignment, large objects and small items. Dimensionality is divided by one-dimensional, two-dimensional, three-dimensional and *n*-dimensional, where *n* is great than 3. The type of assignment was partitioned into: those problems that require all the pieces to be arranged hence minimising waste and those that select and arrange a subset of pieces hence maximising the profit gained from those pieces. The large objects are the stock sheets and may either be single, multiple and identical, or multiple and different. Finally, the small items are the pieces that are partition according to the number and level of homogeneity. Dyckhoff's typology has since been revised by Waescher et al. (2005) in order to remove some ambiguity that has arisen since its

publication and provide a more intuitive classification of problem variants. Their typology retains the first two detailed by Dyckhoff; dimensionality and objective of the assignment. The categories for large object and small objects are redefined, and a new characteristic is identified; shape of the small items. Using Waescher, Hauber and Schumann's typology, nesting problems, in general, would be placed in the *open-dimension problem* category, with the refinement of being two-dimensional and irregular. The new typology provides a useful progression towards a consistent nomenclature for cutting and packing problems.

2.1. How is nesting different from other cutting and packing problems?

A comparison of the physical attributes of cutting and packing problems in order to group them has already been addressed through the discussion of the typology. Here we are interested in the two-dimensional problems, which can be further partitioned into regular packing and irregular packing. These differences are tangible and observable; perhaps the more important question is how these differences influence the approaches we might take to solve the problem. The paper will discuss the increased complexity of ensuring pieces are allocated to feasible placement position when they are irregular. However, simply deriving more sophisticated feasibility tests may not be sufficient to modify a successful rectangle packing approach to also be successful for irregular shapes.

Although there are an infinite number of different rectangles with respect to their size and ratio of length and width, the fact that all pieces are rectangular allows you to cut down the potential placement positions to a finite set. Where as the infinite variety of sizes and shapes of simple polygons provide a much greater challenge. In the case of rectangular pieces, it is easy to reduce the stock sheet to a discrete set of candidate locations by defining a grid of feasible placement positions by the largest common divisor of all the rectangular edges. The same approach can be applied for instances where the pieces only contain edges that are orthogonal to each other and the edges of the stock sheet. However, if this is not the case then defining such a grid can remove good solutions from the solutions space. Hence, in all but the orthogonal case described above, the stock sheet is continuous and for each piece in the data set the number of feasible placement positions on the stock sheet is infinite. As a result, an implementation for nesting problems has to embed mechanisms into the approach for reducing the solution space, preferably without removing the best solutions. In addition to the increased complexity of the solution space, the calculation of feasibility of the solution is significantly more computationally intensive. Hence, many fewer solutions may be evaluated in the same run time. Other differences worth noting are that bounds can be more easily found for rectangular problems (e.g. Beasley, 1985; Letchford and Amaral, 2001; Y and Kang, 2002) and potentially any rotation of the pieces may be considered.

2.2. Why is the geometry important?

The most visible attribute of nesting problems and the first obstacle researchers come up against is the geometry. By this we mean answering the question; given a position on the stock sheet of two pieces, does this result in them overlapping, touching, or are they separated? The answer to this question is trivial to the human eye. To write a computer programme to determine this information is much more complex since the answer can only be found from processing a set of vertices or in some cases categories of pixels. Developing a set of tools to assimilate the geometry is a non-trivial task and potentially a barrier that stifles academic research in this area. There exist a number of solutions to this problem ranging from simple to complex. However, each has their idiosyncrasies. Determining which is the most appropriate approach to implement is not just a matter of how well they perform, but also how difficult they are to implement robustly. Also the selected approach for solving the nesting problem will impact the level of benefit from time invested in implementing and pre-processing highly efficient geometric tools. The remainder of this paper is dedicated to describing the most common approaches found in the literature, these are; the raster method, direct trigonometry, the nofit polygon and the phi function.

3. Pixel/raster method

Raster methods are approaches that divide the continuous stock sheet into discrete areas, hence reducing the geometric information to coding the data by a grid represented by a matrix. However, different authors have used different codification

schemes, which are largely driven by the placement algorithm that will use this geometric information.

Oliveira and Ferreira (1993) propose the simplest coding scheme that uses the value of 1 to code the existence of piece and the value of 0 to denote the empty space, illustrated in (Fig. 2). Placing one piece on the stock sheet is simply a matter of adding the matrix of 0 s and 1 s, which represents the piece, to the matrix representing the actual layout. For a certain position of the layout, the value of the corresponding cell in the matrix gives the number of pieces that occupy that position. If the value is greater than 1 then overlap among pieces occurs.

Segenreich and Braga (1986) devised a different codification scheme, that enables them to detect contacting positions between pieces as well as overlap. In this case, the number 1 is used to code the frontier of the pieces and the number 3 to code the interior. After adding such a matrix to the matrix layout, numbers greater than or equal to 4 indicate infeasibility, in the form of a frontier-interior overlap or interior-interior overlap. Numbers less than or equal to 2 code feasibility, with the number 2 meaning a contact between two pieces (Fig. 3).

Both anterior codifications denote the empty space by 0 and use numbers greater than or equal to 1 to code the piece itself. Babu and Babu (2001) reverse this idea. Fig. 4a illustrates the coding of an irregular stock sheet with defects. Here the pixels completely within the interior of the stock sheet are assigned a zero. Pixels that are outside or on the boundary are coded with a number greater than zero determined by assigning a 1 to the right most non-zero pixel and cumulatively adding 1 moving from right to left. Pieces are coded similarly but the boundary of the piece is also assigned a zero, as shown in Fig. 4b. The purpose of this coding is that the value of each cell gives the number of cells that it is necessary to move right so that a potentially feasible placement is found. A particular benefit is when using a bottom-left placement procedure based on the pieces movement over the layout, as it will allow many cells to be skipped in just one step. However, actualising the layout from the matrix representation is more complex and time consuming when compared with the previous approaches. After placing one piece on the stock sheet the respective cell values of the stock sheet are changed from 0

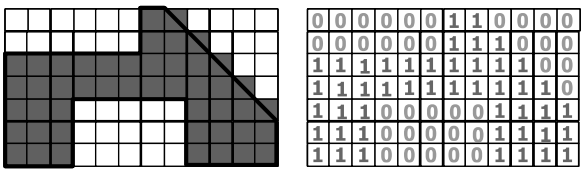


Fig. 2. The 0–1 raster representation for irregular pieces.

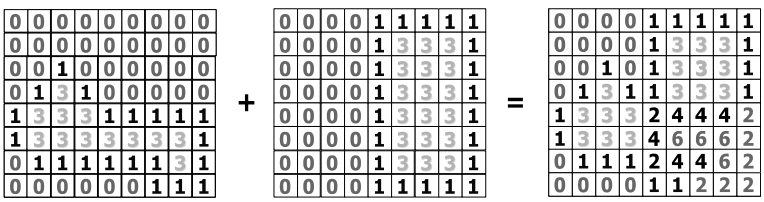


Fig. 3. A non-Boolean raster representation for irregular pieces.

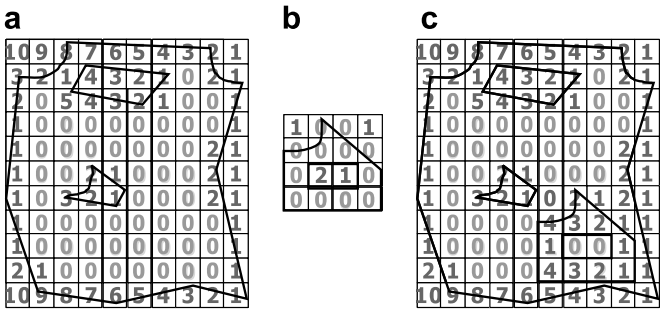


Fig. 4. Raster method proposed by Babu and Babu (2001): (a) stock sheet with defects, (b) piece and (c) stock sheet with a piece placed.

to a positive value following the same codification scheme, as illustrated in Fig. 4c.

The advantage of all the described raster methods is that calculating the distance a piece must shift in order to eliminate infeasibility or where to place two pieces so that they are in contact with each other is just a matter of counting cells in the desired direction. Also raster representations are simple to code, can represent non-convex and complex pieces as easily as simple polygons and are reasonably fast when checking the geometric feasibility of the layouts. However, the disadvantages are that these methods are memory intensive and cannot exactly represent pieces with non-orthogonal edges. Increasing the representation's accuracy by refining the size of the grid unit leads to an increase in the size of the matrices and results in greater memory usage and running times for feasibility checks.

4. Direct trigonometry and the *D* function

Given that the raster representation cannot accurately represent the irregular shape of the pieces, the alternative is to use the polygons directly. When representing the pieces as polygons the amount of information is proportional to the number of vertices and does not depend on the absolute size of the pieces or of the layout. However unlike the raster method, the feasibility or quality of a placement is not implied by the geometry and as a result another evaluation method must be employed. An obvious candidate is to use direct trigonometry where there exist well know tests for line intersection and point inclusion. Although these tests are more computationally complex in comparison to the raster methods, run time cannot be directly compared. For the raster method the time to check feasibility is quadratic in the grid size, whereas for direct trigonometry it is exponential in the number of edges of the pieces.

In this section we describe an efficient approach for evaluating overlap between two polygons using trigonometry. We assume the pieces are represented as a closed series of vertices and edges. Fig. 6 illustrates examples of the relative positions of the two polygons that are identified by the approach. Fig. 6c and d are the two instances where there is overlap, the former can be identified through direct analysis of edges and the latter by identifying that one or more vertices of one polygon is inside the other. However, before considering these tests, there is a fast higher-level test that can be applied as

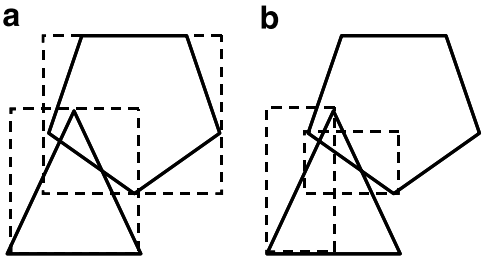


Fig. 5. (a) If two polygons overlap, then the enclosing rectangles of the pieces must overlap. (b) If two edges intersect then the enclosing rectangles of the edges must intersect.

depicted in Fig. 5. Given that the bounding boxes of the two polygons must overlap if there is to be overlap between the polygons (Fig. 5a), then the process can be made more efficient by first employing this simple test. This principle can be extended to the bounding boxes of edges (Fig. 5b). Ferreira et al. (1998) studied the effect of using bounding boxes both at the polygon and edge analysis level for 6 different benchmarks instances. A reduction between 90.7% and 97.6% is reported for the number of direct edge intersection tests required, with a reduction ranging from 96.0% to 99.4% at the edge analysis level. While the complexity of the polygons impact on the effectiveness of the edge bounding box test, Ferreira et al. found that the percentage of reduction was never below 90%. As a result we suggest the following hierarchy of tests.

Test 1: Do the bounding boxes of the polygons overlap?
No – polygons do not overlap (Fig. 6a)
Yes – apply test 2

Test 2: For each pair of edges from different polygons, do their respective bounding boxes overlap?
No for all – polygons do not overlap (Fig. 6b)
Yes – for those edges with overlap apply test 3

Test 3: For each pair of edges from different polygons, does the edge analysis indicate an intersection?
No for all – apply test 4
Yes – as soon as one pair of edges indicate an intersection then the polygons overlap (Fig. 6c)

Test 4: For one vertex of each polygon, does that vertex reside inside the other polygon?

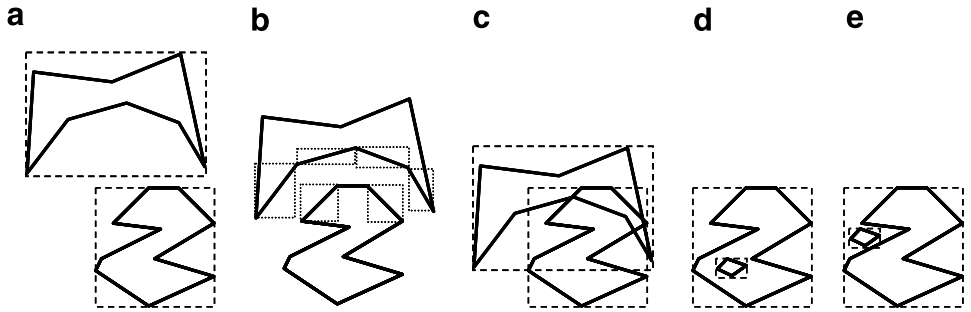


Fig. 6. Examples of the relative position of two polygons identified by tests 1 to 4.

No for both – polygons do not overlap (Fig. 6e)
Yes for one – polygons overlap (Fig. 6d)

The bounding box test is a trivial operation of comparing the minimum and maximum coordinates of the polygon or edge bounding boxes. Preparata and Shamos (1985) described a simple point inclusion test that can be used as Test 4. Hence our focus is on Test 3. The *D*-function has been demonstrated as an efficient tool for characterising the relationship between two edges. The relationship can then be interpreted as to whether it indicates overlap. First we will describe how the *D*-function characterises the edges.

The *D*-function can be defined as follows:

$$D_{ABP} = ((X_A - X_B)(Y_A - Y_P) - (Y_A - Y_B)(X_A - X_P)) \quad (1)$$

Proposed by Konopasek (1981), the *D*-function gives the relative position of a point *P* with respect to an oriented edge *AB* (Fig. 7). It is based on the equation of the distance from a point to a straight-line. The line is unbounded and is therefore called a supporting line for the edge. Its interpretation is the following: if $D_{ABP} > 0$ then point *P* is on the left side of the supporting line of edge *AB*; if $D_{ABP} < 0$ then point *P* is on the right side of the supporting line of edge *AB*; if $D_{ABP} = 0$ then point *P* is on the supporting line of edge *AB*. This interpretation assumes that the origin of the coordinate system, where the coordinates (X_i, Y_i) are defined, is the bottom-left corner, i.e. the *x*-coordinates increase to the right and *y*-coordinates increase upward.

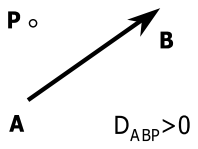


Fig. 7. Interpretation of the *D*-function.

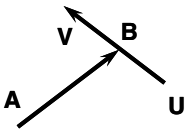
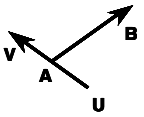
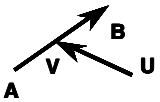
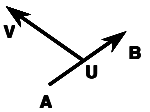
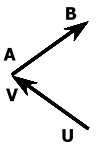
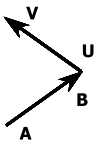
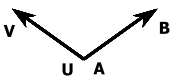
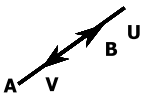
Mahadevan (1984) described how the relative position of two oriented edges can be determined using the *D*-function. In Table 1 the different cases (a to j) that lead to an edge intersection or touch are presented along with their description in terms of their *D*-functions.

The supporting lines that correspond to the bounded edges *A*, *B* and *U*, *V* are considered by the *D*-function to be infinite. As a result for case (j), where the two edges are collinear, the condition presented above is a necessary but not sufficient condition for the two edges to overlap. As a result an additional test is required to identify if a vertex of one edge lies in between the vertices of the other.

Clearly when edges are touching but not overlapping, the *D*-function does not provide a direct answer to the question, do these polygons overlap? The relationships, a to j, detailed in Table 1, only indicate overlap under certain conditions. Hence a second test must be applied. We assume that the polygons have counter clockwise orientation, i.e. that the interior of the polygon is on the left side of the oriented edges, and that the oriented edge AB belongs to one of the polygons and oriented edge *UV* belongs to the other polygon, then the conditions that indicate overlap, with reference to Table 1, are as follows:

1. Case (a).
2. Case (b), when *A* is on the left side of *UV*.
3. Case (c), when *B* is on the left side of *UV*.
4. Case (d), when *U* is on the left side of *AB*.
5. Case (e), when *V* is on the left side of *AB*.

Table 1
D-functions analysis of the relative position of two oriented edges

a	$D_{ABU} \neq 0 \wedge D_{ABV} \neq 0 \wedge D_{ABU} \neq D_{ABV} \wedge D_{UVA} \neq 0 \wedge D_{UVB} \neq 0 \wedge D_{UVA} \neq D_{UVB}$	
b	$D_{ABU} \neq 0 \wedge D_{ABV} \neq 0 \wedge D_{ABU} \neq D_{ABV} \wedge D_{UVA} \neq 0 \wedge D_{UVB} = 0$ Additionally, if $D_{UVA} < 0$ then A is on the right side of the oriented edge UV and if $D_{UVA} > 0$ then A is on the left side of the oriented edge UV	
c	$D_{ABU} \neq 0 \wedge D_{ABV} \neq 0 \wedge D_{ABU} \neq D_{ABV} \wedge D_{UVA} = 0 \wedge D_{UVB} \neq 0$ Additionally, if $D_{UVB} < 0$ then B is on the right side of the oriented edge UV and if $D_{UVB} > 0$ then B is on the left side of the oriented edge UV	
d	$D_{ABU} \neq 0 \wedge D_{ABV} = 0 \wedge D_{UVA} \neq 0 \wedge D_{UVB} \neq 0 \wedge D_{UVA} \neq D_{UVB}$ Additionally, if $D_{ABU} < 0$ then U is on the right side of the oriented edge AB and if $D_{ABU} > 0$ then U is on the left side of the oriented edge AB	
e	$D_{ABU} = 0 \wedge D_{ABV} \neq 0 \wedge D_{UVA} \neq 0 \wedge D_{UVB} \neq 0 \wedge D_{UVA} \neq D_{UVB}$ Additionally, if $D_{ABV} < 0$ then V is on the right side of the oriented edge AB and if $D_{ABV} > 0$ then V is on the left side of the oriented edge AB	
f	$D_{ABU} \neq 0 \wedge D_{ABV} = 0 \wedge D_{UVA} \neq 0 \wedge D_{UVB} = 0$ Additionally, if $D_{ABU} < 0$ then U is on the right side of the oriented edge AB and if $D_{ABU} > 0$ then U is on the left side of the oriented edge AB	
g	$D_{ABU} \neq 0 \wedge D_{ABV} = 0 \wedge D_{UVA} = 0 \wedge D_{UVB} \neq 0$ Additionally, if $D_{ABU} < 0$ then U is on the right side of the oriented edge AB and if $D_{ABU} > 0$ then U is on the left side of the oriented edge AB	
h	$D_{ABU} = 0 \wedge D_{ABV} \neq 0 \wedge D_{UVA} \neq 0 \wedge D_{UVB} = 0$ Additionally, if $D_{ABV} < 0$ then V is on the right side of the oriented edge AB and if $D_{ABV} > 0$ then V is on the left side of the oriented edge AB	
i	$D_{ABU} = 0 \wedge D_{ABV} \neq 0 \wedge D_{UVA} = 0 \wedge D_{UVB} \neq 0$ Additionally, if $D_{ABV} < 0$ then V is on the right side of the oriented edge AB and if $D_{ABV} > 0$ then V is on the left side of the oriented edge AB	
j	$D_{ABU} = 0 \wedge D_{ABV} = 0$	

6. Case (f). In this case it is necessary to look ahead in both polygons and use edges BC and VZ, respectively. Additionally one of the following conditions must hold in order to have an overlap between the two polygons (Fig. 8):
   (i) $D_{ABC} < 0 \wedge D_{ABU} < 0$
   (ii) $D_{ABC} < 0 \wedge D_{BCU} < 0$
   (iii) $D_{ABC} > 0 \wedge D_{ABU} > 0 \wedge D_{BCU} > 0$
   (iv) $D_{UVZ} < 0 \wedge D_{UVA} < 0$
   (v) $D_{UVZ} < 0 \wedge D_{VZA} < 0$
   (vi) $D_{UVZ} > 0 \wedge D_{UVA} > 0 \wedge D_{VZA} > 0$

No other case concerning the superposition of two vertices (g, h, i) needs to be analysed as all those

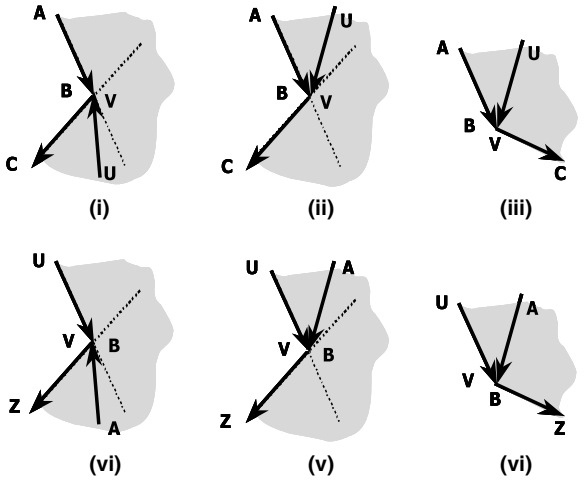


Fig. 8. Cases in which B touches V and overlap exists between the respective polygons.

cases lead, in the previous or in the next edge, to case (f).

7. Case (j) when the oriented edges AB and UV have the same direction and the two edges overlap. These additional conditions are not verifiable by the use of the D-functions, but can be tested simply by comparing coordinates.

Employing the D-function to deal with the geometric level of nesting problems provides an accurate approach, in the sense that polygons are exactly represented. However, from the viewpoint of computational effort, calculations must be performed with floating-point units, which is much slower than addressing memory positions and summing integers (raster representation). Moreover, each time a polygon's position is changed, the feasibility of the placement must be checked. As a result, all these calculations have to be repeated from scratch and performed during the search for a packing arrangement. Hence, this approach does not lend itself to nesting algorithms based on any iterative search process since the time spent on evaluating the geometry can restrict the time, and therefore the extent, of the search. However, constructive algorithms, for instance approaches that build solutions based on a previously defined sequence of pieces, may quite efficiently use D-functions to tackle the geometric issues of nesting problems.

5. The nofit polygon

The nofit polygon has become an increasingly popular option for dealing with the geometry since it is more efficient than direct trigonometry, particularly when using an iterative search, yet shares the benefit of accuracy by using the original edges of the polygon. In addition, the concept opens up new options for placement strategies. In essence the NFP is a polygon derived from aggregating the two component polygons. It can be used, along with the vector difference of the position of the two polygons, to determine whether these polygons overlap, touch, or are separated, by conducting a simple test to identify whether the resultant vector is inside the NFP. Such a test has complexity $O(n)$, where n is the number of edges in the NFP, assuming the calculation of the NFP is performed in a pre-processing phase. The computational efficiency gained by utilising this concept is very attractive. However, the significant drawback of this approach is due to the difficulties in developing a robust NFP generator for general non-convex polygons. Three core approaches exist in the cutting and packing literature; the orbiting algorithm of Mahadevan (1984), later improved by Whitwell (2005); Minkowski sums used by Milenkovic et al. (1991) and Bennell et al. (2001), (in both cases motivated by the work of Ghosh (1991)); and decomposition into star shaped polygons (Li and Milenkovic, 1995) or convex polygons (Watson and Tobias, 1999; Agarwal et al., 2002). Each approach will be briefly explained in the following sections after a more detailed discussion of the properties and application of the NFP.

5.1. NFP – what is it and how does it work?

The NFP of two polygons A and B, denoted as NFP_{AB} is the resulting polygon from a sliding operation between A and B where each has a specific role within the operation. Both polygons have fixed orientation. A has a fixed position where the origin is assumed to be at $(0,0)$, B is the tracing polygon that moves around the perimeter of A to perform the sliding operation. The NFP is defined 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 is performed in such a way that A and B always touch, but never overlap. The locus of the reference point forms a closed path that is NFP_{AB}. Fig. 9a illustrates this tracing movement. If the roles are reversed then the resulting NFP, NFP_{BA} is NFP_{AB} rotated by 180°. Clearly while A is fixed at $(0,0)$, if B is placed so that the reference point is inside NFP_{AB} then A

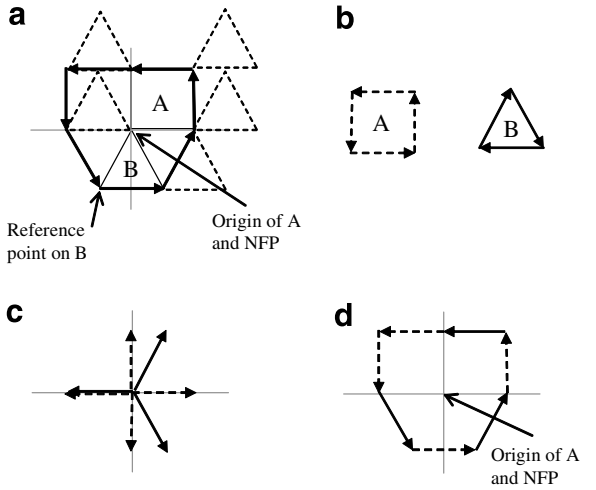


Fig. 9. (a) Tracing movement of polygon B around A to form the NFP, (b) the orientation of A and B and (c) the order of the slopes of A and B, (d) the NFP.

and B overlap and if the reference point is on the boundary then A and B touch. Thus the interior of NFP_{AB} represents all intersecting positions of A and B. This result can be extended for the case when the position of A is not restricted to $(0,0)$. If A is moved to position (x,y) then the position of the reference point of B must first be transposed by $(-x,-y)$ before testing its relative position with $\mathrm{NFP}_{AB.}$ Hence overlap between two polygons can be identified by testing whether the result of subtracting the position of the origin of A from the reference point of B is inside $\mathrm{NFP}_{AB.}$

Cuninghame-Green (1989) presents a simple algorithm for calculating the NFP for purely convex polygons. Although convex polygons are the simplest case, this paper is a good starting point for understanding the NFP. Two key ideas are developed. First, the different roles of each polygon are recognised, and as a result the polygons must have different orientations (Fig. 9b). We will adopt the convention that polygon A (fixed polygon) is counter clockwise and polygon B (orbiting polygon) is clockwise. Second, the order of the edges of the NFP of two convex polygons is equivalent to sorting the edges of both polygons in slope order (Fig. 9c and d). These ideas provide a good foundation for understanding the approaches of both Mahadevan (1984) and Ghosh (1991).

While both polygons are convex, the concept and its realisation are quite simple. However, the difficulties arise when one or both of the polygons contain concavities. A pair of polygons that are able to reach all non-intersecting touching positions through sliding are termed as relatively simply connected polygons, and the resulting NFP will be simply connected. Pairs of polygons that do not have this property may be simply connected themselves but not relative to each other and result in an NFP that contains a hole, defined as a multiply connected polygon. Some examples of the more difficult cases are illustrated in Fig. 10. Although

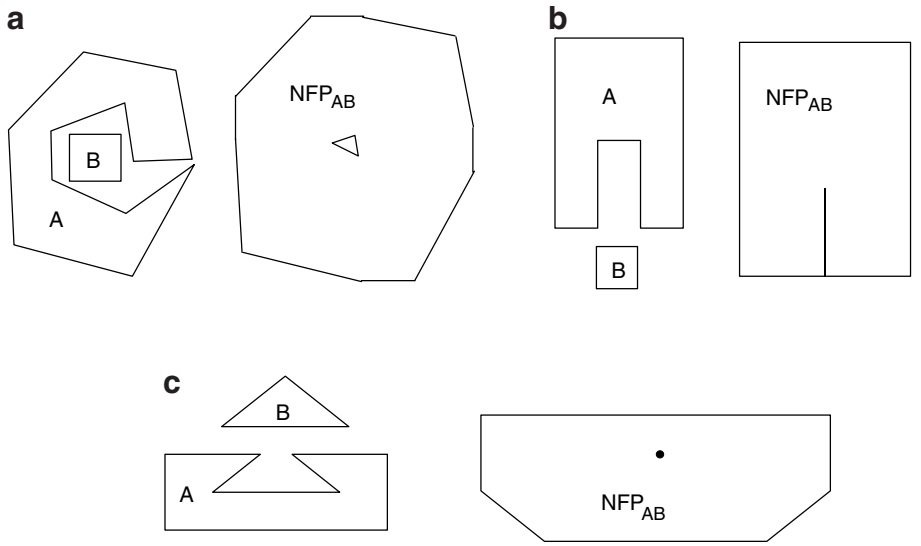


Fig. 10. Combinations of polygons the generate multiply-connected NFPs. (a) NFP where B can fit, with space, inside concavity but cannot slide into the concavity. (b) NFP where B slides into concavity in one direction only and (c) NFP where B can fit at a single point inside concavity but cannot slied into the concavity.

the example in Fig. 10b is not strictly multiply connected, it is included here to illustrate another of the more difficult cases for generating the NFP.

5.2. Sliding algorithm

Mahadevan (1984) proposed a sliding algorithm that models the motion of the tracing polygon around the fixed polygon. In order to ensure the algorithm begins at a point where there is no overlap, the largest y-coordinate of the tracing polygon (B) is placed touching the smallest y-coordinate of the fixed polygon (A). The reference point on B at its starting position defines the first vertex. Each of the following vertices are defined in a counter clockwise direction by identifying which vertex edge combination will slide against each other and the distance available to slide. Both are determined through the use of the D function (discussed in Section 4).

As the direction in which the NFP will be created is predetermined, there exist three possible scenarios for sliding; vertex a_i against edge b_j, b_{j+1}, vertex b_j against edge a_i, a_{i+1} or edge a_i, a_{i+1} against edge b_j, b_{j+1}. These are illustrated in Fig. 11. If the edges have the same slope then the third option is selected, otherwise the edge with the shallowest slope determines the sliding edge. Mahadevan (1984) uses the D-function where the line is the edge of one polygon (for example a_i, a_{i+1}) extended in both directions and the point is the vertex that defines the end of the edge on the other polygon (for example b_{j+1}). The side of the line at which the point is located determines the sliding edge/vertex combination. When there are concavities within either polygon, travelling the full length of the sliding edge may result in overlap between the polygons. To detect such situations and calculate the available sliding distance, each vertex of B is projected the magnitude and direction of the sliding edge (Fig. 12). The D function is used to test for intersections between

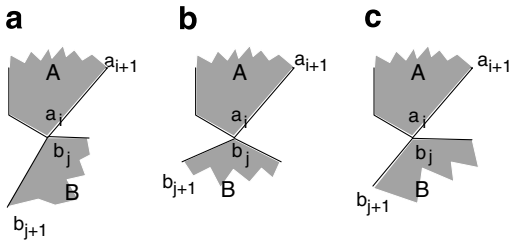


Fig. 11. Scenarios for edge vertex sliding combination.

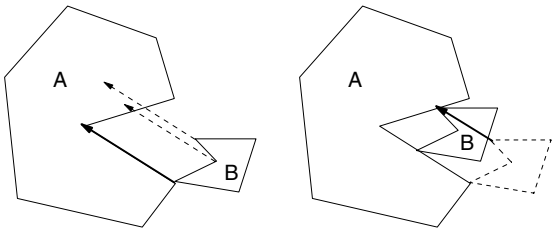


Fig. 12. Sliding edge of A projected from vertices of B to find minimum intersection distance.

the projected edge and the edges of A. The projection is also executed in the opposite direction for the vertices of A and polygon B. For those that intersect, the minimum distance from the original position to the point of intersection is the maximum distance available to move along the sliding edge without overlap.

This approach as described by Mahadevan (1984) only holds for pairs of polygons that are relatively simply connected polygons. The outer boundary of the NFP can be found for instances such as that illustrated in Fig. 10, but not the holes or points of fit. Whitwell (2005) extended Mahadevan's approach in order to address this drawback. The basic premise is that, if an edge of a polygon has not been traversed when executing Mahadevan's approach then these edges potentially represent a hole. If a feasible starting point for the sliding algorithm can be found on the unvisited edges then the equivalent sliding operation, in clockwise direction, can be performed to find the boundary of the hole (or inner-fit polygon).

The first step of Whitewell's orbiting approach essentially follows that of Mahadevan with a modified implementation to speed up the edge-pair detection process. In addition, any edge traversed during this process is flagged. Edges of A and B that are not flagged are then evaluated in the second stage for feasible starting points. There are two elements to this procedure, first is to find an edge-vertex pair that could potentially yield a valid start point, the second is to find a valid start point along the edge.

All edges that are not flagged are considered. Given there is an edge from A that is not flagged, then the vertices from B that can slide along that edge are identified by evaluating whether the edges that connect at that vertex are both on the *right side* of the edge of A. If they are both on the left side or one of the left and one on the right, there is no possibility for feasible sliding between the edge and that vertex. Fig. 13 illustrates this.

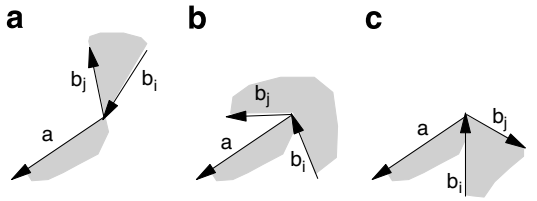


Fig. 13. (a) b_i and b_j on right side of a, (b) b_i on left b_j on right side of a, (c) b_i and b_j on left side of a.

Assuming a vertex from B whose connecting edges are both on the right side is found, then the edge and vertex are placed touching and a test is performed for overlap. If there is overlap, then polygon B is translated along the edge in order to find a feasible start point. The translations are determined through an analogous approach to Mahadevan's method, only here overlap is being detected and resolved rather than avoided. Hence the translation vector derived from the edge being searched is projected from the vertices of B and the closest intersection with an edge of A is recorded, and also from the vertices of A, recording the closest edge intersection with B. Polygon B is then translated through the smallest intersection distance and the position is again tested for overlap. This is done repeatedly until a feasible start point is identified or the entire edge has been searched. Fig. 14 provides an example of this procedure. Once a feasible start point is found, the boundary of the hole is determined by the same orbiting approach as in stage one, flagging edges along the way. All edges that are not flagged are examined in this way for feasible start points. Clearly if the polygons are complex in terms of number of edges and concavities, this process can be quite computationally expensive.

5.3. Minkowski sums

The concept and theory of Minkowski operations comes under the more general field of morphology. The morphological operation that forms the basis of this technique to find the NFP is termed dilation. Dilation grows the image set A through vector addition with set B, and is denoted as A/B. The dilation operator may also be called Minkowski addition defined as follows.

5.3.1. Definition: Minkowski addition in R^2

Let A and B be two arbitrary closed sets of vector points in R^2. S is the resulting sum of adding all vector points in A with those in B. Then the Minkowski sum, S, is defined as

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

The union of geometric translations of sets can also define Minkowski addition. If A_b denotes set A translated by vector b then

$$S = A \oplus B = \bigcup_{b \in B} A_b.$$

The above definition assumes the same orientation for both polygons and as a result will not produce the NFP. However, if polygon B is transposed into its symmetrical set $B' = \{-b : b \in B\}$ then A and B' have the same orientation then A/B' will result in the NFP. Stoyan and Ponomarenko (1977) first formalised the relationship between Minkowski sums and a more rigorous proof can be found in their paper. They termed the NFP as the hodograph. Proof of this relationship is also discussed in Milenkovic et al. (1991) and Bennell (1998) who describe it as the Minkowski difference.

In order to use Minkowski sums and its relationship with the NFP, the realisation of the above definition needs to be formulated into a procedure to obtain the NFP. Ghosh's (1991) develops a set of Boundary Addition Theorems for both the convex case and non-convex case that underpin his method of obtaining the Minkowski sum. These demonstrate that there is sufficient information in the boundary of the polygons to obtain the boundary of the Minkowski sum. The theorems also support the use of *slope diagrams* that form the basis of his approach, for representing the Minkowski sum.

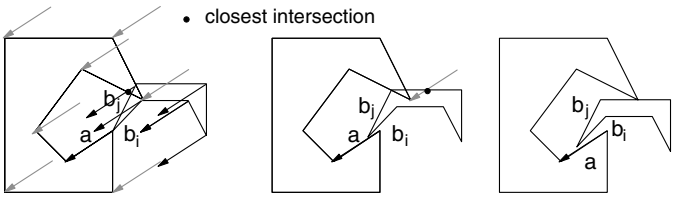


Fig. 14. Whitwell (2005) approach for finding start points.

See Ghosh (1991) for a detailed explanation of these theorems and Bennell (1998) for a discussion of the approach with respect to the NFP.

In order to explain Ghosh's approach a simple example will be used as shown in Fig. 15. Polygon A has one concavity and polygon B is convex. The figure also contains the slope diagram for each polygon, the merged slope diagram for the NFP and the NFP itself. Marking each edge on the diagram according to its slope and orientation creates the slope diagram. B has the opposite orientation to A, hence the edge numbers appear in reverse order when traversing the slope diagram in the counter clockwise direction. Note that the slope diagram for B places the edges in (reverse) numerical order. This is due to the convexity of B. Hence, when both A and B are convex sorting their edges into a single list by slope order will also retain the sequential order of the edges of both polygons. This is equivalent to the approach of Cuninghame-Green (1989). Since A is not convex the edges do not appear in sequential order. However, when visualising the motion of the tracing polygon as it slides around the fixed polygon, intuitively the edges will be encountered in numerical order. As a result the slope diagram is not a circle but modified to traverse the edges in their sequential order.

The NFP is defined by merging the two slope diagrams in which the order of the edges for both polygons must be retained. With reference to the example in Fig. 15, beginning at a_1 we see that b_3 has the same slope and so they are marked together on the slope diagram. Following the counter clockwise direction the next edge is a_2 followed by b_2 and

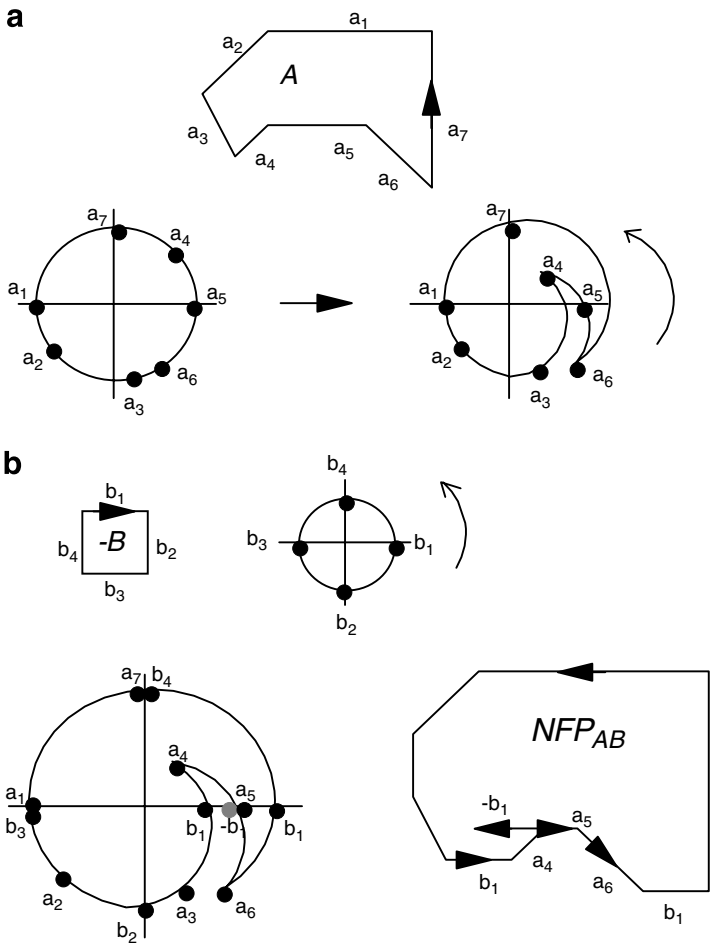


Fig. 15. Calculation of the NFP of one concave and one convex polygon.

a_3. Next a_5 is encountered but cannot be included before a_4. Therefore the traversal goes straight to a_4 passing and including b_1 along the way and then returns to a_5 and on to a_6. However, in order to get from a_4 to a_5 to a_6 b_1 is passed a second time in the opposite direction and so b_1 is included again but with a negative sign. Finally b_1 is included a third time on the traversal between a_6 and a_7 and b_4. The resulting NFP and final merged list is detailed at the bottom of Fig. 15. Note that for each polygon the edges retain their sequential order where changes in direction are permitted. The resulting NFP is not a simple polygon but has some internal edges or loops. This is an inevitable result of the boundary addition theorem and is explained in more detail in Ghosh (1991). These loops need to be examined with respect to their orientation as they may represent holes in a multiply connected polygon. This is dealt with by Ramkumar (1996), Bennell (1998) and Bennell and Song (2005).

The above method can be extended to the case where neither polygon is convex, provided the concavities do not interfere with one another. If, however two concavities interact then there will be areas of the slope diagram where both sets of edges are not in sequential ordered. Ghosh (1991) deals with this by splitting the slope diagram into parallel paths, one for each concavity that interacts, where each path through the slope diagram defines a polygon. The NFP is the union or outer face of these polygons. Although the theory of traversal by parallel paths holds true, there are considerable implementation problems in sorting out paths with more complex instances.

Bennell et al. (2001) propose an approach that extracts key elements of Ghosh's approach and develops a set of algorithmic steps that produce a single path through the slope diagram to produce the NFP. Their approach is based on the observation that the simple-convex case can easily be dealt with by Ghosh's approach. As a result, their approach first replaces B by its convex hull, denoted as conv(B), and solves the simpler case of finding the NFP of a simple and convex polygon, denoted as $\mathrm{NFP}_{A\mathrm{conv}(B)}$. To obtain NFP_{AB} the convex edges that replaced the concavity in the convex hull are substituted with the original edges from the concavity plus any A edges that are traversed wherever they appeared on the slope diagram.

Bennell and Song (2005) illustrate some drawbacks of the Bennell et al. (2001) algorithm, and present some modifications to provide a more robust approach. Instead of generating the convex hull and then repairing the resulting NFP, they propose an approach that retains the concavities of the polygons but partitions one of the polygons into groups of sequential edges according to whether they are convex or concave. Each of the groups can then be individually merged with the slope diagram of A without conflict and then linked. Since the groups are not a complete cycle, the starting edge must be the first B edge in the group. Given the groups will be linked, it is necessary to finish a group moving forward in a counter clockwise direction, equivalent to a positive B edge. Since B edges may appear more than once, the leading B edge must be positive and result in its final appearance also being positive (e.g. $+b_i$, $-b_i$, $+b_i$). When combining the merged lists, linking edges need to be included in order to maintain the precedence order of the edges in each polygon.

Fig. 16, provides an example of their approach. The edge points of B can be divided into the following five groups according to their appearance in consecutive counter clockwise direction (convex) or clockwise direction (concave) on the slope diagram (Fig. 16b).

1. b_{12}, b_1, b_2 (counter clockwise)
2. b_3, b_4 (counter clockwise)
3. b_5, b_6, b_7 (counter clockwise)
4. b_8 (clockwise)
5. b_9, b_{10}, b_{11} (counter clockwise)

For each group the precedence order of A edges is followed, searching for the next B edge in the group. First, by sorting A and the group of B edges into slope order, merging the lists and following the precedence of A, the starting B edge and the number of traversals of each edge can be determined. For example for group 1 (Fig. 16c), begin with the first B edge on the list, b_{12}, at the occurrence that follows a_1. The next B edge is b_1, following the path of A, traverse b_{12}, a_2, a_3, $-b_{12}$, a_4, b_{12}, a_5, a_6, b_1. Note that the admissible B edges that can be included are either $-b_{12}$ or b_1, if other B edges had been encountered on route to b_1, they would have been ignored. This can be observed in other groups. Next, search for b_2, which is encountered directly after b1. Although, b_{12}, b_1 and b_2 have been found, according to the initial count each must be traversed three times, hence the search continues through a_7, $-b_2$, a_8, $-b_1$, a_9, a_{10}, b_1, a_{11}, b_2. In order to link each group, some additional A edges need to be added. For group 1 the

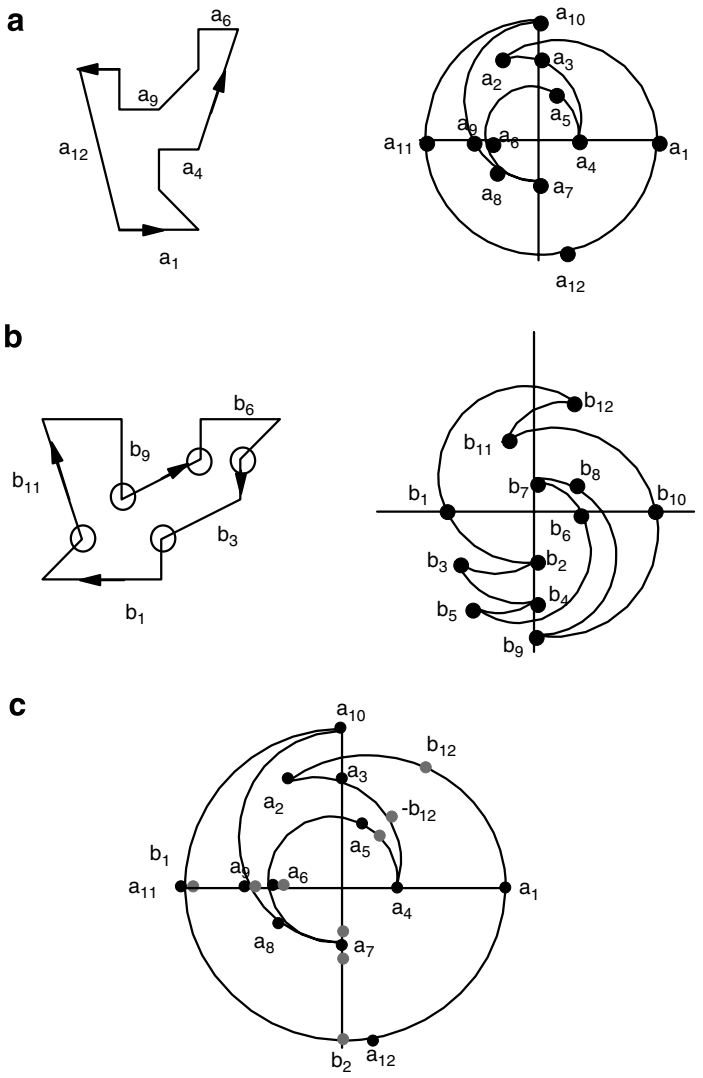


Fig. 16. Example to illustrate Bennell and Song (2005) approach.

final A edge is a_{12} and the first A edge in group 2 is a_7, hence the path returns through $-a_{12}$ to $-a_7$ to retain the precedence order. The full edge list is detailed below, where the A edge that precedes the starting B edge is included in square brackets to indicate the starting point, but is not part of the edge list. The linking edges are underlined.

1. $[a_1]$, $\boldsymbol{b_{12}}$, a_2, a_3, $-\boldsymbol{b_{12}}$, a_4, $\boldsymbol{b_{12}}$, a_5, $\boldsymbol{b_1}$, a_6, $\boldsymbol{b_2}$, a_7, $-\boldsymbol{b_2}$, a_8, $-\boldsymbol{b_1}$, a_9, a_{10}, $\boldsymbol{b_1}$, a_{11}, $\boldsymbol{b_2}$, $\underline{-a_{11}, -a_{10}, -a_9, -a_7}$
2. $[a_6]$, $\boldsymbol{b_3}$, $\boldsymbol{b_4}$, a_7, $-\boldsymbol{b_4}$, a_8, $-\boldsymbol{b_3}$, a_9, a_{10}, a_{11}, $\boldsymbol{b_3}$, $\boldsymbol{b_4}$, $\underline{-a_{11}, -a_{10}, -a_9, -a_7}$
3. $[a_6]$, $\boldsymbol{b_5}$, a_7, $-\boldsymbol{b_5}$, a_8, a_9, a_{10}, a_{11}, $\boldsymbol{b_5}$, a_{12}, $\boldsymbol{b_6}$, a_1, $\boldsymbol{b_7}$, a_2, $-\boldsymbol{b_7}$, a_3, $-\boldsymbol{b_6}$, a_4, $\boldsymbol{b_6}$, a_5, $\boldsymbol{b_7}$, a_6, a_7, a_8, a_9, $-\boldsymbol{b_7}$, a_{10}, $\boldsymbol{b_7}$, $\underline{-a_{10}, -a_9, -a_7, -a_6, -a_5}$
4. $[-a_5]$, $\boldsymbol{b_8}$, $-a_4$, $-\boldsymbol{b_8}$, $-a_3$, $-a_2$, $\boldsymbol{b_8}$, $\underline{a_2, a_3, a_4, a_5}$
5. $[a_6]$, $\boldsymbol{b_9}$, a_7, $-\boldsymbol{b_9}$, a_8, a_9, a_{10}, a_{11}, $\boldsymbol{b_9}$, a_{12}, $\boldsymbol{b_{10}}$, a_1, $\boldsymbol{b_{11}}$, a_2, $-\boldsymbol{b_{11}}$, a_3, $-\boldsymbol{b_{10}}$, a_4, $\boldsymbol{b_{10}}$, a_5, $\boldsymbol{b_{11}}$, a_6, a_7, a_8, a_9, $-\boldsymbol{b_{11}}$, a_{10}, $\boldsymbol{b_{11}}$, $\underline{-a_{10}, -a_9, -a_7, -a_6, -a_5, -a_4, -a_3, -a_2}$

The resulting Minkowski sum is a complex (self crossing) polygon where the edges include all the edges of the NFP and some internal points. The edges that are internal need to be removed. Note that this procedure will find holes and exact fit, as illustrated in Fig. 10, as well as the boundary. Bennell and Song (2005) explain that the negative edges and linking edges cannot be part of the boundary of the NFP and can be removed. As a result the

sequence of edges is broken up into *polygonal trips*.

The next step identities all the intersection points between the polygonal trips, and flags them with a '−', indicating it is *entering* the trip, or with a '+', indicating it is *leaving* the trip. Consider the example in Fig. 17a, where the current trip is trip 2. Imagine standing at the beginning of the first edge of trip 2 facing along the edge, then consider that trip 1 intersects trip 2 from right to left. Trip 1 is said to *enter* trip 2 and is marked with '−'. Continuing along trip 2, eventually intersection with trip 4 is found, where trip 4 intersects from left to right. Trip 4 is *leaving* trip 2 and marked with '+'. Hence, entering a trip means the beginning part of the intersecting edge is on the right side of the trip edge and leaving corresponds to the beginning part of the intersecting edge is on the left side of the trip edge. The fragments of trips that span a '−' intersection to a '+' intersection are kept to form the boundary of the NFP. All other fragments are discarded.

5.4. The origin

The method described above results in the shape and orientation of the NFP but not its position. In order to use the NFP to determine overlap there must exist an origin from which the position of each polygon is measured. This cannot be arbitrarily set but is determined with respect to the origin of the original pieces. Fig. 18 illustrates three different origins of the NFP with respect to different origins on the original polygons, where the origin of the tracing polygon, B, is the reference point that traces the NFP.

For an arbitrary position of the origin of A and reference point of B, the origin of the NFP can be determined by placing A and B touching such that a vertex of B will slide along an edge of A, edge a_1 in Fig. 18. The roles of each polygon is important since edges of B are replicated in the NFP in their opposite orientation. Edge a_1 can be found in the NFP and represents the path of the reference point

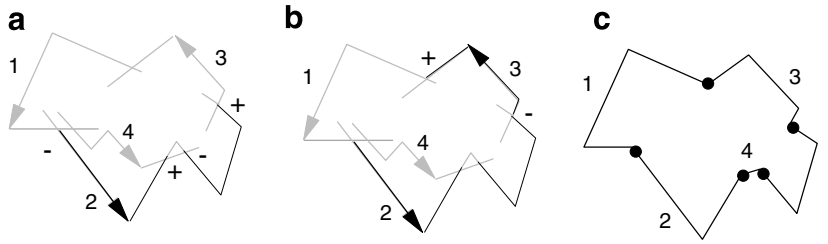


Fig. 17. Procedure for removing internal edges from the Minkowski sum. (a) Identify each intersection as entering or leaving the trip, (b) truncate each trip retaining parts between '−' and '+', (c) repeat and link parts.

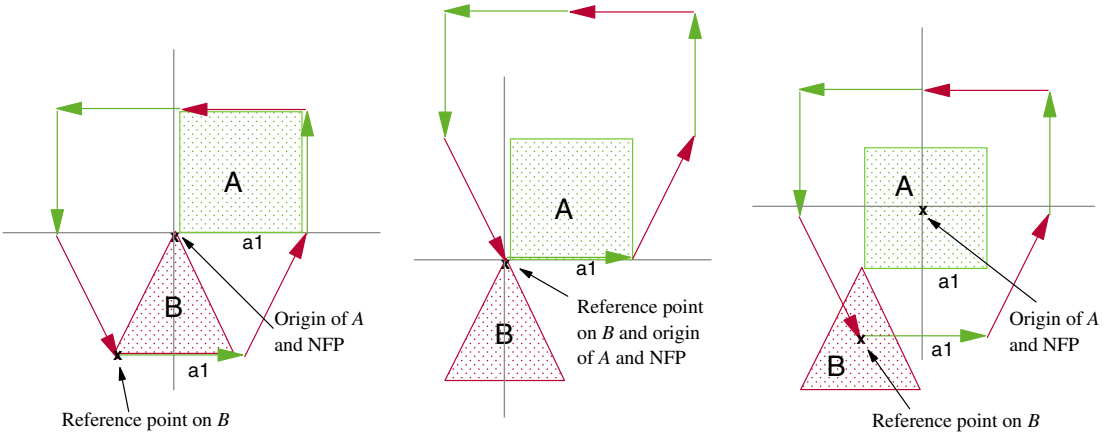


Fig. 18. Different origins of the same NFP determined by the origin of A and B.

as B slides along edge a_1. By positioning A and B touching at the start of a_1 and aligning the start of a_1 in the NFP at the reference point of B, the origin of the NFP is given by the origin of A.

In the case of assigning the bottom left corner of the enclosing rectangle of both component polygons as the origin, then the origin of the NFP is found at the top right corner of the enclosing rectangle of the tracing polygon if it is placed at the bottom left corner of the enclosing rectangle of the NFP. The theory that underpins this can be found in Bennell (1998).

5.5. Decomposition

Given the complexity of obtaining the NFP when concavities between polygons interact, an alternative approach is to decompose them into a number of more manageable shapes. Watson and Tobias (1999) and Agarwal et al. (2002) decompose simple polygons into convex sub-polygons. As described by Cuninghame-Green (1989) the NFP of two convex polygons can be calculated quickly and easily and is also a convex polygon. Li and Milenkovic (1995) decompose into star shaped polygons. A star shaped polygon is one that contains at least one point, called the kernel, such that a line drawn between that point and any point on the boundary is wholly contained within the polygon. Li and Milenkovic show that the Minkowski difference of two star shaped polygons is also a star shaped polygon. Both decomposition methods are attractive as they remove the requirement of detecting holes when calculating the NFP of each pair of decomposed parts of the polygons. Here we will focus on convex decomposition.

The principle of these approaches is to decompose each polygon into convex sub-polygons, generate the NFP of each pair of sub-polygons where the members of the pair arises from different polygons, and finally combine the NFPs of the sub-polygons to generate the NFP of the two original polygons. The approach can be illustrated by considering Watson and Tobias (1999). They decompose a simple polygon into a set of convex polygons by cutting between pairs of concave vertices. If there are an odd number of concave vertices then the last cut is between a concave and a convex vertex. The NFP of each sub-polygon can be found by the simple edge sorting procedure described earlier in the paper. The final step is to combine the sub-NFPs into one polygon. Such a procedure requires the identification of all edge intersections, since these

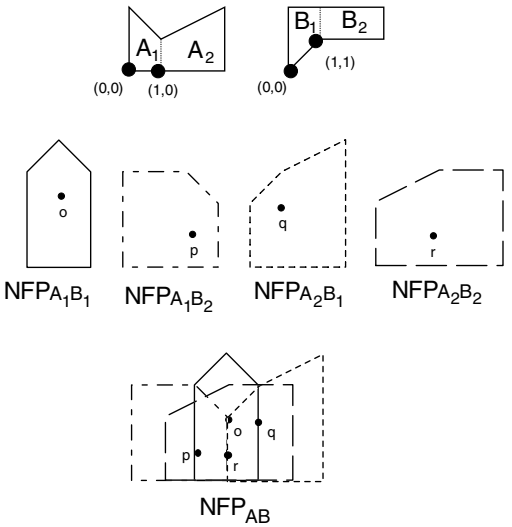


Fig. 19. Calculation of the NFP using convex decomposition.

may form vertices of the final NFP, and removal of the redundant edges (i.e. edges contained inside one or more sub-NFPs and therefore not part of the boundary). An example of how two concave polygons might be decomposed into convex sub-polygons and reconstructed into the NFP is given in Fig. 19.

An important feature to note with such an approach is arranging the origins of each sub-NFP according to the relative origins of the decomposed parts in order to form the NFP. In the example given in Fig. 19 both polygons have a single concave vertex and are thus decomposed into two convex polygons. Each sub polygon has an origin at its bottom left corner, marked on the figure by a dot. Hence the origin of the sub-NFPs is found at the top right corner of B_i as described in the previous section. Fig. 19 illustrates each sub-NFP (NFP_{A1B1}, NFP_{A1B2}, NFP_{A2B1}, NFP_{A2B2}) and their origins marked as o, p, q and r respectively. When reconstructing the sub-NFPs the position of their origin is determined by the vector difference $a - b$ of the origins. In the example the origin of A1 and B1 is at $(0,0)$ and therefore the origin o of NFP_{A1B1} is at (0,0) i.e. equivalent to the origin of the NFP. $B2$ is at (1,1) and so p is placed at position $(-1, -1)$ relative to the origin on the NFP. $A2$ is at (1,0) and so q is placed at position (1,0) and r is placed at position $(0, -1)$.

Clearly the advantage of this approach is found in removing the difficulty of the NFP generation operation. However, the price of this simplification

is the complexity of the decomposition and combining phases. Also, the effectiveness of the decomposition phase has a direct effect on the efficiency of the combining phase. For example, if A is decomposed into p sub-polygons and B decomposed into q sub-polygons then $p \times q$ sub-NFPs need to be found and assembled in the correct way to form NFP_{AB}. It stands to reason that the fewer sub-polygons required to decompose the polygon, the fewer sub-NFPs are generated and as a result the fewer edges need to be analysed for intersection and inclusion in the combining process. However, decomposing a polygon in to the minimum number of convex polygons is significantly more computationally expensive than greedy methods. Agarwal et al. (2002) evaluate this issue further by experimenting with a range of different decomposition methods including triangulations, optimal decomposition and heuristics, in order to examine the effect on computational time. They found that although optimal decomposition in general significantly reduces the computation time for the Minkowski sum construction, the decomposition is computationally expensive and heuristics that approximate optimal decomposition are preferred. In addition they discovered that the relationship between the number of convex sub-polygons and the time to construct the final NFP was not a straight forward trade-off. Experimentation demonstrated that the relationship between the two polygons to be summed impacts on the Minkowski sum construction. Hence, they decompose the polygons simultaneously in order to minimise a cost function that takes account of this relationship. Other issues that arise from the construction phase are the identification of degenerate cases as illustrated in Fig. 10b and c. Case 10b would be represented by co-linear edges not contained in any sub-NFP, and 10c would be found at an intersection point of more than two edges not contained in any sub-NFP. These instances would all need to be tested directly.

5.5.1. Summary

The NFP is an efficient tool for detecting the relative position between pieces. However, calculating the NFP is a non-trivial task and software to do this is not publicly available. Hence many researchers do not exploit the benefits of the NFP due to the significant investment in developing the tools required. In addition all the approaches above have their limitations. Mahadevan's approach can not identify nesting positions that would result in a multiply connected NFP; Ghosh's approach overcomes this but can become complex resulting in many internal loops. In addition, instances where one piece fits exactly into a nested concavity would be defined as a point or line within the NFP. Mechanisms to identify the orientation of this type of "loop" largely resort to direct testing (Bennell and Song, 2005; Whitwell, 2005).

6. Phi function

The phi-function is the most recent innovation in dealing with the geometric issues for nesting problems. Its purpose is to represent all mutual positions of two objects (in this context polygons), and therefore it is often associated with the NFP. However, this association is misleading as the NFP is only a special case of the broader theory. The Phi-function for cutting and packing were conceived and applied by Stoyan et al. (2001, 2004) and this research group continue to be the main users of this methodology. The lack of an algorithmic process for generating the phi-function for arbitrary shapes may explain why this approach has not been more broadly adopted. However, the phi-function is a powerful tool that warrants further research in order to facilitate its use in the wider research community. In this section we will simply illustrate the concept and its application. See the original papers for detailed derivation of the mathematical functions.

The phi-function is a mathematical expressions that represent the mutual positions of two objects. Specifically the value of the phi-function is greater than zero if the objects are separated, equal to zero if their boundaries touch and less than zero if they overlap. When the phi-function is normalised its value is the Euclidean distance between the two objects, otherwise it is an estimate of the distance. Stoyan et al. derive the phi-function for primary objects; these are circles, rectangles, regular polygons, convex polygons and the compliment of these shapes. Shapes that are not primary objects can be represented as a union or intersection of the primary objects. Note that this is not decomposition since overlap of primary objects is permitted. Hence we will focus here on the phi-function for primary objects. The derivation of the functions is based on trigonometry, and as far as we are aware, they are derived by hand.

As an example we will consider the case of two circles. Intuitively the centre of the circles must be at least the sum of the radii apart for the circles to

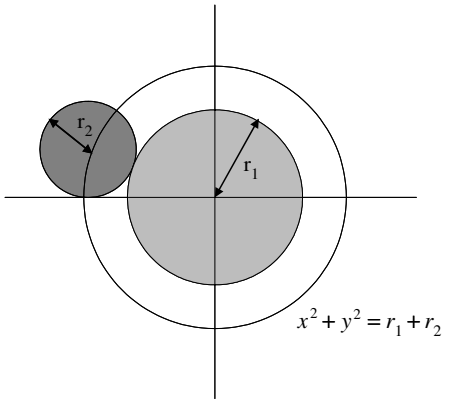


Fig. 20. Representation of all touching positions of two circles.

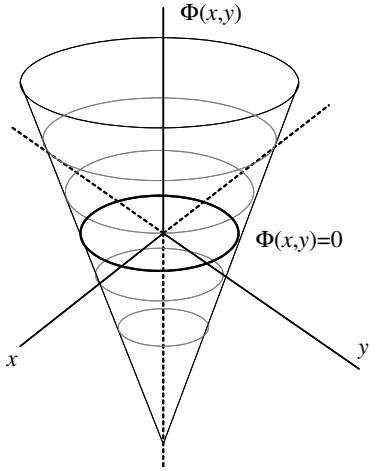


Fig. 21. Plot of the phi-function for two circles.

not overlap. This is illustrated in Fig. 20 where circle 1 as radius r_1 and circle 2 has radius r_2. Assuming circle 1 is fixed at co-ordinate position (0,0), the equation $\sqrt{x^2+y^2} = r_1 + r_2$ describes all the co-ordinate positions of circle 2 (x,y) such that the two circles touch but do not overlap. Clearly if the circles were separated then $\sqrt{x^2+y^2} > r_1 + r_2$, and the difference would be the Euclidean distance between the circles. Hence the phi-function can be directly stated as: $\Phi(x,y) = \sqrt{x^2+y^2} - (r_1+r_2)$. Finally if we remove the assumption that circle 1 is fixed at (0,0) but instead is placed at co-ordinate point (x_1, y_1) and circle 2 is placed at co-ordinate point (x_2,y_2), then we need to translate both circles through $(-x_1, -y_1)$ so that circle 1 is again at the origin and circle 2 has the equivalent relative position to circle 1. Therefore we can state the phi-function for two circles at arbitrary positions as follows:

$$\Phi(x_1,y_1;x_2,y_2) = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} - (r_1+r_2)$$

Fig. 20 illustrates the phi-function only for the instance when $\Phi(x,y) = 0$, where $x = x_2 - x_1$, $y = y_2 - y_1$, which is also the NFP. We can also plot the value of $\Phi(x,y)$ over all possible values of (x,y) on a three dimensional graph, as in Fig. 21.

Two circles is the simplest case since the phi-function can be derived from a single function. Other primary objects require a series of functions where the function that should be used to obtain the value of the phi-function dominates the other functions through being either the maximum or minimum. For example, the NFP of two rectangles is simply a rectangle, and mathematically each side of the rectangle is part of the phi-function. The relevant function depends on whether the moving rectangle is above, below, to the right or to the left of the fixed rectangle. If each edge of the rectangle and the position of the orbiting rectangle defines an equation where its value is positive if the orbiting rectangle is on the side of the line that is outside and negative when it is on the side that is inside, then the maximum of these four functions will give the phi-function. However, as the rectangles move apart a new region of the phi-function emerges, where the moving rectangle is not purely to the side or purely above or below. In these regions a new set of functions are required. This is illustrated in Fig. 22 where the NFP or $\Phi(x,y) = 0$ is a rectangle and $\Phi(x,y) > 0$ is a rectangle with curved corners.

Stoyan et al. (2001) describe how to derive the phi function for not only all the primary objects but also the complement of the primary objects, which is equivalent to the containment problem.

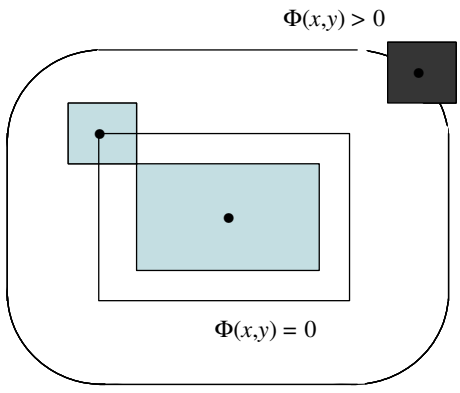


Fig. 22. Two instances of the Phi-function.

This concept appears to have great potential for contributing to the advancement of the field of nesting problems. However, the process of obtaining the phi-function has thus far acted as a barrier to wider adoption of this approach.

7. Conclusion

The paper has given a detailed description of the operations of a number of different approaches for dealing with the geometry required in the solution of nesting problems, these are; the raster method, direct trigonometry, the Nofit polygon and phi-functions. Each has its advantages and limitations and a common theme is that the more computationally efficient the approach the more complex it is to realise. There is no doubt that there is a significant investment required in order to develop the geometric tools necessary for tackling nesting problems. However, we hope that through this paper some of the mystery of these approaches has been dispelled and researchers are able to make an informed choice when selecting their methodology.

Acknowledgement

The authors would like to sincerely thank Dr. Kathryn Dowsland for reviewing this manuscript and providing valuable feedback.

References

Agarwal, P.K., Flato, E., Halperin, D., 2002. Polygon decomposition for efficient construction of Minkowski sums. Computational Geometry Theory and Applications 21, 29–61.

Babu, A.R., Babu, N.R., 2001. A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. Computer-Aided Design 33, 879–891.

Beasley, J.E., 1985. Bounds for 2-dimensional cutting. Journal of the Operational Research Society 36, 71–74.

Bennell, J.A., 1998. Incorporating problem specific knowledge into a local search framework for the irregular shape packing problem, PhD thesis, University of Wales, UK.

Bennell, J.A., Song, X., 2005. A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums, Working paper series, Centre for Operational Research, Management Science and Information Systems, University of Southampton, UK.

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

Cuninghame-Green, R., 1989. Geometry, shoemaking and the milk tray problem. New Scientist 1677 (12 August), 50–53.

Dyckhoff, H., 1990. A typology of cutting and packing problems. European Journal of Operational Research 44, 145–159.

Ferreira, J.C., Alves, J.C., Albuquerque, C., Oliveira, J.F., Ferreira, J.S., Matos, J.S., 1998, A Flexible Custom Computing Machine for Nesting Problems. In: Proceedings of XIII DCIS, Madrid, Spain.

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

Konopasek, M., 1981. Mathematical Treatments of Some Apparel Marking and Cutting Problems, U.S. Department of Commerce Report 99-26-90857-10.

Letchford, A.N., Amaral, A., 2001. Analysis of upper bounds for the pallet loading problem. European Journal of Operational Research 132, 582–593.

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

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

Milenkovic, V., Daniels, K., Li, Z., 1991. Automatic marker making. In: Proceedings of the Third Canadian Conference on Computational Geometry, Simon Fraser University, Vancouver, BC, pp. 243–246.

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

Preparata, F.P., Shamos, M.I., 1985. Computational Geometry: An Introduction. Springer-Verlag.

Ramkumar, G.D., 1996. An algorithm to compute the Minkowski sum outer-face of two simple polygons. In: Proceedings of the 12th Annual Symposium on Computational Geometry FCRC 96.

Segenreich, S.A., Braga, L.M., 1986. Optimal nesting of general plane figures: a Monte Carlo heuristical approach. Computers & Graphics 10, 229–237.

Stoyan, Y.G., Ponomarenko, L.D., 1977. Minkowski sum and hodograph of the dense placement vector function, Reports of the SSR Academy of Science, SER.A 10.

Stoyan, Y.G., Terno, J., Scheithauer, G., Gil, N., Romanova, T., 2001. Phi-functions for primary 2D-objects. Studia Informatica Universalis 2 (1), 1–32.

Stoyan, Y., Scheithauer, G., Gil, N., Romanova, T., 2004. Φ-functions for complex 2D-objects. 4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies 2, 69–84.

Watson, P.D., Tobias, A.M., 1999. An efficient algorithm for the regular W_1 packing of polygons in the infinite plane. Journal of the Operational Research Society 50 (10), 1054–1062.

Waescher, G., Haussner, H., Schumann, H., 2005. An improved typology of cutting and packing problems, European Journal of Operations Research, forthcoming.

Whitwell, G., 2005. PhD. thesis, School of Computer Sciences, University of Nottingham, UK.

Y, G.G., Kang, M.K., 2002. A new upper bound for unconstrained two-dimensional cutting and packing. Journal of the Operational Research Society 53 (5), 587–591.