The Industrial Application of New Irregular Cutting and Packing Algorithms

A Thesis submitted to the University of Nottingham for the Degree of Doctor of Philosophy

by

Robert Hellier, BSc (Hons)

University of Nottingham, Nottingham

School of Computer Science and Information Technology

February 2013

Contents

Figures

Tables

Algorithms

Abstract

This thesis investigates the field of irregular two-dimensional stock cutting from the perspective of an industrial practitioner. The new approaches and developments in the thesis have been motivated by industrial considerations and have been used directly in industrial software applications.

Irregular two-dimensional stock cutting problems occur in a wide range of industries and in almost all cases, for reasons of effective use of staff time, solution speed and efficiency, the application of the automated nesting algorithms are advantageous over manual methods. Reduction in the use of raw materials is a direct cost saving for any business and therefore has a significant impact on operating profitability.

The approaches developed in this thesis have, at the time of publication, produced the best-known solutions for all of the 26 known irregular problem instances from the scientific literature. In order to explore some of the unique features of the approaches, motivated by industrial concerns, the thesis also introduces additional benchmark problems.

In order to achieve these high quality solutions and attain the level of reliability required in industrial applications this work introduces a complete and robust technique for the production of no-fit polygons for irregular shapes including arcs, interlocking concavities and holes. The robustness of the technique and its ability to handle arcs and holes make this the first algorithm in the literature not to suffer from degenerate cases and makes it highly valuable to industrial practitioners, as well as a valuable tool for research scientists.

The placement algorithms presented in this work take advantage of the geometry of the shapes being placed in order to produce solutions very rapidly and to a high degree of accuracy. These placement techniques, in combination with numerous local search techniques, achieve high quality solutions for a wide range of problems. Indeed these techniques are being used in industrial settings worldwide today on a vast range of problems and in numerous industrial sectors.

Acknowledgements

I would like to take this opportunity to thank my co-researcher, and now colleague at Aptia Solutions Ltd., Dr. Glenn Whitwell with whom I have thoroughly enjoyed working on this research and on all of our subsequent endeavours.

To Professor Graham Kendall my supervisor on the Teaching Company Scheme project and thereafter on my part time PhD studies. His encouragement, help, advice and support has been invaluable over the last 10 years of my involvement with ASAP as both staff and student.

To Professor Edmund Burke my co-supervisor on the Teaching Company Scheme project.

To my examiners, Professor Julia Bennell and Dr. Ender Özcan, for their comments that have helped to significantly improve this thesis.

To my family and friends who have helped to keep my focus and belief over such a long period of research and study.

I would like to express my gratitude for the support of the Teaching Company Scheme and the Engineering and Physical Sciences Research Council (EPSRC) who, along with Esprit Automation Ltd., supported the projects that lead to this research being undertaken.

I offer special thanks to the directors and staff, past and present, of Esprit Automation Ltd. for whom I worked full time for the vast majority of the past 10 years and who instigated and sponsored the CASE / TCS projects.

This work is dedicated to my father Stephen and my grandfather James.

CHAPTER ONE

1. Introduction

Cutting and packing problems have been found in various industries for many decades. There are numerous variants of the problem to be found and useful typologies have been presented to help the researcher to delineate between the problem subtypes, notably (Dyckhoff, 1990) and (Wäscher et al., 2007). All the problems types are concerned with minimising the waste of material or maximising usage of material or space. The many of the variants of the cutting and packing problem set have been shown to be NP complete (Fowler et al., 1981) including the Open Dimensional Problem (ODP) , which this thesis tackles, the OPD is defined in the (Wäscher et al., 2007) typology. The ODP with two-dimensional irregular shapes is described as the packing of an arbitrary set of small items into a larger container with one variable dimension with the aim of minimising the space used within the larger container.

The Open Dimensional Problem with irregular two dimensional shapes variant of the cutting and packing problem impacts upon several important manufacturing industries such as textiles, plastics, metal cutting and others. These problems usually consist of a number of irregular pieces that are to be placed onto one or more sheets of material in the most efficient layout possible, such that all pieces are assigned and do not overlap. Additionally, there are usually rotational constraints enforced on the pieces due to the physical properties of the problem such as grain on the material, patterns on textiles and the cutting technology being employed. Sometimes rotational constraints may be used for non-physical reasons such as to restrict pieces to a finite set of rotations thus simplifying layout construction procedures and allowing for faster solutions to be obtained. Cutting and packing algorithms can also be applied to other spatial problems such as floor plan layouts. The main objectives are to maximise space utilisation and minimise the computation time required. In addition to these fundamental objectives there are often industry specific requirements which are normally dictated by the material to be cut, the cutting method and the required cut quality. For example, within the textile industry, consideration must be given to the weave of the cloth and printed designs which is in contrast to the sheet metal industry where heat distribution and material warping are of concern because plasma or oxy-fuel cutting processes are being employed. Research into automatic packing approaches has steadily increased over the years due partly to industrial competitiveness but also to greater academic interest from the scientific community (Sweeny & Paternoster, 1992).

1.1. Background and Motivation

The work presented in this thesis was generated as part of two intertwined projects a Collaborate Awards in Science and Engineering (CASE) grant and a Teaching Company Scheme (TCS), now known as a Knowledge Transfer Partnership (KTP). At the heart of both of these projects was an industrial partner, Esprit Automation Ltd., they wished to improve the automatic layout algorithms in their Computer Aided Design (CAD) /Computer Aided Manufacturing (CAM) product Procut.

My role in the two projects was as a TCS associate for 3 years. During this period I began to study for this thesis part time and the majority of the work presented in this thesis was developed over the course of that program. My co-researcher, on the CASE award, was Glenn Whitwell with whom I worked very closely on all aspects of the research, more often than not we sat at the same computer and whiteboard working our way through problems, generating ideas and coding the solutions, as was the intension of linking the projects at the outset.

At the end of the TCS project I was offered a full time software development position at Esprit. I took up this offer and over the course of the next seven years I was gradually promoted to the Head of Software Development with responsibility for a team of developers and all of their software products. I held the position of Head of Software from 2006 until 2010.

As the software lead I was responsible for both the Procut CAD/CAM tool and specialised Enterprise Resource Planning (ERP) software, called ProManager, which was specifically designed to work for metal cutting companies. ProManager was built from scratch during my tenure at Esprit. The software is able to aid its users with the full lifecycle of metal cutting order fulfilment from drawing creation (via Procut) raw material and services ordering, quotation and order generation, part placement (via the Nesting module of Procut), job cutting and post-processing, dispatch and invoice.

Such a long tenure overseeing the development of Procut and such wide-ranging product like ProManager has given me a great deal of experience in the practical concerns of companies who use plasma, laser and oxy-fuelled metal cutting machines. Many of these concerns often influence the composition and priority of layouts to be produced to fulfil orders, i.e. additional or pre-processes, deadlines and transport options.

In September 2010, just one month before the 10th anniversary of first working on the TCS project, I left Esprit Automation to work for Aptia Solutions Limited as their Chief Technology Officer. Aptia is a University of Nottingham spin-out company specifically set up to exploit the intellectual property outlined in this thesis. They have thus far specialised in knife cutting applications and my extensive experience in a related field and commercial software development has helped the company move on rapidly since I took up the role.

1.2. Aims and Scope

At the heart of the brief for the TCS project, and to some degree the CASE award, was the imperative that that research developed on the projects would be applicable to the Esprit Automation Ltd. range of CAD/CAM products.

The implication of this was that the work developed on the projects had to:

- Deliver results *quickly*, i.e. in real time in front of a waiting end user
- Be able to cope with *any* shape the user wished to nest
- Deliver *improved* results on the existing layout solution methodology already in the Esprit products
- Produce reliable and accurate layouts without overlap or excessive waste
- To be compatible with the Procut suite of software (a MFC C++ application)

To achieve improved results is obviously the most important of the elements of the brief outlined above. Therefore it was determined that the programmes should attempt to use the fundamental geometry of the shapes being placed. The Esprit algorithms, at the start of the CASE / TCS project, were based on a bitmap representation and, as a result, were inefficient from the point of view of nesting as a result of the inaccuracy in this type of representation when used in nesting applications. Additionally the Esprit nesting process was a single pass algorithm, with length or area based orderings available as user interface options. To improve on this it was determined that the projects should introduce meta-heuristic search mechanisms into the automatic nesting process.

All of the work presented in the thesis is motivated by these initial decisions; the use of fundamental geometry during the layout process and the introduction of meta-heuristic search routines. Other factors, such as solution speed and generality with respect to input shapes have remained the overall aims throughout the TCS / CASE projects and also underpin this thesis.

1.3. Overview of Thesis

The thesis is structured in the following manner:

Chapter two presents a literature review of the irregular two-dimensional packing field.

Chapter three describes the various methodologies that are used in the later chapters to develop irregular packing approaches, including search methods, geometric techniques and the no-fit polygon.

Chapter four introduces a new bottom-left-fill algorithm that was developed to improve on the bitmap based implementation that Esprit Automation Ltd. were using when the TCS and CASE research programmes were conceived. This approach improved on many literature benchmarks and was one of the first to be tested against a full range of problems from literature.

Chapter five describes the work performed to produce a fast and highly robust no-fit polygon generation algorithm. This was the first no-fit polygon generation algorithm ever published to overcome all of the known degenerate cases.

Chapter six develops the work presented in chapter five. The work presented involves the extension the no-fit polygon algorithms to include arcs, in addition to lines, whilst remaining able to cope with all of the known degenerate cases found in other methodologies. Experimental results are presented and show that, in combination with the new bottom-left fill heuristic presented in chapter four, the no-fit polygon drastically increases the performance of the layout algorithm. This increased performance increases the ability of the heuristic to improve on the literature benchmarks.

Chapter seven presents a simple non-greedy placement mechanism that utilises the line and arc no-fit polygon generation technique of chapter five to explore the effectiveness of non-greedy placements during layout generation. The placement approach utilised in this chapter avoids inaccuracies inherent in the placement approach of chapters four and six.

Chapter eight presents a case study of Aptia Solutions, a spin out company formed to exploit the research presented in this thesis, including work on specific industry problems from the composites industry.

Chapter nine presents the conclusions drawn from the thesis and outlines possible future avenues of research, including practical manufacturing considerations across industry sectors.

1.4. Contributions

The work presented in this thesis has made several contributions to the field of irregular cutting and packing. The following list briefly outlines each contribution and refers to later sections in which it is described and later summarised:

- The development of a new placement technique for irregular packing (For description see section 4. For discussion see section 9.1)

- Presentation of the first complete and robust no-fit polygon generation algorithm for non-convex polygons (For description see section 5. For discussion see section 9.2)

- Extension of complete and robust no-fit polygon algorithm to allow the generation from shapes including arc segments (For description see section 6. For discussion see section 9.3)

- Repeated generation of new benchmark solutions for 25 of the 26 existing literature problems (For description see section 4.13, 6.6, 7.7. For discussion see section 9.5)

- Introduction of new benchmark problems to the field literature including new features such as holes and arc segments (For description see section 4.9. For discussion see section 9.4)

- Introduction of a new simple non-greedy placement mechanism that produces improved benchmark results for this thesis. (For description see section 7.6. For discussion see section 9.6)

1.5. Publications

The research presented in this thesis has produced the following publications.

The work presented in chapter 4 was published in:

- E. K. Burke, R. S. R. Hellier, G. Kendall, and G Whitwell, "A New Bottom-Left-Fill Heuristic Algorithm for the Two-Dimensional Irregular Packing Problem," ***Operations Research***, vol. 54, no. 3, pp. 587-601, 2006.

The technique shown in chapter 5 was published in:

- E. K. Burke, R.S.R. Hellier, G Kendall, and G. Whitwell, "Complete and Robust No-Fit Polygon Generation for the Irregular Stock Cutting Problem," ***European Journal of Operations Research***, vol. 179, no. 1, pp. 27-49, 2007.

The approach outlined in chapter 6 was published in:

- E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell, "Irregular Packing Using the Line and Arc No-Fit Polygon," ***Operations Research***, vol. 58, no. 4-Part-1, pp. 948-970, 2010.

CHAPTER TWO

2. Stock Cutting Problems

2.1. Introduction

Numerous industrial processes require the sub-division of raw stock materials into sub-pieces. Given the varying material types and cutting processes, researchers have developed numerous automated techniques for the division of one and two dimensional stock materials. Traditionally these problems have been addressed by human solvers but the advent of powerful computing platforms in the 1960s and their subsequent quick adoption in industrial settings as described in (Hatvany, 1984), (Goldhar & Jelinek, 1985) and (Pennings, 1987) has allowed the solution generation for problems of this kind to be carried out by automated methodologies.

This chapter provides an overview of the field of irregular two-dimensional cutting and packing by following the field's progression from orthogonal focused early work through various approaches that have used some form of geometric approximation to avoid the complexities of dealing with full two-dimensional geometry and on to those techniques that are able to cope with the full two-dimensional geometric representations.

Techniques that use a true shape sequence based layout generation approach are given attention in section 2.5 as these are strongly related to the work presented in chapters 4, 6 and 7 of this thesis. Approaches using full geometric representations are discussed including those methods which allow for placement based or layout improvement based layout generation rather than sequence based solution generation techniques similar to the technique developed in chapter four of this thesis. Of additional focus in this chapter techniques for generating layouts of single parts and finally those techniques that have allowed for circular arc representations to be included in their problem sets, rather than linear approximations, which is a major feature of the work presented in chapters four and five of this thesis. The benchmark problems used throughout this thesis are presented in section 2.8.

Following this, chapter three focuses upon approaches from the scientific literature relating to the geometric techniques used in the development of new work in this thesis. A section on the literature related to geometry and no-fit polygon is presented in section 3.1 and 3.2 respectively and the literature relating to the search heuristics used to navigate the solution space is discussed in section 3.5.

2.2. Typological Categorisation

Due to wide range of applicability of cutting and packing techniques, i.e. fields where the problem is relevant include computer science, engineering, manufacturing and operational research, many of the same problem types have developed different names in the various fields, for example bin-packing, trim loss problem and one-dimensional packing all refer to the same problem type.

In (Dyckhoff, 1990) the author presented a typology for the wide range of cutting and packing problems that existed in the literature. The author noted that packing problems are divided by dimensionality, specifically those problems that involve some spatial dimensions and those that involved non-spatial dimensions.

The problems involving spatial dimensions (up to three dimensions in Euclidean space) are regarded as packing and loading problems and those without spatial dimensions include temporal and other dimensions such as computer memory blocks. Problems with Euclidean dimensions are then sub-divided into cutting and packing problems, cutting which involved a large object which is to be cut into smaller objects and packing problems which involve filling bins with a number of shapes.

In (Wäscher et al., 2004) the authors pointed out several drawbacks of the proposed Dyckhoff typology.

Importantly not all cutting and packing problems are uniquely defined by the typology's criterion, for example the vehicle loading problem can be categorised using the Dyckhoff typology as both:

- 1-dimensional | resources accommodate all items | many identical resources | *many different items*
- 1-dimensional | resources accommodate all items | many identical resources |*few different items*

Additionally, as noted in (Gradisar et al., 2002), different problems can be classified in the same category – for instance the strip packing problem and the bin packing problem fall in the same categorisation.

In (Wäscher et al., 2007) the authors presented an improved typology for the range of cutting and packing problems. Although partially based on the Dyckhoff typology the authors introduced new categorisation criteria that differ from the original. Additionally the newer typology introduced a more consistent naming scheme and the authors demonstrated its usefulness by categorising the published cutting and packing literature for the years 1995 to 2004. This new topology is widely accepted as a useful addition to the cutting and packing literature and has already been cited 256 times (Google Scholar 16/12/10).

According to the Wäscher, Haußner and Schumann typology the problems that are used to benchmark the new methods introduced in this thesis fall into the category of the Open Dimensional Problem (ODP) as the problems display the follow attributes:

- One large object of variable dimensions for smaller items to be placed in
- Smaller items to be placed are irregular and strongly heterogeneous

2.3. Initial Approaches

Early approaches to cutting and packing problems were constrained by the limited computational power of the computer equipment of the time. However by extended techniques developed for orthogonal packing several authors were able to produce useful techniques that could produce layouts within acceptable run times.

Amongst the earliest pioneers in the cutting and packing field were Gilmore and Gomory who in 1961 conducted some of the first research in the area and solved one dimensional problems to optimality using linear programming (Gilmore & Gomory, 1961). The size of the problems that were able to be tackled was limited by the long computation times required. In 1963 the authors used a column generation technique to restrict the number of possible solutions (Gilmore & Gomory, 1963). This technique was applied to real problems from a paper roll cutting factory. This was the first time that stock cutting techniques had been applied to real world problems.

In 1965 Gilmore and Gomory extended their linear programming approach to work on the orthogonal variation of the problem (Gilmore & Gomory, 1965). Their technique was able to generate layouts applicable to guillotine cuts; this restriction allowed a reduction in complexity of the problem and was a good fit given the nature of the paper problems they attempted to solve. In 1966 the authors used a dynamic programming technique to tackle the knapsack problem (Gilmore & Gomory, 1966). The four papers by Gilmore and Gomory noted above are widely regarded as the seminal cutting and packing works, indeed they are the most widely cited papers in the relevant literature.

An exact approach, simpler than Gilmore and Gomory's was introduced by (Barnett & Kynch, 1967). Their technique allowed non-guillotine layouts to be generated; indeed this was a requirement of the relaxation used to produce this simpler approach. The placement algorithm used to place rectangles was based on removing the two opposite corners of a chess board and then attempting to place dominos over the remaining uncovered board area.

Industrial considerations were taken into account by Haessler in (Haessler, 1971) and (Haessler, 1975) when the author tackled the problem of rectangle placement whilst taking into account the potential cost of setting up machines to produce a layout. An optimal layout may require a machine to setup several times which has an associated cost, which could outweigh the benefits of a more efficient layout. The author formulated the problem as a mathematical programming model utilising a heuristic technique, first presented in (Haessler, 1968), to generate solutions.

The very first presentation of a bottom-left heuristic, a technique very widely used in two dimensional irregular packing, for the placement of orthogonal parts is found in (Baker et al., 1980), the Bottom-Left Fill extension to this heuristic is used throughout this thesis for the placement of irregular parts. Baker et al explored various input sequences to this iterative placement algorithm and determined that the worst case scenario for layout height, when using a decreasing width ordering was not greater than three times the known optimal height.

These early techniques for the placement of orthogonal shapes were the foundations upon which later authors would build techniques for irregular shape placement.

2.4. Rectangular Enclosures & Simplification Techniques

Like the earliest work in the field the initial approaches to the irregular packing problem largely concentrated on pre-clustering of irregular shapes into rectangular enclosures to take advantage of the subsequent reduction in computational complexity during the placement phase.

An early example of this approach is found in the work of Haims who approached the problem in the Ph.D. thesis (Haims, 1966) and later extended this work in (Haims & Freeman, 1970). The approach in both publications was to determine the minimum bounding rectangle for a cluster of shapes and then use a dynamic programming technique for the packing of the rectangular enclosures.

The technique of minimum bounding rectangle shape clusters is also used in (Adamowicz & Albano, 1976a). The authors used the no-fit polygon, which defines a polygon along which the two polygons it is generated from will touch but not intersect with one another. This geometric construct was first proposed in (Art, 1966) and is the subject section 3.2 and of chapters 5 and 6 of this thesis. Following the generation of the clusters using the no-fit polygon Adamowicz and Albano packed the clusters on to sheets using dynamic programming.

In (Adamowicz & Albano, 1976b) the approach was extended so that techniques for efficiently packing pairs of polygons into sub-clusters could be used before the rectangular clustering in order to achieve higher packing density. However the work still produced loss of stock sheet between rectangles and (Albano, 1977) further extended the approach by displaying solutions to users and allowing them to manually manipulate the solutions to improve the yield.

Irregular shape clustering into orthogonal enclosures was also used in (Marques et al., 1991). The paper describes a dynamic programming and simulated annealing approach to the layout problem. The technique produces layouts by clustering irregular shapes into rectangular containers by manipulations of translation, rotation and inversion. The shapes are laid out using a grid structure of feasible positions in order to reduce the computational complexity. Only feasible solutions are allowed, if an infeasible layout is generated that branch is thereafter discounted from further consideration. This paper introduced one of the literature benchmark problems that are used to assess the effectiveness of the work described later in this thesis. A novel feature of this work is the use of bounding circles, rather than rectangles, for the elimination of the expensive operation of examining all potential overlaps between polygons on element by element basis.

(Jakobs, 1996) work is aimed at the sheet steel stamping/punching industry and develops work on rectangle packing from (Baker et al., 1980) . By using a genetic algorithm the author is able to improve upon the results of the orthogonal approach of the earlier work. The author then extends this approach to tackle problems involving irregular polygons using the polygons in their minimum bounding rectangle orientation, rather than clusters of shapes in orthogonal enclosures. The algorithm places the polygons using the improved genetic algorithm orthogonal method which is followed by a compaction phase during which a step wise bottom left algorithm runs until no shape can be moved any further towards the bottom or left of the sheet respectively. The paper includes discussion of clustering several shapes, finding the minimum bounding rectangle of these clusters and packing them orthogonally and then utilising the same compaction phase to improve the layout however no results from the implementation of this approach are included.

Other approaches to reducing the complexity of dealing with full irregular geometry have also been used by several authors; these often involve approximating the irregular geometry to overlays of simpler shapes or bitmap approximations of the shapes. These approximations allow the authors to avoid the overhead of computing full overlap and intersection detection routines; however they do come at the cost of accuracy of representation which can often lead to wasted space in the generated layouts.

(Qu & Sanders, 1987) tackled the problem of nesting irregular shapes. The authors presented two strategies for shape representation, a full representation of the shape geometry and additionally an overlay of the shape built from rectangles to reduce computational overhead. The full geometric representation was discarded as it used too much computer memory and required too much computation time. The rectangular overlay introduced a degree of inaccuracy into the nesting but this inaccuracy was reduced by using smaller rectangles in greater quantities to allow a greater degree of fidelity in representing small shape features. The shapes were placed in a bottom-left manner with overlap detection being carried out by performing many simple rectangular overlap tests.

A shape covering technique is also presented in (Han & Na, 1996). The authors introduce an unusual approach to the irregular packing problem, using fundamental shapes (circles, rectangles) to cover parts to reduce complexity of calculation of overlap. These fundamental shapes are placed interactively using a GUI tool and these covering shapes introduce a degree of inaccuracy as it is required that the covering shapes completely cover each shape and an overlap beyond the shape edge is quite probable. Another novel aspect of the work is that it is uses a Neural Network for initial layout in combination with simulated annealing for solution improvement. In the proposed approach the artificial neural network model is used for generating rectangular pattern configurations with an acceptable scrap. Rectangular patterns of different sizes are used as the input of the neural network to generate location and rotation of each pattern when they are combined. The pattern configurations generated through the neural network are represented as decision variables of a mathematical programming model for determining an efficient nesting of different sizes of rectangular patterns. The authors note that the results produced would be relevant for the thermal cutting industry, presumably as the solutions produced do not place shapes in touching positions. This is relevant to the thermal cutting problem as the destructive process of plasma, flame or laser cutting requires that the shapes are places a minimum distance apart from one another. However, the authors do not note that their technique was used in any industrial setting.

Bitmap and grid based approximations are also a useful way to reduce the computational overhead of dealing with irregular two dimensional geometry. (Ratanapan & Dagli, 1997) use a grid approximation, like Qu and Sanders previously, in order to reduce the computation time required for the generation of layouts. The approximation approach is similar to the computer graphics technique of anti-aliasing (Foley et al., 2003). Having described the approximation technique the authors outline an evolutionary algorithm approach which utilises simple operators to generate layouts. The authors report improved packing utilisations when they increase the resolution of the approximation and note the possible applicability of the work to parallel computation.

A gird approximation approach was also presented in (Bennell & Dowsland, 1999). The paper describes a tabu-thresholding approach which is used in order to overcome the author's observations that simulated annealing approaches have often been shown to get stuck in local optima and converge upon these optima too soon in the search. A tabu-thresholding approach is preferred in this work as it allows specific control of the diversification of solutions through specially designed moves incorporated into the solutions generation process. The authors use the (Ghosh, 1991) method to generate no-fit polygons to reduce computation time based during layout construction. As the layouts generated can contain overlap the authors incorporate a simple overlap penalty derived from the horizontal translation distance required to resolve the overlap between any two pieces that are intersecting, after initial experimentation the authors later revise this penalty to use a weighted rectangular measure. The problems tackled do not allow for rotation of shapes and the placement positions of shapes are limited by the grid of possible placement locations. The authors discuss the tuning of the parameters in the tabu search and note that the approach appears to be competitive with other generic approaches.

In (Ramesh Babu & Ramesh Babu, 2001) the authors tackle a variation on the nesting problem, producing layouts on irregular shaped sheets. The approach they adopted also used a bitmap type representation with a novel feature, rather than simply use binary pixel values to represent the shape they stored a horizontal distance from the shape boundary in each pixel. This allows the efficient resolution of overlaps during the layout process. The layout process in this work was driven by a hybrid genetic algorithm and heuristic approaches. Comparisons are made with their own work, from (Ramesh Babu & Ramesh Babu, 1999), and that of (Jakobs, 1996). The authors report that they have made improvements on both previous methods.

Bitmap representations of the shapes, to reduce computational overhead, are also used in (Wong et al., 2009). The paper introduces a two-stage packing approach to the irregular packing problem. The approach consists of a genetic algorithm manipulating the ordering of shapes for placement and a bottom-left fill placement heuristic. The authors compare results on eight new problems between bottom-left random search algorithm and the outlined approach and show clear improvements using the two phase algorithm. Unfortunately the authors do not compare with any literature benchmarks.

In (Clay & Crispin, 2001) the authors introduce the no-fit mask which is a construct akin to the no-fit polygon but generated using and represented in bitmaps. To generate the no-fit mask two datum points are tracked as the shapes orbit each other creating a region analogous to the no-fit polygon of the two shapes. These no-fit masks can be produced rapidly for any set of angles required and can cope with shapes including concavities. The layout procedure used for placing shoe parts onto hides is similar to that in (Crispin et al., 2003) which is discussed in the next section.

A placement technique for shoe parts is also presented in (Yang & Lin, 2009). The paper presents a genetic algorithm solution to problems from Taiwan's shoe manufacturing industry. The authors develop a combined in-house heuristic and genetic algorithm approach for two particular shoe lines. The authors compare the results of their work to real production figures and find that their approach reduces the average material requirements by 2.64% and reduces the time taken to produce a nest by over 69%. However, the authors note that further work would be required to produce a more comprehensive approach as the presented solution is the result of only a few of the many in-house heuristics available and represents a highly tuned genetic algorithm which may need to be retuned for each new problem instance and in-house heuristic.

2.5. Full Irregular Geometry Approaches without Overlap

The two-dimensional stock cutting problem has been shown to be NP hard and is therefore intrinsically difficult to solve (Garey & Johnson, 1979). There have been many different strategies for producing solutions to the irregular stock cutting problem. These include linear programming approaches, heuristic placement methods, metaheuristic guided search techniques and other novel approaches such as the iterative jostling of pieces (Dowsland et al., 1998).

Survey papers can be found in ((Dowsland & Dowsland, 1992); (Sweeny & Paternoster, 1992); (Dyckhoff, 1990); (Wäscher et al., 2007)). However, the feature that connects all of the approaches is that they are all required to cope with the geometry of the problem. This can be especially complicated when highly irregular shapes are used which may include holes or concavities.

The papers described in this section all use the actual geometry of the shapes that constitute a problem, rather than an approximation or simplified enclosure. The techniques noted in the first part of this section do not allow layouts with overlap to be regarded as admissible; this is in common feature of the layout generation techniques that this thesis presents in chapters 4, 6 and 7. Section 2.5.1 discusses techniques that allow interim overlap between shapes as a valid state and section 2.5.2 explores works that are designed to improve upon already generated layouts.

(Gurel, 1969) presents an early irregular shape placement technique for generating layouts based in a graph minimisation approach where joined nodes in a graph represent geometrically touching shapes. Larger shapes are first placed around the edge of the container then smaller shapes are then placed into the large shape layout.

(Albano & Sapuppo, 1980) presented another of the early papers on the irregular nesting problem where rectangular containers were not used for pre-packing of the shapes to be placed. The approach utilised the no-fit polygon in a bottom-left placement heuristic, i.e. no hole filling in the layout was possible. In order to make a shape placement the right edge of the current layout is tracked and the no-fit polygon of that edge is generated with the shape about to be placed. The lowest left most position on that no-fit polygon is then used as the placement position for the shape. The right edge is then updated and the process repeats until the layout is complete. The heuristic is driven by an A* search algorithm. (Moreau & DeSaint Hardavin, 1969) and (Tanaka & Wachi, 1973) also approached the layout problem without pre-packing into rectangular enclosures however neither set of authors used a search algorithm to explore the solution space.

(Fujita et al., 1993) approach the problem by separating the process of producing layouts into two phases. The authors use a genetic algorithm to tackle the combinatorial optimisation problem involved in determining a good ordering for placement of shapes. To produce a layout they use a compaction technique which is sensitive to the ordering produced by the genetic algorithm. The authors restrict the problems tackled to only instances containing convex polygons in order to reduce the computation time required. Solutions generated by the compaction technique do allow for some "overhang" with the edge of the sheet during the process but all shapes must be placed in a feasible position by the end of the process. The authors note that the geometric operations required significantly affect the time required to produce quality results. It is noted that a reduction in the computation time required for calculating overlap and distance between pieces would result in significantly improved time performance.

A genetic algorithm is also used in (Dighe & Jakiela, 1996) however rather than being used to manipulate a sequence of shapes for placement the genetic algorithm approach produces full layouts, unlike the earlier work of (Fujita et al., 1993). The technique involves a detailed chromosome which generates fully placed solutions. The first stage generates tight clusters of shapes through vertical compaction; these clusters are then packed on to layouts. Partial sub-trees are cached to increase computation speed of the more expensive geometric arrangement phase of the two part genetic algorithm. This work shows a significant performance improvement upon the work of (Smith, 1985) who used a similar approach on orthogonal problems.

(Bounsaythip & Maouche, 1997) presented a technique that improves upon the results from Fujita et. al (1993) in both efficiency and speed of solution generation. The paper describes a two stage approach based on an evolutionary algorithm approach which can only produce valid arrangements. The authors present a 'comb' representation which is used to track the difference between a shape and its bounding rectangle. The comb information is used to generate minimum bounding rectangle clusters of shape pairs whilst the evolutionary algorithm searches for possible shape combinations. This work is applied to problems from the textile industry which introduces some additional constraints, for example wishing to generate layouts based on strips of a width that would result in the roll of material being used up by some multiple of the width.

A two phase placement approach is also described in (Cheng & Rao, 1997). The implementation is an improvement to the work of Adamowicz and Albano (1976b) and latterly Grinde and Cavalier (1995). The authors extend the clustering technique that uses a shape sliding approach (akin to the Mahadevan (1984) approach that this thesis extends) to produce dense clusters of irregular shaped polygons for subsequent layout. The authors develop a less computationally expensive clustering approach to the forerunning techniques. This is achieved by a quasi-newtonian approach the authors dub the "stringy effect" which is used both to arrange shapes into clusters and place clusters into layouts. This is achieved by simulating a gravitational attraction of shapes to one another and utilising it to prompt their sliding around one-another searching for efficient arrangements. The approach is shown to be more efficient than the other techniques in terms of both speed and solution quality.

The use of the no-fit polygon has become an increasing feature of work in the irregular two-dimensional field since the mid-1990s, indeed few subsequent papers on the subject have not used the no-fit polygon to help reduce the computational overhead of dealing with the complex geometry. The remaining papers discussed in this section all use the no-fit polygon as part of their technique, sections 3.2, 3.3 and 3.4 describe the numerous techniques used to generate no-fit polygons and chapters five and six describe the extension of the algorithm first presented in (Mahadevan, 1984) in order to remove known degenerate cases and include circular arcs.

(Grinde & Cavalier, 1995) present a method that generates layouts based on the no-fit polygon generation approach described in (Mahadevan, 1984). The shape sliding approach used to generate no-fit polygons is utilised whilst searching for the minimal enclosure of a set of shapes. The authors note the various degenerate cases in the Mahadevan technique which later chapters in this thesis tackle in order to produce a complete and robust no-fit polygon generation technique, these degenerate cases include exact fits and tight channels leading to larger areas which should be part of the no-fit polygon but cannot be reached by Mahadevan's approach without extensions. The authors use convex hulls to add additional breakpoints to the sliding procedure in order to find the minimal enclosure. However, the approach is computationally expensive and was later improved on in (Cheng & Rao, 1997).

In (Dowsland et al., 1998) the authors generate no-fit polygons using the method devised by (Ghosh, 1991), a Minkowski sum based approach. The paper presents an extension to the (Dowsland & Dowsland, 1993) "Jostle" technique for improving layout utilisations, a technique discussed in section 2.5.2. In this development to the original work the authors extend the jostle approach to use no-fit polygons, in order to improve performance by avoiding repeated computation. The work includes a more comprehensive review of the strengths of the jostle approach based on experiments on more comprehensive datasets allowing the authors to identify key problem features such as shape concavity and shape length which the jostle approach works well upon. Clear performance gains when using the no-fit polygon are shown on all problem types, even when the shapes are all convex. The authors note that the computation of the no-fit polygons is a non-trivial endeavour in its own right.

A Minkowski sum technique for no-fit polygon generation is also used in (Milenkovic, 1997). The author presents a nesting algorithm based upon a computational geometry approach. The paper describes the utilisation of this work in the fabrics industry and notes that large cost savings have been made by taking advantage of the quality of the solutions generated. Further work was carried out on the problem in (Milenkovic, 1998).

In (Degraeve et al., 2002) the authors also explore a problem related to nesting in the fabrics industry. The paper focuses on reducing over production of fabric markers cut from high value material. In this formulation the problem consists of low production demands where over production is a major issue. The paper does not directly tackle the generation of layouts but rather uses a integer programming approach to optimise the combinations and number of layers of cloth for each size of garment to be cut in order to fulfil the orders, without significant over production.

A further solution to a problem also related to the fabrics industry, but not focused on layout generation directly, is presented in (Niemi, 2003). The author attempts to find a method for determining the correct quantities of parts that should be contained in a given set of layouts where the parts in question are repeatedly produced but are under variable demand. The technique aims to avoid over stocking and shortages by using a mathematical optimization model that can be used iteratively in connection with nesting. This approach is useful where effort spent producing a set of very high quality layouts for regularly produced parts give a trim-loss benefit when compared to constantly producing lower quality but more focused layouts.

In (Oliveira et al., 2000) the authors present a solution to the generation of nests of irregular shapes using a no-fit polygon layout technique that improved numerous benchmark instances. The authors use the technique of generating no-fit polygons "on the fly" whilst solutions are built. The technique involves generating and retaining a shape outline for all currently place shapes. The next shape in the placement sequence is then applied to the shape outline to produce a no-fit polygon, this no-fit polygon defines all the positions that the shape can placed and touch but not overlap the already placed shapes. The placement is then decided upon with respect to producing the smallest bounding rectangle or the minimum length layout. After placement the shape is added to the overall shape outline and the process is repeated. The results presented in this paper are compared an implementation the work of (Albano & Sapuppo, 1980) and benchmarked against the results of (Blazewicz et al., 1993) and (Dowsland & Dowsland, 1993). The authors show clear improvements on three of the five literature problems attempted. Later in (Gomes & Oliveira, 2001) the authors use a greedy heuristic to search for good orderings of shapes which are then laid out using a technique similar to the prior paper. The results obtained do not quite match the quality of the prior paper.

(Gomes & Oliveira, 2002) extends the work from the authors' previous papers in 2001 and 2000. It introduces a 2-exchange shape ordering heuristic for manipulating the order of shapes to be nested using an improved bottom-left fill layout mechanism called TOPOS. The layout process is improved by the inclusion of an inner-fit rectangle, the no-fit polygon of the inside of the sheet and a shape to be placed. The intersection points and vertexes of the no-fit polygons of the placed shapes the inner-fit rectangle with respect to the shape about to be placed define all feasible non intersection positions for placement. The paper explores the effectiveness of the 2-exchange heuristic by numerous tests on various initial shape orderings i.e. random, greatest length, greatest area. The 2-exchange heuristic is then used to improve the layout over fixed numbers of iterations. The paper presents a statistical analysis of the performance of the heuristic and several new benchmarks are set for some of the problems used in this thesis namely SHAPES1, SHAPES2, TROUSERS and SHIRTS, these benchmark problems and the others used to test the techniques presented in this work are summarised in section 2.8. The TOPOS placement technique is used to generate layouts whilst investigating a non-greedy placement approach in chapter seven of this thesis.

The usefulness of the no-fit polygon is also demonstrated in (Dowsland et al., 2002). The authors describe the implementation of a bottom-left fill placement algorithm. The authors use the no-fit polygon to optimise overlap detection. The authors note that they use "some appropriate method" to calculate no-fit polygons but do not state the exact method utilised. The paper introduces a concept of front and back edges on the no-fit polygon as a useful optimisation for reducing computation placement. By reducing the placement possibilities to the front edges of the no-fit polygon the authors report an execution time saving of approximately 80% in this context. The restriction to front edges is admissible as the presented placement algorithm precludes the usefulness of back edges as they would be "dominated" by already place shapes so any back edge placement would result in an overlapping placement. Additionally the authors use caching of no-fit polygons and retain information about previous placements positions to improve performance as it will not be possible to place shapes left of their most recently place position in a layout. The algorithm is not tested with an iterative search technique instead the authors use a set of static orderings to produce benchmarks. Various methods are used to generate orderings including shape area, length and perimeter of enclosing polygon. Although no rotations were included in the tested benchmarks the approach could be used with such problems.

(Gomes & Oliveira, 2006) use a linear programming compaction approach in combination with a meta-heuristic approach for layout generation. The no-fit polygon is used for overlap detection and part compaction however the no-fit polygons are generated from slightly simplified shapes, in that holes are removed and small concavities are removed from shapes for the generation process. A bottom left fill layout technique is used to generate solutions followed by a two phase linear compaction and separation technique. To improve solution quality these phases are all driven by a hybrid simulated annealing algorithm. Layouts generated by the bottom left fill heuristic, which is initialised to a random weighted shape length ordering, are then improved on by the linear compaction and separation approach solution acceptance is determined by the state of the simulated annealing algorithm. Pieces are swapped in the layout, causing overlap, so separation algorithm is used to generate feasible solution again before compaction improves it to local optima. Whilst the larger parts in the problems are tackled by the above approach the smaller parts are placed after the larger parts using a 2-exchange greedy method. The authors note that this approach is common within the garment production industry where parts naturally fall into large and small categories, similar to the approach of (Grinde & Daniels, 1999). Once the large and small parts have been placed the whole layout is compacted and assessed with respect to the state of the simulated annealing process. This technique produces excellent results and indeed improves on all the known best results for the literature problems tackled.

In (Martins & Tsuzuki, 2010) the authors explore a rotational approach to shape placement in fixed size containers. This variant of the cutting and packing problem is, as the authors note, not well represented in the literature indeed the authors refer to the typological work of (Wäscher et al., 2007) which found only 14 papers on this variant in the 413 papers they considered. The presented technique requires the utilisation of no-fit polygons; these are generated using a Minkowski sum approach employing decomposition for non-convex shapes. The authors use a simulated annealing algorithm to determine the angles of placement and the order in which shapes will be placed. Computational results are presented for new benchmark problems that have known optimal results, including one problem where the container has a hole, around which shapes must be packed. In all cases the presented algorithm is able to produce the optimal result for the benchmark.

(Bennell & Song, 2010) introduces the use of a beam search to aid the efficient construction of ordered lists of shapes for placement using a modified TOPOS layout heuristic as documented in (Gomes & Oliveira, 2002). The authors improve on some of the deficiencies of the original TOPOS approach by using the improved Minkowski sum no-fit polygon generation technique described in (Bennell & Song, 2008), discussed in section 3.4.5, which allows for the generation of no-fit polygons including interlocking shape pair positions which the original work by Oliveria et al. could not achieve as it was utilising a unmodified edge sliding introduced by (Mahadevan, 1984). Additionally, whilst the original TOPOS approach lost information regarding small holes in the packing by joining the polygon outlines together ignoring holes the author's presented approach does not lose this information, allowing for additional hole filling. The beam search approach is used to generate a good layout ordering by searching both on a local and global level in the search space. The authors present computational results highlighting the speed with which the solutions are generated and the usefulness of constructive placement heuristics that do not exclude local or global optimal solutions.

The concept of the collision free region was introduced in 2012 in the work of Sato et al.. The collision free region represents all feasible placement positions for a shape in a given layout, including some degenerated positions such as exact fits between shapes, single feasible points between shapes and exact fits inside the sheet. The technique used to generate layouts uses the parallel calculation of the collision free regions in combination with a simulated annealing search technique which control shape placement position and shape sequence input to the placement algorithm (Sato et al., 2012). The technique finds some new best results for the benchmark problems attempted; however the run times for the algorithm are very long for the more geometrically complex problems.

In the scientific literature the majority of problems tacked are to be packed into rectangular enclosures with one open dimension. However a small number of authors have attempted to nest onto irregular sheets using the full geometry of the nesting shapes or their no-fit polygons. It should be noted from in section 2.4 the use of approximations has often allowed irregular enclosures to be used as the additional complexity introduced by using an irregular sheet can be mitigated especially when using a bitmap placement approach. Bitmap placement approaches are discussed in section 3.4.2.

One such approach to placement on an irregular sheet is presented in (Tay et al., 2002). The authors introduce a layout approach driven by a genetic algorithm. Shapes are nested along the sheet boundary with at least one vertex touching the boundary and the shape is "removed" from the outline of the sheet resulting in a smaller sheet. The placement algorithm is then repeated exhaustively on the shapes in the sequence that the genetic algorithm generates, the genetic algorithm is also responsible for the shape placement angles and distance along the shape boundary that the shape is placed from an origin point. The authors' approach of edge and corner placement is designed to mirror their observations of human nesting behaviour. The algorithm is tested on both rectangular and irregular sheets.

Irregular enclosures are also explored in (Crispin et al., 2003). The paper offers a detailed description of the nesting problem as it pertains to the production of leather goods, specifically shoes. The additional constraints discussed include directional strength, quality areas and sheet defects. The presented approach uses the no-fit polygon for convex shapes and calculating feasible positions along the complex sheet geometry. Various shape quality requirements have to be taken into account, i.e. heal and toe pieces for shoes require different strength properties and qualities. In order that the quality constraints can be satisfied the problem definition is extended so that shapes are placed into their minimum quality zone and maximum deviations from the directional strength lines are also taken into consideration. The generation of the no-fit polygons and sheet inner-fit polygon is achieved by a technique derived from the field of image processing. A genetic algorithm is used to control angular placement and the allowed deviation from the required strength direction.

2.5.1. Full Irregular Geometry Approaches that Allow Overlap

The techniques presented in this thesis regard the overlap of shapes with one another and with the edges of the container to be an illegal state. The methods used to layout generation are therefore designed to avoid these states. However not all approaches to the two-dimensional irregular problem regards these states as illegal both during the packing process and even in the final layout, although solutions with overlap cannot be regarded as correct and are often highly penalised in the evaluation process. Some of the techniques that allow overlap have generated some of the best known solutions to the benchmark problems, described in section 2.8.

(Oliveira & Ferreira, 1993) presents one such method that allows overlap during the layout process. The authors tackle the problem using both a vector and a faster raster based approach for shape overlap detection. Both simulated annealing and local search are used to control the probability of accepting solutions generated. In order the generate solution the shapes are placed randomly inside a container. The shapes are then moved using a randomly selected move in one of the four cardinal directions. . The raster overlap detection approach produces interim layouts with overlap. However the vector approach does not as the use of D-Functions (Konopasek, 1981) allow full overlap resolution in the cardinal direction. The simulated annealing approach uses a cooling schedule to determine if a solution that is worse than the current best is acceptable. The authors present detailed analysis of the relative benefits of the various approaches and discuss the necessity for avoiding local optima to achieve improved overall results. An approach that also uses D-Functions to resolve overlap in the positive y-axis for pairs of shapes during a layout generation process is presented in chapter four.

Shapes are allowed to move off the container in the approach described in (Anand et al., 1999) which introduces an example of a solution designed for industrial leather goods production. The paper outlines a process of digitisation and polygonisation of hides and markers, to allow for the complex and unique shapes of hides and the possibility for the hide to contain defects. The complex shapes are simplified for the layout process by a process of convex decomposition to reduce overlap computation complexity. The paper describes a system which contains a database of parts, sheets and production requirements, common requirements for industrial production. A genetic algorithm approach is used to generate pseudo layouts which are used as the genetic chromosome representation including initial position and rotation angle. The pseudo layout is then converted to a real layout via a more computationally expensive process. As overlaps are resolved shapes can move off the sheet and in this case the shape is removed from the layout. Non overlapping parts are translated to a touching position based on the centre of gravity of the parts already placed. The authors also explore further uses of their technique by highlighting its use for generating good parings of shapes which further improve generated results. As is common in an industrial environment production requirements may not be satisfied by the production of a single layout and the work also tackles these multiple sheet problems.

Overlap during the process of layout generation is allowed in the technique described in (Egeblad et al., 2007). The authors use a guided local search to drive a translational nesting implementation. The paper finds many new best solutions for the benchmark instances attempted. The presented technique is a relaxed placement method that can handle irregular polygons including those with holes. This geometric approach is used in combination with guided local search to avoid the search stopping at local minima. Initially the algorithm places the all the shapes onto the sheet using a simple bounding rectangle approach and then iteratively reduces the sheet length translating outlying shapes into the newly resized container. The translations of the outliers may cause overlaps in the layout which the local search technique then attempts to resolve by reducing the overlap in the layout at each step. The authors also extend this approach to three dimensional problems and present experimental results for the three dimensional problems.

Sheet compaction and interim overlap are also features of the work presented in (Imamichi et al., 2009). The proposed algorithm utilises shape swapping procedures, using sheet compaction to induce compression of the layout and separation technique to resolve overlaps. In combination with a local search the algorithm produces good results on a number of literature benchmarks; indeed the paper presents some new best results. The technique requires the use of no-fit polygons in order to determine penetration depth between shapes. The nonlinear programming separation algorithm is able to reduce overlap in generated layouts and resolve overlaps during the improvement phases. The no-fit polygons are pre-computed for each problem prior to the running of the algorithm using a Minkowski sum approach. The authors compare their results with the work of (Gomes & Oliveira, 2006), (Egeblad et al., 2007) and (Burke et al., 2006) (the subject of chapter four of this thesis). The authors show improvements to existing published benchmarks eight of the fifteen attempted problem instances although runtimes for the presented algorithm are consistently longer than those of the compared papers.

The Imamichi et al. technique is used and extended in the approach presented in (Leung et al., 2012) wherein the authors add a tabu search, to avoid local optima, and add a compaction phase to the original technique. The tabu list is used to avoid part repeated part type moves for some set of iterations of the algorithm. The compaction operation is applied at the end of each run time to further improve the generated result. Utilising these additional steps the authors are able to produce some new improvements to the published benchmark problems discussed at the end of this chapter.

2.5.2. Techniques that Improve Existing Layouts

Improving completed layouts of irregular two dimensional shapes is a technique that has not been widely explored in the scientific literature. However one of the earliest papers describing an approach to perform an improvement to an existing layout produced a significant improvement on a benchmark problem used in this thesis. The approach was presented, along with other techniques for improving existing layouts, by (Dowsland & Dowsland, 1993). The work discusses techniques for identifying overlap in layouts and presents several complementary algorithms for producing and improving layouts. One approach that produced very good results when improving an existing layout is the "jostle". The technique uses the simple idea of shaking a box of small items in order to make the contents settle into a smaller volume. The jostle process shifts the placed parts, in turn, towards the left and then the right sheet boundaries. On each move holes available are filled with the moving shapes and over the iterations the packing is improved. The jostle algorithm is also reported to be one of the quickest methods of improving solutions explored returning improvements within a few minutes. Furthermore the jostle technique, in combination with the original placement algorithm, produced a long standing best result for one of the widely used benchmarks (SHAPESO) which was over 20% better than the Oliveira and Ferreira paper of the same year (Oliveira & Ferreira, 1993).

A further approach based on already nested sheets uses part complete layouts of larger shapes then attempts to tackle the placement of significantly smaller trim pieces into the containers formed between the larger parts. The approach outlined by (Grinde & Daniels, 1999) is designed to work as part of an interactive nesting process for the apparel industry, specifically trouser production. The trim pieces are grouped and linear programming techniques including a Lagrangian Heuristic, used to ensure reasonable completion times, are utilised to find good placements. The larger of the small trims are given greater weight as if they are not successfully placed they could result in a worse solution as they can affect the overall length of the generated solution.

In (Chernov et al., 2010) the authors use a fast but approximate nesting algorithm to generate layouts which are then improved using phi-function techniques. The authors discuss the principles of phi-functions and their relation to the no-fit polygon specifically that the no-fit polygon is a special case of the phi-function – the zero level set phi-function. The paper described how various primary object phi-functions e.g. rectangles, circles and triangles can be used to describe two-dimensional polygons by phi-function primary object union and disjunction. The authors generate initial layouts using a probabilistic search combined with an approximate bitmap packing algorithm this gives many results quickly as the authors wish to find a high quality initial layout. The initial layout quality is paramount to the quality of the layouts that the phi-function based improvement phase is able to generate. The improvement phase allows the phi-functions to be used to continuously translate and rotate shapes in order to reach some local optimal packing. Due to the mathematical building blocks of the phi-functions this phase completes very quickly. The authors report all problems attempted were optimised by the improvement phase in less than 125 seconds. This speed is also displayed for complex problems, involving the combination of many phi-functions, akin those introduced in this thesis i.e. from the metal cutting industry e.g. gaskets, flanges and rings and another based on the alphabet.

2.6. Approaches to Generating Single Shape Layouts

An industrially relevant variant to the two-dimensional irregular packing problems, which are studied in the later chapters of this thesis, is the packing of as many as possible of a single irregular part onto a rectangular enclosure. This is a reasonably common requirement in various industries where large numbers of parts may be required and waste minimisation is of interest. This is often the case where stamping or punching techniques are used to cut the shapes from the sheet material, rather than a more flexible moving cutting head approach.

One such approach is presented in (Dori & Ben-Bassat, 1984), the authors present a solution to the problem of tiling a single shape onto a rectangular sheet. The approach is based on tiling congruent shapes into one of a number of hexagonal containers which are then used for repeated placement. To determine the best of the hexagonal containers in which to contain the shape the authors generate the convex hull of the shape and compare it to a set of containers known to tile efficiently.

In (Prasad & Somasundaram, 1991) the authors also approached single irregular part placement problems with respect to the metal punching or "blanking" problem. The cutting process involves the production of a set of dies for repeated use to punch through metal sheet, cutting out the required shapes. For this purpose the authors approach the problem as a single congruent shape nesting problem although they note that their techniques can be applied to up to three distinct shapes. Clusters of shapes are produced in a pairwise manner by finding a good orientation for two shapes and then joining their outer contours to make a 'supershape' which is then fed into the next pairwise phase with another independent shape. The authors compare their results using this technique to a hill climbing approach and find significant improvement in yield. In (Prasad et al., 1995) the authors also worked on the metal punching problem but with additional technical constraints such as bridge width and metal grain orientation taken into account.

(Jain et al., 1990) compared the results of a simulated annealing approach and a multi-start hill climbing search technique for the nesting blanks problem. The authors found that a significant reduction in trim loss could be achieved when using the simulated annealing approach. The approach allowed irregular shapes to be placed in groups of up to three and the pattern generated is then repeated across the sheet.

A genetic algorithm was used by (Cheng & Rao, 2000) to tackle the problem of nesting multiple congruent parts. The genetic algorithm approach was used to finding useful angles for the shapes to be oriented in. They use an approach similar to the sliding no-fit polygon generation technique, discussed in chapters five and six of this thesis, to determine good orientations of shapes which can then be efficiently repeated. The authors report improved results from their previous work (Cheng & Rao, 1997).

(Joshi & Sudit, 1994) presented an exact polynomial time algorithm for a single row blank punching production problem. The single shape placement algorithm is tested on stripes which are large enough to contain the shapes in any orientation and strips that, due to their width, restrict the possible shape orientations. The authors also explore the problem of determining the optimal strip width given a stock size and present the results of their unconstrained approach.

(Stoyan & Pankratov, 1999) also worked on the packing of congruent polygons on rectangular sheets using a dense double lattice approach. The authors highlight the applicability of this problem type to the footwear industry. (Teresa Costa et al., 2009) also presents a new solution approach to the single congruent shape packing problem on fixed containers. The authors present three heuristics to solve the problem based on a period placement on a lattice technique. The approach utilises the no-fit polygon which is generated by the authors using a Minkowski sum technique. Computational results are compared with other literature approaches to the same problem. The authors report that whilst the heuristics developed produce the best results on the problems tackled its average results are usually worse than those other the literature techniques.

2.7. Techniques that allow for Circular Arcs in Shape Boundaries

The inclusion of circular arcs in problems found in the scientific literature is rare. The following papers present techniques that allow circular arcs or circles to be used in the shape boundaries of the irregular two dimensional packing problems of the type used in this thesis. Chapters four and six present new techniques for nesting and no-fit polygon generation respectively and both allow for the inclusion of circular arcs as part of the boundary of the irregular shapes being nested. No-fit polygons including circular arcs are also used in the work presented in chapter seven.

Circular arcs are included in the polygon boundaries of the problems tackled in (Blazewicz et al., 1993). The authors present a tabu search approach incorporating a bottom-left-fill layout heuristic which attempts to fill holes in the layout before placing a shape on the right hand edge of layout. A tabu search is used to drive simple shape moves between layouts. The authors report good results for their chosen benchmarks. The authors use a novel method of waste calculation to evaluate their results wherein they subtract the area of the rectangular container of each shape from the sheet area to determine the utilisation figure. This work is an extension of the technique introduced by (Albano & Sapuppo, 1980).

In (Blazewicz & Walkowiak, 1994) the authors extended the approach presented in (Blazewicz et al., 1993) to include a degree of interactivity by allowing users to rearrange automatically generated layouts. They note that that the interactivity has improved the quality of the solutions produced in the paper and highlight that this would not be possible without retaining the "segment-arc boundary" throughout the work. The avoidance of arc approximation and the associated benefits are discussed later in this thesis.

An approach related to layout generation that uses circles for layout evaluation is presented in (Dickinson & Knopf, 2000). The authors propose a novel evaluation matrix for two dimensional and three dimensional layouts. The paper argues that the widely used metrics, utilisation and overall length, cannot determine between two layouts of equal "quality". The "point moment measure" proposed judges the quality by evaluating clusters of shapes in circles and spheres based around the point moments of the shapes involved. The approach is applicable beyond the three dimensional case into N-dimensional problems. The authors assert that this point moment based approach is applicable as they show that there exists an optimal clustering of shapes when placed within a circle.

2.8. Summary of Benchmark Problems

(Hopper, 2000) made a useful contribution to the field by bringing together many other authors' datasets. The author then generated and compared results from both industrial software applications and techniques implemented for the work against this wider pool of problem types. Often it was reported that the commercial software outperformed the presented techniques. Hopper's approaches consisted of layout sequences generated by genetic algorithm placed by bottom-left and bottom-left-fill placement heuristics. These techniques are not only applied to the wide range of irregular problems the work gathers but also to orthogonal guillotine and non-guillotine instances. The work also introduced nine new randomly generated problem instances containing convex and non-convex polygons in quantities ranging from fifteen to seventy five. The full range of irregular problems collected by Hopper has been used to test the effectiveness of the techniques presented in this thesis. The following tables outline these problems; additionally Table 3 outlines the best results published during the period of the development of this thesis (2002 to 2012).

Ten additional problems were introduced in order to test unique features of the work presented in chapters 4, 5, 6 and 7. These problems are described in section 4.9 and fully outlined in the appendices.

Table 1. Length Evaluated Benchmark Problems

Original Author	Name	Shape Count	Rotational Constraints	Sheet Width
(Blazewicz et al., 1993)	Blasz1	28	0, 180 Absolute	15
(Ratanapan & Dagli, 1997)	Dagli	30	90 Incremental	60
(Fujita et al., 1993)	Fu	12	90 Incremental	38
(Jakobs, 1996)	Jackobs1	25	90 Incremental	40
(Jakobs, 1996)	Jackobs2	25	90 Incremental	70
(Marques et al., 1991)	Marques	24	90 Incremental	104
(Hopper, 2000)	Poly1A	15	90 Incremental	40
(Hopper, 2000)	Poly2A	30	90 Incremental	40
(Hopper, 2000)	Poly3A	45	90 Incremental	40
(Hopper, 2000)	Poly4A	60	90 Incremental	40
(Hopper, 2000)	Poly5A	75	90 Incremental	40
(Hopper, 2000)	Poly2B	30	90 Incremental	40
(Hopper, 2000)	Poly3B	45	90 Incremental	40
(Hopper, 2000)	Poly4B	60	90 Incremental	40
(Hopper, 2000)	Poly5B	75	90 Incremental	40
(Hopper, 2000)	SHAPES	43	90 Incremental	40
(Oliveira & Ferreira, 1993)	SHAPES0	43	0 Absolute	40
(Oliveira & Ferreira, 1993)	SHAPES1	43	0, 180 Absolute	40
(Oliveira & Ferreira, 1993)	SHIRTS	99	0, 180 Absolute	40
(Oliveira et al., 2000)	SWIM	48	0, 180 Absolute	48
(Oliveira et al., 2000)	TROUSERS	64	0, 180 Absolute	64

Table 2. Density Evaluated Benchmark Problems

Original Author	Name	Shape Count	Rotational Constraints	Sheet Width
(Albano & Sapuppo, 1980)	Albano	24	90 Incremental	4900
(Blazewicz et al., 1993)	Blasz2	20	90 Incremental	15
(Dighe & Jakiela, 1996)	Dighe1	16	90 Incremental	100
(Dighe & Jakiela, 1996)	Dighe2	10	90 Incremental	100
(Bounsaythip & Maouche, 1997)	Mao	20	90 Incremental	2550

Table 3. Summary of benchmark results

	Year Publication and Benchmark									
Problem	**2000**		**2002**		**2005**		**2008**		**2012**	
Blasz1	(Gomes & Oliveira, 2002)	27.3	(Burke et al., 2006)	27.2	(Burke et al., 2010)	26.8	(Bennell & Song, 2008)	26.57	(Bennell & Song, 2008)	26.57
Dagli	(Hopper, 2000)	65.6	(Burke et al., 2006)	60.57	(Burke et al., 2010)	59.94	(Bennell & Song, 2008)	57.64	(Sato et al., 2012)	57.4
Fu	(Fujita et al., 1993)	34	(Burke et al., 2006)	32.8	(Burke et al., 2010)	31.57	(Egeblad et al., 2007)	30.8	(Egeblad et al., 2007)	30.8
Jackobs1	(Hopper, 2000)	13.2	(Burke et al., 2006)	11.86	(Burke et al., 2010)	11.5	(Egeblad et al., 2007)	11.35	(Sato et al., 2012)	11
Jackobs2	(Hopper, 2000)	28.2	(Burke et al., 2006)	25.8	(Burke et al., 2010)	24.7	(Burke et al., 2010)	24.7	(Sato et al., 2012)	22.75
Marques	(Hopper, 2000)	83.6	(Burke et al., 2006)	80	(Burke et al., 2010)	78	(Egeblad et al., 2007)	76.67	(Egeblad et al., 2007)	76.67
Poly1A	(Hopper, 2000)	14.7	(Burke et al., 2006)	14	(Burke et al., 2010)	13.3	(Burke et al., 2010)	13.3	(Burke et al., 2010)	13.3
Poly2A	(Hopper, 2000)	30.1	(Burke et al., 2006)	28.17	(Burke et al., 2010)	27.9	(Burke et al., 2010)	27.9	(Burke et al., 2010)	27.9
Poly3A	(Hopper, 2000)	40.4	(Burke et al., 2006)	40.33	(Burke et al., 2006)	40.33	(Burke et al., 2006)	40.33	(Burke et al., 2006)	40.33
Poly4A	(Hopper, 2000)	56.9	(Burke et al., 2006)	54.93	(Burke et al., 2010)	54.6	(Burke et al., 2010)	54.6	(Burke et al., 2010)	54.6
Poly5A	(Hopper, 2000)	71.6	(Burke et al., 2006)	69.37	(Burke et al., 2010)	68.84	(Burke et al., 2010)	68.84	(Burke et al., 2010)	68.84
Poly2B	(Hopper, 2000)	33.1	(Burke et al., 2006)	30	(Burke et al., 2010)	29.63	(Burke et al., 2010)	29.63	(Burke et al., 2010)	29.63
Poly3B	(Hopper, 2000)	41.8	(Burke et al., 2006)	40.74	(Burke et al., 2010)	40.5	(Burke et al., 2010)	40.5	(Burke et al., 2010)	40.5
Poly4B	(Hopper, 2000)	52.9	(Burke et al., 2006)	51.73	(Burke et al., 2010)	51.18	(Burke et al., 2010)	51.18	(Burke et al., 2010)	51.18
Poly5B	(Hopper, 2000)	63.4	(Burke et al., 2006)	60.54	(Burke et al., 2006)	60.54	(Bennell & Song, 2008)	57.53	(Bennell & Song, 2008)	57.53
SHAPES	(Hopper, 2000)	63	(Burke et al., 2006)	59	(Burke et al., 2010)	56	(Burke et al., 2010)	56	(Burke et al., 2010)	56
SHAPES0	(Dowsland & Dowsland, 1993)	63	(Dowsland & Dowsland, 1993)	63	(Burke et al., 2010)	60	(Gomes & Oliveira, 2006)	60	(Imamichi et al., 2009)	58.3
SHAPES1	(Gomes & Oliveira, 2002)	59	(Burke et al., 2006)	58.4	(Burke et al., 2010)	55	(Burke et al., 2010)	55	(Leung et al., 2012)	53
SHIRTS	(Gomes & Oliveira, 2002)	63.13	(Burke et al., 2006)	63	(Burke et al., 2006)	63	(Bennell & Song, 2008)	61.33	(Imamichi et al., 2009)	60.83
SWIM	(Hopper, 2000)	6568	(Burke et al., 2006)	6462.4	(Burke et al., 2006)	6462.4	(Bennell & Song, 2008)	5895.2	(Imamichi et al., 2009)	5875.17
TROUSERS	(Gomes & Oliveira, 2002)	245.75	(Burke et al., 2006)	243.4	(Burke et al., 2006)	243.4	(Bennell & Song, 2008)	241	(Bennell & Song, 2008)	241
Albano	(Hopper, 2000)	86%	(Burke et al., 2006)	86.5%	(Burke et al., 2010)	87.23%	(Bennell & Song, 2008)	87.88%	(Sato et al., 2012)	89.21%
Blasz2	(Blazewicz et al., 1993)	68.6%	(Burke et al., 2006)	79.9%	(Burke et al., 2010)	80.41%	(Gomes & Oliveira, 2006)	83.61%	(Gomes & Oliveira, 2006)	83.61%
Dighe1	(Hopper, 2000)	72.4%	(Burke et al., 2006)	78.9%	(Burke et al., 2010)	83.84%	(Gomes & Oliveira, 2006)	100%	(Gomes & Oliveira, 2006)	100%
Dighe2	(Hopper, 2000)	74.6%	(Burke et al., 2006)	84.3%	(Burke et al., 2010)	86.5%	(Gomes & Oliveira, 2006)	100%	(Gomes & Oliveira, 2006)	100%
Mao	(Hopper, 2000)	71.6%	(Burke et al., 2006)	79.5%	(Burke et al., 2010)	80.9%	(Egeblad et al., 2007)	85.15%	(Egeblad et al., 2007)	85.15%

CHAPTER THREE

3. Methodology

3.1. Geometric Techniques

Before it is possible to tackle any two dimensional packing problem it is important that the practitioner has access to a robust and efficient geometry library. Without such a library it would not be possible to go on to create new layout techniques or utilise search mechanisms to discover good results. If the library operates inefficiently or inaccurately the quality of the results produced by a new approach or technique will be seriously compromised, no matter how innovative or potentially effective it may be.

The implementation of robust and efficient geometry routines can be laborious and can take longer than the development of packing strategies and search techniques themselves. In particular, the geometry must handle all of the interactions between shapes such as detecting whether two shapes are overlapping or calculating the translation distance required in a given direction so that the overlap is resolved. The following section discusses various aspects that were considered in the generation of the geometry library used throughout the thesis. This section does not include a full description of the geometry library used in the following chapters of this thesis however the majority of the algorithms utilised can be found in (Preparata & Shamos, 1985) and (O'Rourke, 1998).

3.1.1. Shape structure

In order to facilitate point in shape tests it is required that the polygons which constitute a shape are ordered in a particular direction i.e. clockwise or anti-clockwise. This way the polygon forms a single loop starting and ending at the same point. By building shapes up using this consistent directional rule, anti-clockwise in the case of this library (in keeping with the industrial partner's methodology), and marking internal holes as internal via a Boolean flag it has been possible to have a single implementation of the point in polygon test which works on all polygons in a shape.

Using the single point in polygon test it is possible to determine the relationship of any two shapes with respect to containment. To do this the winding number approach is used (also called Ray Crossing (O'Rourke, 1998)), the technique is outlined below.

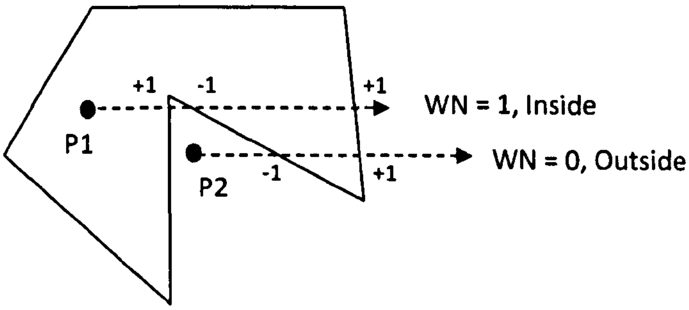


Figure 1. Winding number point inside test

Figure 1 shows an example of the winding number approach. Here P1 is inside the polygon and P2 is outside the polygon. To find the winding number (WN) we cast a horizontal ray from the point we are testing, on each crossing of the ray and the anti-clockwise polygon found we determine if it is a upward or a downward edge (by checking the Y coordinates of the start and end points i.e. start Y > end Y = downward and vice versa). If it is an upward crossing we add one to the point's winding number (WN) and if it is a downward edge we subtract one from winding number. Only points with a winding number equal to zero are outside the polygon.

The technique used for the point inside test is shown below in Algorithm 1 and Algorithm 2. The implementation in the library has also been adapted to cope with line arc polygons.

Algorithm 1. Winding Number Algorithm

```
Input: Point p =  point to be tested
       PolygonEdgeList[1..q] = edges of the polygon to be tested in
       anti-clockwise order and orientation


Begin
int    wn = 0;    // the winding number counter

// loop through all edges of the polygon
for (int i=0; i<q; i++)
{
       if (PolygonEdgeList[i].y <= p.y)
       {
             if (PolygonEdgeList[i].y > p.y)
             {
                     if (isLeft(PolygonEdgeList[i], p) > 0
                          wn++;
             }
       }
       else
       {
             if (PolygonEdgeList [i+1].y <= P.y)
             {
                    if (isLeft(PolygonEdgeList[i], p) < 0)
                         wn--;
             }
       }
}

return wn;

End
```

Algorithm 2. isLeft Test

```
Input: Point p =  point to be tested
       Edge e = edge of the polygon to be tested

//            >0 for p left of the edge e
//             =0 for p on the edge e
//             <0 for p right of the edge e

Begin

double    result = 0;

result = ( (e.p2.x - e.p1.x) * (p.y - e.p1.y)
           - (p.x - e.p1.x) * (e.p2.y - e.p1.y) );

return result;

End
```

3.1.2. Arc Geometry

Almost uniquely in the cutting and packing literature the work presented in this thesis tackles problems containing arc segments. In order to generate layouts and no-fit polygons containing arcs it is important that the geometry library is able to detect intersection of line segments with arc segments and arc segments with arc segments. The following section outlines some of the techniques used to achieve these intersection tests. These are presented as most geometry texts usually concentrate on line segments only for intersection tests.

3.1.3. Line Arc Intersection

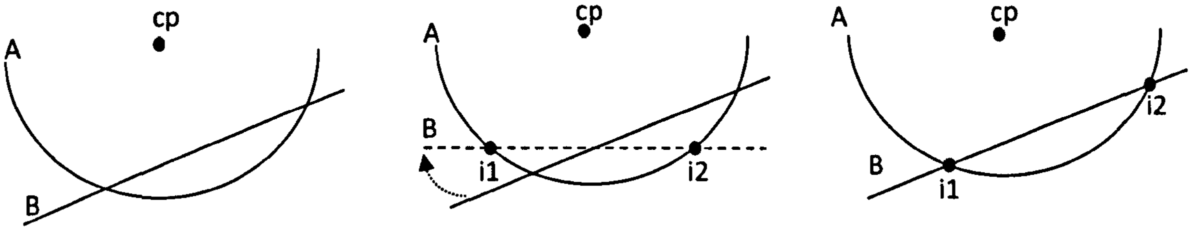


Figure 2. Arc Segment – Line Segment Intersection

Whilst several potential approaches to line arc intersection calculation are possible the method used in the geometry library is shown in Figure 2.

Given arc segment A and line segment B in the arrangement above we first test that the line, when rotated to horizontal about the arc A's centre point (cp), is within the y coordinate range of the arc, if not it cannot be intersecting. If it is then the intersection points are calculated for the horizontal line's infinite form and the arc. These points are then rotated back about cp by the reverse angle the line was rotated in order to make it horizontal. The rotated points are then tested against both the line and arc to ensure that they are on both elements as it is possible that the points generated against the infinite horizontal may not by on the line or may have rotated off the arc segment. Algorithm 3 summarises the process used below.

Algorithm 3. Arc Segment Line Segment Intersection

```
Inputs: Line ls = line segment for intersection test
        Arc as = arc segment for intersection test

Begin
if (ls is vertical)
{
        double relativeXValue = ls.GetStart().x - as.GetCentre().x;
        double y;
        double d = relativeXValue / as.GetRadius();

        if (d >= -1.0 && d <= 1.0) // is in the y range of the arc
        {
                double a = asin(d);
                y = as.GetRadius() * cos(a);
                pt1.x = ls.GetStart().x;
                pt1.y = as.GetCentre().y + y;
                pt2.x = ls.GetStart().x;
                pt2.y = as.GetCentre().y - y;
        }
}
else if (ls is horizontal)
{
        double relativeYValue = ls.GetStart().y - as.GetCentre().y;
        double x;
        double d = relativeYValue / as.GetRadius();

        if (d >= -1.0 && d <= 1.0) // is in the x range of the arc
        {
                double a = acos(d);
                x = as.GetRadius() * sin(a);
                pt1.x = as.GetCentre().x + x;
                pt1.y = ls.GetStart().y;
                pt2.x = as.GetCentre().x - x;
                pt2.y = ls.GetStart().y;
        }
}
else // get the horizontal y Value by rotating the line
{
        Point lineStart(ls.GetStart());
        double originalAngle = ls.GetAngle();

        lineStart.Rotate(as.GetCentre(),-originalAngle);

        yValue = lineStart.y;
        double relativeYValue = yValue - as.GetCentre().y;
        double x;
        double d = relativeYValue / as.GetRadius();
        if (d >= -1.0 && d <= 1.0) // is in the x range of the arc arc
        {
                double a = acos(d);
                x = as.GetRadius() * sin(a);
                pt1.x = as.GetCentre().x + x;
                pt1.y = yValue;
                pt2.x = as.GetCentre().x - x;
                pt2.y = yValue;
                pt1.Rotate(as.GetCentre(),originalAngle);
                pt2.Rotate(as.GetCentre(),originalAngle);

        }
}
if (pt1 is on ls and on as) return pt1
if (pt2 is on ls and on as) return pt2
End
```

This method was chosen over several other possible approaches as it yielded the best intersection detection accuracy. This accuracy is required in order to avoid overlap and is particularly important during the generation of no-fit polygons, covered in chapters 5 and 6.

3.1.4. Arc Arc Intersection

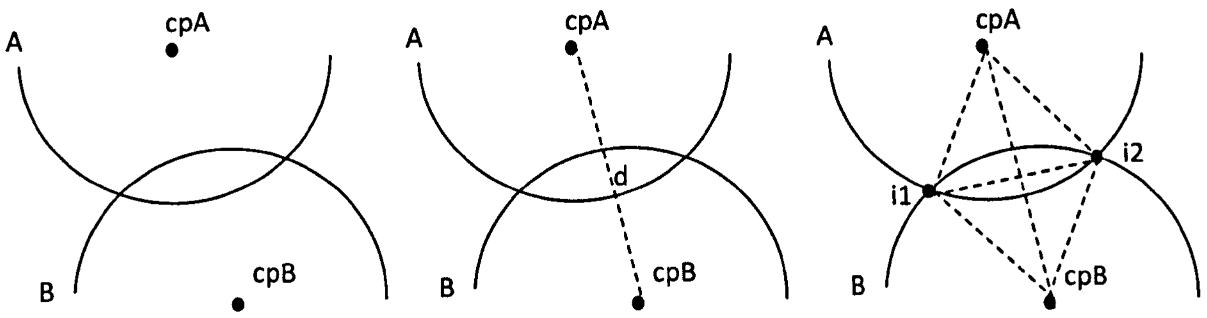


Figure 3. Arc Arc Intersection

Given arc segment A and arc segment B in the arrangement shown in Figure 3 we first check that the distance between the centre points (d) is less than the sum of their radii, otherwise they cannot be intersecting. We then use Pythagorean calculations to determine the potential intersection points i1 and i2.

Finally we check that the intersection points are on both arcs A and B as the calculation of the intersection points works on the circles of the arcs rather the arc segments and therefore could produce an invalid point. Algorithm 4 outlines the process in more detail.

Algorithm 4. Arc Arc Intersection

```
Inputs: Arc arc1
        Arc arc2

Begin

double r1= arc1.GetRadius();
double r2= arc2.GetRadius();
double d= arc1.GetCentre().DistanceToPoint(arc2.GetCentre());

double rBig, rSmall;

if(r1 > r2)
{
        rBig = r1;
        rSmall = r2;
}
else
{
        rBig = r2;
        rSmall = r1;
}

double rBig_Minus_rSmall = rBig - rSmall;
double r1_Plus_r2 = r1 + r2;

if(d >= r1_Plus_r2)
        return; // no intersect
else if(d <= rBig_Minus_rSmall)
        return; // small contained in big
else
{
        Point pt1, pt2;
        Arc big, small;
        if (r1 > r2)
        {
                big = arc1;
                small = arc2;
        else
        {
                big = arc2;
                small = arc1;
        }

        double value=((rSmall*rSmall)+(d*d)-(rBig*rBig))/(2.0*rSmall*d);
        double angle= acos(value);
        double centresAngle=small.GetCentre().AngleToPoint(big.GetCentre())

        double addedAngles = centresAngle + angle;
        double subtractedAngles = centresAngle - angle;

        pt1 = small.GetCentre().GetPointAtAngleAndDistance(addedAngles,rSmall);
        pt2 = small.GetCentre().GetPointAtAngleAndDistance(subtractedAngles,rSmall)
}

if (pt1 is on arc1 and pt1 is on arc2)
        return pt1;

if (pt2 is on arc1 and pt2 is on arc2)
        return pt2;

End
```

3.1.5. Accuracy

Using floating point numbers to represent polygons increases the accuracy of the representation of the shape that is ultimately desired for cutting. However it complicates many other tasks during the preparation of layouts and no-fit polygons, due to the inherent rounding errors whilst performing floating point calculations on processors. Many techniques exist to cope with such problems, see (Milenkovic, 1988), (Juster, 1992) and (Jackson, 1995) for potential approaches.

In the development of the geometry library for this thesis the technique known as finite precision geometry was utilised. In implementing this technique in the library each line, arc or point represented is regarded as a region of space of some acceptable error tolerance in size, as shown in Figure 4. This allows errors produced when manipulating the floating point data to be disregarded, providing that the implementation of intersection and touch detection always takes the region around the entities into account. The acceptable error tolerance utilised in the generation of the geometry library was 10^{-7} as the library which this was developed from, the industrial partner's Procut geometry library, used a unit base of 1mm. This means the 10^{-7} is equivalent to nanometre scale errors in representations, which are far smaller than the accuracy of industrial cutting processes and can therefore be disregarded.

Many other researchers continue to develop approaches for computationally exact geometry systems, information on current exact geometry systems can be found in (Yap, 1997) and rounding considerations in (Guibas & Marimont, 1995), (Goodrich et al., 1997), (Milenkovic, 2000) and (Halperin & Packer, 2002). However, the additional accuracy obtained through computationally exact approaches is usually at the detriment of computation times.

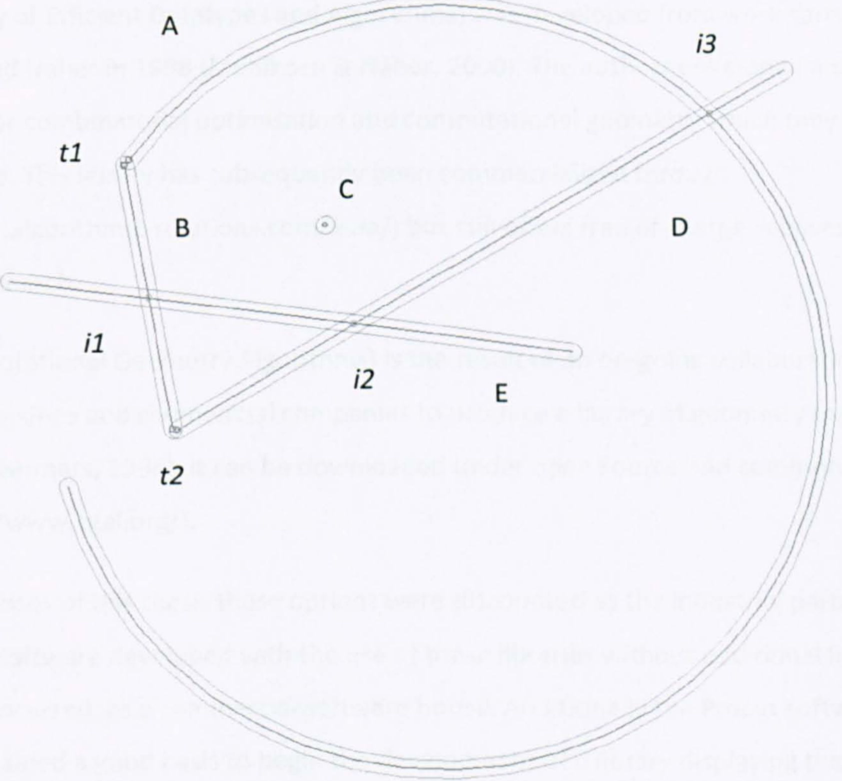


Figure 4. Geometry representation showing finite precision, intersection and touch points

Figure 4 is an annotated screen shot from a CAD tool developed alongside the library in order to input shape data and test the geometry library's accuracy and speed. The lighter grey paths outside each of the entities drawn show the region the library regards as representing the entity – the region of error (epsilon). The figure shows and arc segment (A), three lines (B, D and E) and a point (C) drawn close to the epsilon value in order to highlight the regions which the library regards as the entities.

Additionally this screenshot shows that the library has been used to calculate intersection and touch points between all of the entities (*i1*, *i2*, *i3* and *t1* and *t2*).

3.1.6. Alternative Approaches

The development of such a geometry system is a complex and time-consuming undertaking and it is likely that other practitioners may like to consider general geometry libraries such as LEDA and CGAL.

LEDA (Library of Efficient Datatypes and Algorithms) was developed from work carried out by Mehlhorn and Naher in 1998 (Mehlhorn & Näher, 2000). The authors developed a set of algorithms for combinatorial optimisation and computational geometry which they released under license. This library has subsequently been commercialised through (http://www.algorithmic-solutions.com/leda/) but still offers free of charge licenses for academic research.

CGAL (Computational Geometry Algorithms) is the result of an on-going collaboration between several universities and commercial companies to produce a library of geometry representation and tools (Overmars, 1996). It can be downloaded under open source and commercial licenses from (http://www.cgal.org/).

For the purposes of this thesis these options were discounted as the industrial partner could not use any software developed with the use of these libraries without additional licensing costs being incurred, as a commercial software house. Additionally the Procut software product already contained a good basis to begin the development of a library displaying the correct attributes to support the development of the work in this thesis and the CASE/TCS projects.

3.2. The No-Fit Polygon – An Overview

The following section describes the functionality of the no-fit polygon and compares it to the more traditional trigonometric based overlap and intersection tests covered in previous section. A brief overview of the many techniques that have been used to generate no-fit polygons within the previous literature is also provided.

The first application of no-fit polygon techniques within the field of cutting and packing was presented by (Art, 1966), although the term "shape envelop" was used. It was ten years later that the term "no-fit polygon" was introduced by (Adamowicz & Albano, 1976b) who approached the irregular stock cutting problem by using no-fit polygons to pack shapes together using their minimum enclosing rectangles. The term "configuration space obstacle" is often used to denote the NFP within the field of engineering and robot motion planning but the term has also been used with respect to cutting and packing in (Cunninghame-Green, 1989).

The "hodograph" is often used to describe the no-fit polygon within the mathematics community ((Stoyan & Ponomarenko, 1977); (Scheithauer & Terno, 1993); (Bennell et al., 2001)).

In (Bennell & Oliveira, 2008) the authors present a discussion of the importance of geometry in tackling irregular nesting problems. Specifically focusing on the generation of no-fit polygons as a key element required for any practitioners of irregular nesting problems. The work covers the extended Mahadevan approach presented later in this thesis; the Minkowski sum approach and discusses the Phi function approach (Bennell et al., 2008a). The authors hope that the paper will encourage potential practitioners to attempt the subject by giving a clear guide to the geometry which the authors note can be a major barrier to undertaking research in the field.

The main function of the no-fit polygon is to describe the region in which two polygons intersect. The following example gives an overview of the NFP construct.

Given two polygons, A and B, the no-fit polygon can be found by tracing one shape around the boundary of another. One of the polygons remains fixed in location and the other traverses around the fixed polygon's edges whilst ensuring that the two polygons always touch but never intersect. Throughout this thesis the convention of the first polygon being fixed and the second being the traversing/orbiting polygon is used. Therefore when polygon B traces around the fixed polygon A, the resulting no-fit polygon is denoted by NFP_{AB}. In order to create the NFP_{AB} object we must choose a reference point from B which will be traced as B moves around A. In the implementation the first vertex within the shape vertex list is used as the reference point (see Figure 5).

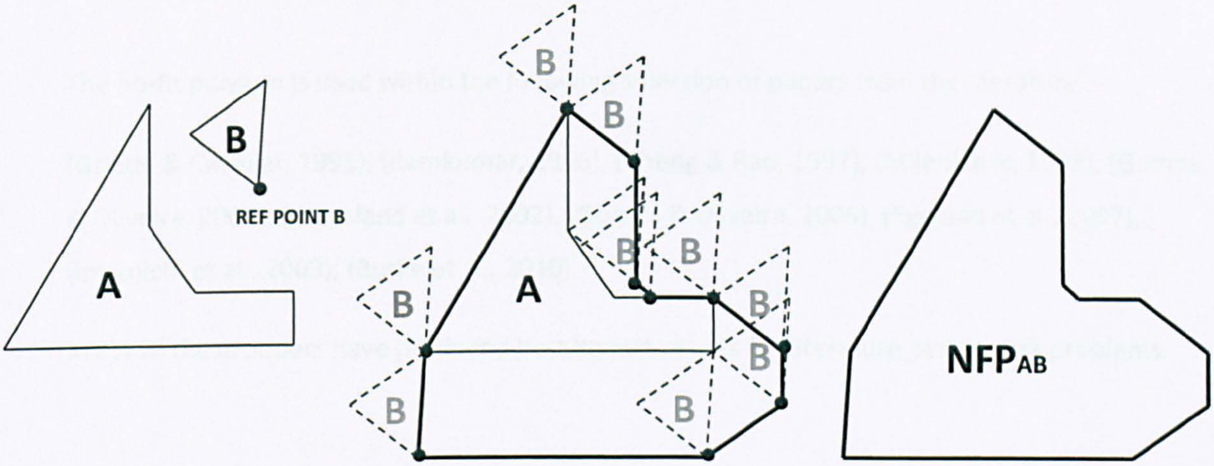


Figure 5. The no-fit polygon of two shapes A and B

The reference point can be any arbitrary point providing it mimics the movements of the orbiting polygon. It is also important to maintain the relative position of the reference point with respect to polygon B as this is required when using the NFP to test for overlap.

In order to test whether polygon B overlaps polygon A we use NFP$_{AB}$ and B's reference point. If polygon B is positioned such that its reference point is inside the polygon NFP$_{AB}$ then it overlaps with polygon A. If the reference point is on the boundary of NFP$_{AB}$ then polygon B touches polygon A. Finally, if the reference point is outside of NFP$_{AB}$ then polygons A and B do not overlap or touch. The three possibilities that we have described above are illustrated below in Figure 6.

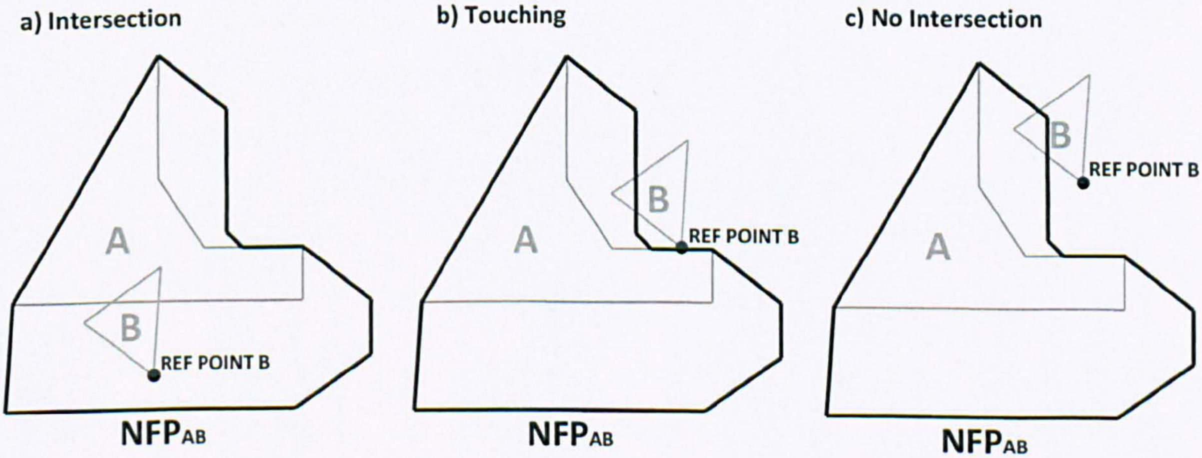


Figure 6. Using the no-fit polygon to test for intersection between polygons A and B

The no-fit polygon is used within the following selection of papers from the literature:

(Grinde & Cavalier, 1995), (Ramkumar, 1996), (Cheng & Rao, 1997), (Milenkovic, 1999), (Gomes & Oliveira, 2002), (Dowsland et al., 2002), (Gomes & Oliveira, 2006), (Egeblad et al., 2007), (Imamichi et al., 2009), (Burke et al., 2010)

Many of these papers have produced best known results for literature benchmark problems.

3.3. No-Fit Polygon vs. Standard Trigonometry Overlap Detection

Although the no-fit polygon is an excellent tool for conducting intersection tests between pairs of polygons, it has not been widely applied for two-dimensional packing problems in both the literature and in real world manufacturing industries. Undoubtedly this is due to the no-fit polygon's complex implementation and lack of complete and robust algorithms. Instead, many intersection implementations use standard trigonometry approaches which especially occur in the case of packing software for real-world applications wherein it is important that distributed software is able to handle all possible polygons without errors. However, whilst both approaches have the same overall effect, use of the no-fit polygon can be several times quicker than even the most efficient trigonometrical routines. For example, where it may be required to attempt numerous iterations of the same layout problem the pre-generation of no-fit polygons can significantly reduce the total computation time as over numerous iterations, trigonometric approaches are likely to repeatedly detect and resolve the same overlapping shapes in repeated orientations and positions.

Where a nesting overlap resolution approach requires the calculation of all intersection points between two intersecting shapes the benefits of a no-fit polygon approach are multiplied further as numerous intersection calculations are required for the full detection of collision when the no-fit polygon method is not used. By utilising no-fit polygons it is possible to reduce the overlap detection problem (which is a major factor in the computational overhead of nesting process) to a significantly less expensive point inside polygon test (Dowsland et al., 2002). In addition to this when utilising the overlap resolution technique used in the chapter 4 and published in (Burke et al., 2006) the intersecting shapes are resolved through the repeated resolution of intersecting edges in the y-axis direction, by utilising the no-fit polygon the resolution technique is more efficient, resolving the y–axis overlap in one complete movement. Further to this, the no-fit polygon would also allow for the overlap to be resolved in *any direction* by casting out a ray from the relevant reference point and finding the nearest intersection with the no-fit polygon boundary.

The following section compares the computation necessary to detect and resolve the overlap between two polygons where we have pre-calculated the NFP and where we have not. Given a nesting method where we wish to detect all points of intersection for polygons A and B (see

Figure 7), every edge is tested against every other edge leading to 42 tests to determine intersection status (given that polygon A has 7 edges and polygon B has 6 edges). Furthermore, if we find no intersection, in order to eliminate the possibility of the polygons intersecting through their vertices, or one polygon completely containing the other, we must perform a point inside test for all points on both shapes, requiring 56 tests in total, in the worst case. These can be regarded as a large, but in the main unavoidable, overhead for the detection of intersection between any two polygons in a layout, which can only increase as we add an increasing number of polygons to our problem. Whilst many optimisations can be attempted and fast intersection libraries can be developed, intersection detection remains a considerable portion of the computational overhead inherent in the generation of packing solutions. Additionally where the nesting process requires the generation numerous solutions using a combinatorial optimisation approach these, often repeated, calculations will have considerable impact on the overall computation time.

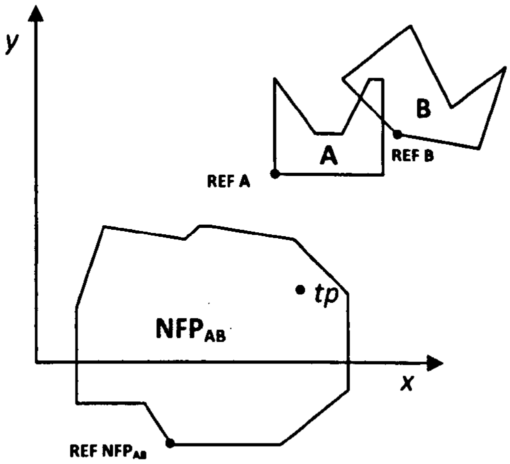


Figure 7. Intersection testing with the no-fit polygon

In Figure 7 the no fit polygon of the two polygons, NFP$_{AB}$, is also shown at some arbitrary position in the problem space, in this case the NFP$_{AB}$ has been generated using the Mahadevan edge sliding technique where polygon B traversed the edges on the polygon A and the NFP was formed by tracking the reference point, REF A, on the traversing polygon. In the context of the overlap detection process the position of the NFP is rendered inconsequential by generating the test point (*tp*) using a simple translation, (REF NFP$_{AB}$ + REF B – REF A – *Offset*), where *Offset* is the positional offset from the reference point of polygon A and the reference point of the no-fit polygon during generation (REF NFP$_{AB}$ – REF A).

The status of *tp* with respect to NFP_{AB} can be calculated using a winding number technique described in section 3.1.1 or the ray-crossing algorithm (O'Rourke, 1998). If the point *tp* is found to be within NFP_{AB} then the polygons A and B are colliding with one another, colliding includes both intersection and containment of one polygon by another. If *tp* falls on the NFP_{AB} then the polygons are known to be touching and if *tp* falls outside of NFP_{AB} then the shapes are neither touching nor colliding. Both touching and not colliding are viable states for polygon placement in numerous nesting approaches. By undertaking the calculation of all no-fit polygons for all possible pairs of polygon rotations, we can save considerable computation time when undertaking multiple iteration nesting.

3.4. Approaches for No-Fit Polygon Construction

The following subsections review the main techniques that have previously been used for construction of no-fit polygons within the literature. For each method there is a brief overview of the approach and discussion any benefits and drawbacks that may occur from its usage.

3.4.1. Convex Shapes

The basic form of no-fit polygon generation occurs when both polygons are convex. Given two convex shapes, A and B, the no-fit polygon is created by the following steps:

- Orientate shape A anticlockwise and shape B clockwise (see Figure 8a)
- Translate all edges from A and B to a single point (see Figure 8b)
- Concatenating these edges in anticlockwise order yields the no-fit polygon (see Figure 8c)

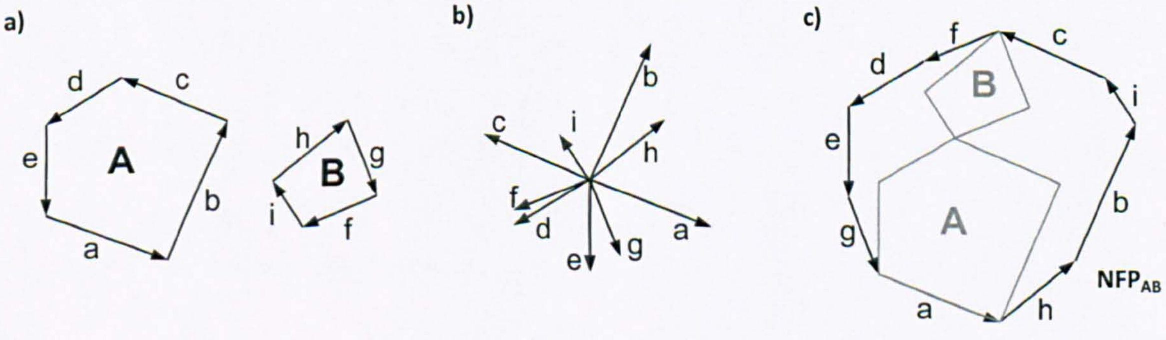


Figure 8. No-fit polygon generation with convex shapes

(Cunninghame-Green, 1989) used this approach to produce "configuration space obstacles" between pairs of convex polygons which are then used for intersection tests during the packing of shapes. For instances involving non-convex pieces, Cuninghame-Green firstly calculates the convex hull of each non-convex shape (the minimal containing convex polygon) and then calculates no-fit polygons using the respective convex hulls. The benefit of convex no-fit polygon generation is that it is simple and is extremely quick using a standard sorting algorithm in combination with edge reordering through translation. The obvious disadvantage is that no-fit polygons cannot be generated for non-convex shapes and the reduction to convex hulls results in concavity sections being unavailable in packing and non-traversable in robot motion planning. As this can adversely affect solution quality so other approaches are required. There have been many different approaches to producing no-fit polygons from non-convex shapes. These can be placed into four general categories: digitisation, decomposition, Minkowski sums and orbital approaches and are discussed in the following sections.

3.4.2. Digitisation

One of the simplest approaches to producing no-fit polygons from non-convex shapes is to discretise the shapes through digitisation. Firstly, the shapes are represented as 0-1 bitmap objects whereby "ones" represent the solid body of a shape and "zeros" indicate empty spaces (see Figure 9a). As intersection detection can be implemented easily with bitmaps through boolean operations, the no-fit polygon can be created through multiple scans of shape B across shape A or through orbital approaches. The problem with such an approach is that there is speed accuracy trade-off. For example, with a low resolution grid the process can be very fast. However, increases in resolution will require more and longer scan lines and thus speed will be compromised.

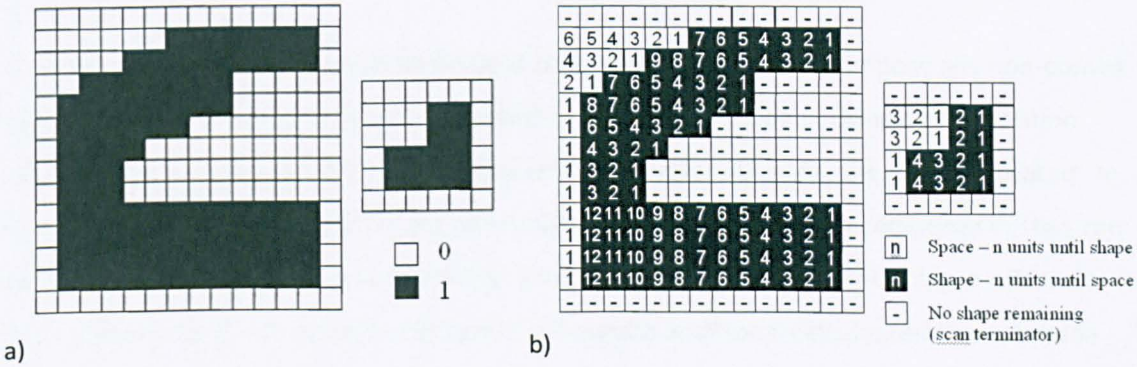

a)
0
1
b)
Space – n units until shape
Shape – n units until space
No shape remaining (scan terminator)

Figure 9. Bitmap representations of non-convex pieces with a 0-1 representation or index optimisations

The discrete representation of the bitmaps may also allow for certain optimisations. Ramesh Babu and Ramesh Babu utilise indexing within their bitmap representations whereby "zeros" and "ones" are replaced with index entries that indicate how many 'pixels' must be traversed horizontally until an empty space is reached (Ramesh Babu & Ramesh Babu, 2001). Although they do not calculate no-fit polygons within their layout generation implementation, it is apparent that the fundamentals of their indexed scanning technique are also applicable to digitised no-fit polygon generation (see Figure 9b). Another imaging technique has been used by (Crispin et al., 2003) for the generation of leather lay plan layouts within the footwear industry. The no-fit polygon is found using image processing techniques by filtering an image of the cow hide boundary with an image of the shape to be placed. The convolution between the two shape images yields the no-fit polygon. This is another form of the Minkowski sum that is discussed later in this section. Ultimately, all digitisation or imaging approaches will result in inaccurate no-fit polygons because of the loss of information resulting from the discretisation of a continuous space. While it is true that the resolution could be set to an arbitrarily high value, this may severely affect computation times. One benefit of the digitisation approach is that it is able to handle most of the traditional problem cases, such as holes and interlocking concavities however it may be possible that jigsaw shapes, after digitisation, may not fit together whereas they would in their pure vector form.

3.4.3. Decomposition

An alternative to the digitisation techniques described above is to decompose any non-convex shapes into sub pieces that can be 'managed' more easily. However, using decomposition usually results in several sub pieces and therefore several no-fit polygons must be created. In order to conduct intersection testing, the no-fit polygons can remain decomposed or they can be recombined through union operations. Where possible, a single no-fit polygon offers the fastest intersection computation times but will require additional calculations to recombine constituent parts. This can be a computationally expensive and a difficult undertaking if several subparts are present and this may be further complicated if holes are present. For example, (Agarwal et al., 2002) conducted experiments on different decomposition and recombination operations with respect to constructing Minkowski sums of non-convex polygons without holes. They conclude that it is counterproductive to use optimal decompositions because the computation times to calculate them outweigh the benefits achieved during recombination. The authors report that the recombination operations are the most costly and give example execution times that range from a few seconds to produce the no-fit polygon for shapes involving a small amount of concavities and up to twenty minutes for highly irregular shapes.

3.4.3.1. Convex Pieces

As previously noted the no-fit polygon generation is trivial when convex shapes are involved. If shapes with concavities can be divided into convex pieces, fast convex no-fit polygon generation techniques can be employed. The simplification of geometrical intersection through the decomposition of non-convex shapes to convex pieces was suggested and discussed in (Avnaim & Boissonnat, 1987) and (Cunninghame-Green, 1989). The main difficulties with such an approach are the decomposition and recombination algorithms.

There are many well-known approaches that can be used for this decomposition including triangulation and convex partitioning. (Seidel, 1991) suggests a fast polygon triangulation algorithm which has complexity of an O(n log n) complexity. Further implementation details for triangulation can be found in (Amenta, 1997). However, for our purposes, the triangulation approaches produce more sub pieces than is necessary and will ultimately impact upon computation time within the generation process. Unlike triangulation, which can be seen as a special case of convex partitioning, generalised convex partitioning algorithms aim to represent polygons with as few convex pieces and as quickly as possible. The two key approaches are suboptimal partitioning, one of which has an O(n) complexity (Hertel & Mehlhorn, 1983), and optimal partitioning which has an O(n^3) (Chezelle & Dobkin, 1985). (Agarwal et al., 2002) state that it is generally more efficient to use suboptimal partition algorithms because of the computational overhead inherent with optimal partitioning but they also suggest that an alternative and possibly more efficient approach is to perform convex covering instead. Some possible decompositions of an irregular polygon into convex pieces are shown in Figure 10.

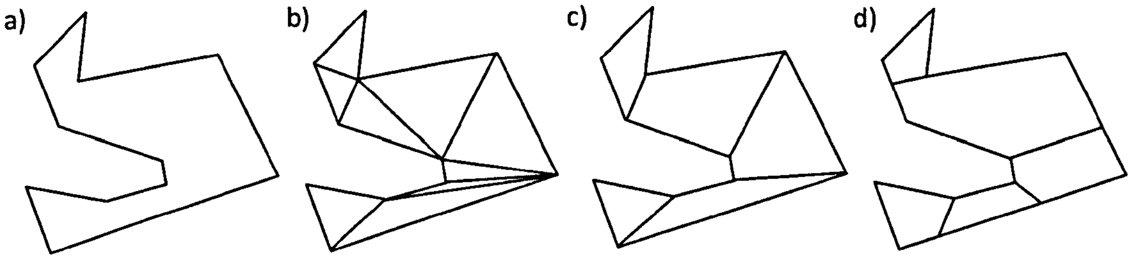


Figure 10. Convex Decompositions
a) irregular polygon, b) triangulation, c) convex division (vertex to vertex),

d) convex division (vertex to edge)

Once any irregular polygons have been decomposed into convex pieces, the no-fit polygon may be generated by recombining the no-fit polygons produced by passing each convex piece of shape B around each convex piece of shape A. Providing these no-fit regions have been generated in relation to a reference point, they may then be recombined. The disadvantage with this approach is that recombination can be difficult because the no-fit polygons of the convex partitions may cross and, therefore, care must be taken when constructing the singular no-fit polygon entity. Particular difficulty occurs if the original shapes contain holes as it is unclear whether intersecting no-fit polygon subsections define holes or whether they define discardable regions.

3.4.3.2. Star-Shaped Polygons

(Li & Milenkovic, 1995) decompose shapes into convex and star-shaped polygons. A star-shaped polygon has the property that there exists at least one internal point, or 'kernel point', that can 'see' the entire boundary of the polygon, (Preparata & Shamos, 1985) and (O'Rourke, 1998). Figure 11 shows one non-star-shaped polygon and one star-shaped polygon. By extending the concavity edges and elimination of invisible regions it is evident that the star-shaped polygon has a region, R_{kernel}, that can be defined within which a kernel point can be placed to see the entire polygon boundary. Conversely, this is not the case with the non-star-shaped polygon because no region can be defined in which to place a kernel. Thus, star-shaped polygons are situated somewhere between convex and non-convex shapes in terms of generality.

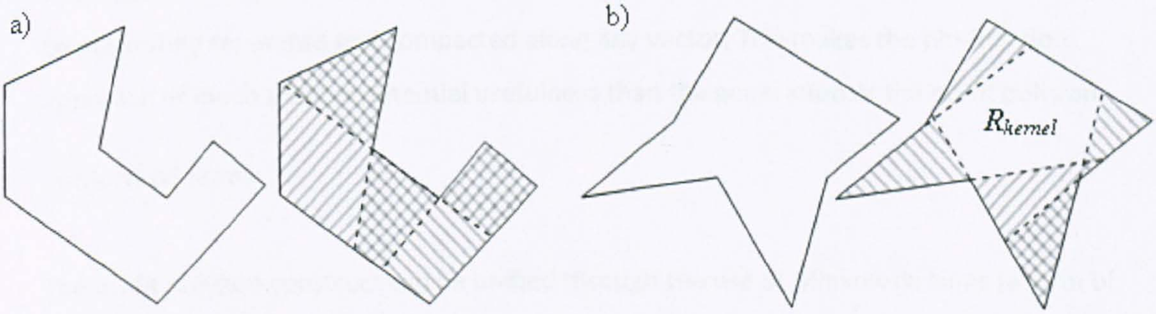


Figure 11. a) Non-star-shaped polygon, b) Star-shaped polygon

Li and Milenkovic state that star-shaped polygons are 'closed' under Minkowski sum operations and provide a proof that the Minkowski sum of two star-shaped polygons also yields a star-shaped polygon. The no-fit polygon is then created through use of an angular sweep line algorithm. The authors do not state whether they recombine the no-fit polygon regions into one no-fit polygon entity or whether they perform multiple no-fit polygon intersection tests during the layout generation stage.

3.4.4. Phi-Functions

An alternative approach is presented by (Stoyan et al., 1996) and is based on the use of "phi-functions". Phi(Φ)-functions define mathematical intersection relationships between pairs of standard or "primary" objects such as rectangles, circles and convex polygons (Stoyan et al., 2001). The authors further develop their work to enable the definition of mathematical intersection relationships for non-convex polygons through the union, intersection and complement of primary objects in (Stoyan et al., 2002). The resultant intersection test between two shapes during layout generation is performed through comparisons of phi-functions between all pairs of the primary objects that define shape A and shape B. (Bennell et al., 2008a) investigate Φ-functions and their relationship with Minkowski sums and the NFP. They outline the advantages of Φ-functions over other approaches, providing a clear definition for the set of objects for which Φ-functions may be derived. A procedure for deriving Φ-functions, together with examples, is also presented. Whilst the generation no-fit polygons is possible using the phi-function approach this represents only a small fraction of their potential usefulness to the field of irregular packing. Phi-functions describe the relationship between set of shapes, the zero level phi-function describing the no-fit polygon, which allows for shapes to be accurately separated and compacted along any vector. This makes the phi-function approach of much greater potential usefulness than the generation of the no-fit polygon.

3.4.5. Minkowski Sums

The no-fit polygon construct can be unified through the use of Minkowski sums (a form of vector addition) and was first suggested by (Stoyan & Ponomarenko, 1977). The concept is as follows: given two arbitrary point sets, A and B, the Minkowski sum of A and B is defined by the following: $A \oplus B = \{a + b: a \in A, b \in B\}$

In order to produce no-fit polygons, the Minkowski difference $A \oplus -B$ must be used. This is equivalent to two input polygons in opposing orientations and is easily shown through simple vector algebra (Bennell et al., 2001). This can be verified using the simple convex only case (section 3.4.1) whereby a shape A is placed in its anticlockwise orientation and shape B in its clockwise orientation. This example shows that the method of Cuninghame-Green, involving convex shapes only, is *also* using the Minkowski difference in its most basic form.

Whilst non-mathematical implementation details of such approaches are scarce (Ghosh, 1991) and subsequently (Bennell et al., 2001), provide excellent explanations and implementation details for no-fit region calculation. Ghosh presents details about the production of no-fit polygons through the notion of Minkowski difference and demonstrates a simple extension to the convex only case that allows for the generation of no-fit polygons for non-convex shapes. The drawback is that this approach will only work providing that the concavities of the two shapes do not interfere or interlock (see (Ghosh, 1993)). Where this is not the case, a modification is proposed but is only briefly outlined.

In (Bennell et al., 2001) the authors present a powerful extension to the earlier work of (Ghosh, 1991). The authors state that Ghosh's modification would cause a "cumbersome tangle of intersecting edges" which would be difficult to recombine to form the no-fit polygon. They introduce a further implementation that reduces the amount of 'tangled edges' and give thorough implementation details and pseudocode of the entire process. They also report fast generation times of around 0.3 seconds for each of five separate datasets from the literature. However, they state that their approach cannot deal with internal holes as it is difficult to detect which of the internal no-fit polygon edges can be discarded and which form the internal no-fit regions. The technique derived allows for the efficient generation of no-fit polygons for concave shapes with few degenerate cases. The approach uses convex hulls to overcome the problems in the original technique where interacting concavities resulted in unsolvable situations. Where interacting concavities are detected they are initially avoided by replacing one of the concavities with the relevant part of the convex hull of the shape. The convex hull substitution is then returned to later in the algorithm. One major advantage of this technique is that it does not require any form of decomposition, itself a potentially complex problem, such as the work described in (Li & Milenkovic, 1995).

(Dean et al., 2006) offers an extension to the work of (Ghosh, 1991) and the subsequent paper by (Bennell et al., 2001). The extensions improve on the speed of generation of no-fit polygons using the Minkowski sum approach for large complex instances like those often found in the garment industry. Whilst the authors did not test their approach on any of the literature benchmarks as they were too simple to exploit their extension they do report a speed up of 64% for no-fit polygon generation time using a divide and conquer technique.

(Huyao et al., 2007) present a new approach for generating the NFP that is simple, intuitive and computationally efficient. It is based on the novel concept of trace line segments that are derived from the interaction of two-component polygons. The complete set of the trace line segments contains all the boundary edges of the NFP and internal points that need to be discarded. Algorithms for deriving the trace line segments, efficiently determining those segments that form the boundary of the NFP and the identification of holes and degenerate cases are described.

In (Bennell & Song, 2008) the authors extend the work described in (Bennell et al., 2001) on the generation of no-fit polygons using a Minkowski sum technique. The authors develop the approach to remove degenerate cases from the no-fit polygon generation procedure. This is achieved by the addition of two post Minkowski edge list generation steps designed to remove edges that are not part of the no-fit polygon. Firstly the updated procedure removes the negative Minkowski sum edges as these cannot represent a sliding boundary between the shapes. Secondly the authors use a concept of a polygonal trip which allows the identification of the no-fit polygon boundary, without using this trip technique can be very complex to detect. This is due to the fact that the remaining Minkowski edges can represent a complicated self-crossing polygon. Additionally the authors present a method for solving a previously degenerate can caused by "jigsaw" pieces. The approach involves looking for cycles of paths in the Minkowski edges which can be identified and allow the authors to detect the single point of touch to be added to the no-fit polygon. The authors present computational benchmarks for a range of literature problems and for a set of specifically designed problems involving degenerate cases. For all of these problems the improved algorithm is able to generate correct no-fit polygons in less than one second. This is the first other example of a complete and robust no-fit polygon generation technique for irregular concave and convex polygons in the literature, following the algorithm presented in chapter 5 of this thesis.

3.4.6. Orbital Sliding Approach

(Mahadevan, 1984) introduced a method for generating no-fit polygons through the technique of edge sliding. By utilising D-Functions (Konopasek, 1981) the author produced a simple algorithm that is able to produce no-fit polygons for pairs of non-convex shapes. Later chapters in this thesis discuss the work performed to eliminate the degenerate case from Mahadevan's algorithm and extend it to allow the inclusion of shapes containing arcs as part of their boundary definition.

The orbital sliding approach involves using standard trigonometry to physically slide one polygon around another. The no-fit polygon is defined by tracing the motion of a locus point of the sliding polygon when orbiting. The first discussion and implementation of an orbiting approach for the generation of 'envelopes' is detailed in (Mahadevan, 1984).

The key elements of Mahadevan's approach are: calculation of touching vertices and edges, determination of the translation vector and calculation of the translation length. The calculation of intersecting edges is performed using the notion of D-functions introduced in

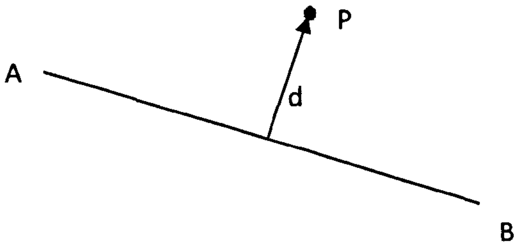
(Konopasek, 1981), see the next section for further description of D-functions. Mahadevan modifies the D-function test to also calculate touching points which is a necessity for both Mahadevan's algorithm and the new approach outlined in chapter 5. This information is then used to select a translation vector based on the touching edge. The translation vector is then projected through each vertex of the orbiting shape and then the intersecting edge testing is used to calculate the translation distance. It is also important to project the translation vector in the reverse direction of the stationary polygon. The orbiting shape is then translated along the translation vector by the smallest distance (from projection and intersection tests). This ensures the two polygons never intersect but always touch. The process continues until the orbiting polygon returns to its original starting position. The major disadvantage of Mahadevan's original algorithm is that it cannot generate the full no-fit polygon for shapes involving holes or some concavities. The problem occurs when the orbiting polygon can be placed inside a concavity of the stationary polygon but the concavity has a narrow entrance. In this case the orbiting polygon is too wide and will simply slide over the concavity. (Oliveira et al., 2000) also use Mahadevan's implementation within their work on irregular packing and include steps to detect such problems. Chapters 5 and 6 present extensions and improvement to this sliding technique that allow this approach to robustly generate no-fit polygons.

3.4.7. D-Functions

Given that we have been able to generate a geometry library using the techniques outlined in section 3.1 capable of accurately detecting intersection and touching between polygons, and the shapes they form; it is useful to begin to build some low-level logical operations from which we can build more powerful operations, such as the generation of the no-fit polygon.

(Konopasek, 1981) introduced the concept of D-Functions which allow the determination of the relationship between pair of line segments which can be useful when determining feasible moves whilst generating the no-fit polygon using a sliding approach, as described in chapter 5. At each point of the generation decisions must be made about which edge to traverse, D-Functions are a key tool in making that decision. The following section outlines the functionality of D-Functions as utilised by Mahadevan in no-fit polygon creation.

The fundamental building block of the D-Function is the distance from a point to a line, shown in Figure 12.



$$d = \frac{(X_A - X_B)(Y_A - Y_P) - (Y_A - Y_B)(X_A - X_P)}{\alpha}$$

Where

$$\alpha = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2}$$

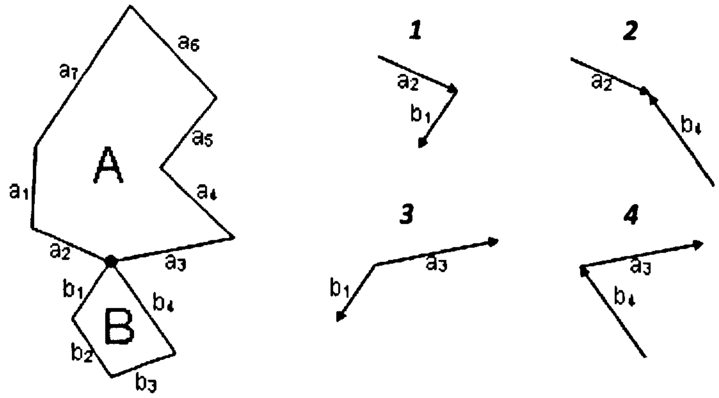
Figure 12. Distance of a Point from a Line

Therefore given vector AB and the point P the D-Function:

D_{ABP} = sign[(XA - XB) (YA - YP) – (YA - YB) (XA - XP)]

allows us to determine which side of the vector the point P is. The use of the *sign* function means that the value returned from the D-Function can only be -1, 0 or +1 dependent upon the result being respectively negative, zero or positive.

If the result of D_{ABP} is zero the point P is co-linear with the line segment AB, although it may not be on the segment AB, when the result is negative P is right of AB and when the result is positive P is left of AB.



1	2	3	4
a2 and b1 touch because	**a2 and b4 touch because**	**a3 and b1 touch because**	**a3 and b4 touch because**
$D_{[a2][b1Start]} = 0$	$D_{[a2][b4Finish]} = 0$	$D_{[a3][b1Start]} = 0$	$D_{[b4][a3Start]} = 0$
$D_{[b1][a2Finish]} = 0$	$D_{[b4][a2Finish]} = 0$	$D_{[b1][a3Start]} = 0$	$D_{[a3][b4Start]} = 0$
$D_{[b1][a2start]} = 1$	$D_{[a2][b4start]} = -1$	$D_{[a3][b1Finish]} = -1$	$D_{[a3][b4start]} = -1$
And a2 is right of b1	And a2 is left of b4	And a3 is left of b1	And a3 is right of b4

Figure 13. Touching Shapes and D-Functions

Figure 13 shows two shapes touching at some point, which is a very common situation as when no-fit polygons are generated using the sliding technique. In order to determine the state of the lines with respect to one another D-Functions are vital tool. The use of D-Functions here gives the information that the lines are touching and their orientation with respect to one another. This information informs the sliding process logic. A full description of a complete and robust sliding no-fit polygon generation technique can be found in chapter 5 of this thesis.

3.5. Optimisation and Search Methods

This section presents a review of search methods used in numerous literature approaches to the automatic nesting problem, including the Hill Climbing and Tabu Search approaches used in this thesis.

3.5.1. A*

The A* heuristic search algorithm (Hart et al., 1968) uses domain knowledge to search for solutions to the applicable search space. As a greedy algorithm the uses its domain knowledge to make the best possible move at each search step, however this can lead to poor quality solutions as poor moves that were previously discounted must be made towards the end of the search. As the search proceeds the exact cost of the solution generated thus far is recorded and added to the estimated cost, via a relevant heuristic, to derive a total estimate for the final cost of the solution being generated.

Providing that the estimation heuristic never underestimates the cost (estimation heuristics displaying this characteristic are called *admissible*) the A* algorithm is provably able to produce optimal solutions. However for most problems the number of possible solutions that must be generated in order to find the optimal solution rises exponentially.

Although not widely used in the nesting literature, likely due to the potential for poor solutions and the vast number of solutions required in order to find an optimal solution, a notable use of the A* algorithm is in the work of (Albano & Sapuppo, 1980).

3.5.2. Local Search Methods

Local search methods are the most widely represented group of search strategies in the irregular cutting and packing literature. These approaches often maintain a single best solution and move from that state to a neighbouring state, defined by some operator or set of operators, in order to try and find a superior solution.

3.5.2.1. Hill Climbing

The hill climbing algorithm is a simple search method that starts with a random point in the search space and generates one or more neighbourhood solutions. If one of the generated states improves on the current solution (and out performs other generated solutions) then that neighbour is adopted as the current solution and the search continues. If no improved neighbour solution is found the generated solutions are discarded and the search continues without changing the current solution. This leads to one of the major pitfalls of the hill climbing approach, it is possible for the technique to get stuck in local optimum without any means of escaping from this area. This is due the lack of provision for accepting worse solutions or backtracking. One method of avoiding this problem is to allow a random restart of the search if no improved solution has been generated for a specific number of iterations. In this situation the search will move to another completely random start point in the search space and continue with the search process. Examples of the hill climbing algorithm can be found in (Rich & Knight, 1990) and (Russell & Norvig, 2009).

The hill climbing algorithm is used to generate solutions in chapters 4 and 6 of this thesis. Other practitioners have also used this technique. A selection of literature approaches utilising this technique can be found in (Hopper & Turton, 2001), (Prasad et al., 1995), (Jain et al., 1990) and (WenQi et al., 2001).

3.5.2.2. Tabu Search

The tabu search algorithm is an extension of the hill climbing technique outlined above. The major addition to the hill climbing approach is to keep a list of states already visited in the solution space.

This memory of previously visited solutions allows the search to avoid returning to a previous solution for a predetermined number of search iterations. This ensures that the search will explore new states in order to potentially find improved solutions. An additional difference from the hill climbing technique is the constant adoption of the best solution found from the neighbourhood of generated solutions, whether it represents an improvement on the current solution or not, providing it is not in the tabu list. This allows the search to escape local optima and continue exploring the search space throughout the process.

The algorithm was originally proposed (Glover, 1977) but was not widely adopted for solving combinatorial problems until considerably later. Indeed the major references for the technique are regarded as (Glover, 1989), (Glover, 1990) and (Glover & Laguna, 1998).

The tabu search approach is used in the cutting and packing literature, examples of its use can be found in (Blazewicz et al., 1993), (Blazewicz & Walkowiak, 1994) and (Kido et al., 1994) .

Additionally a tabu search variant called tabu thresholding was used in (Bennell & Dowsland, 1999), the tabu threasholding variant technique was introduced in (Glover, 1992). Later in (Bennell & Dowsland, 2001) the authors presented a hybridised tabu search approach based on their previous thresholding approach but, as shape overlap is an acceptable but penalised state, adapts it further to avoid convergence upon infeasible solutions via an additional optimisation stage.

The tabu search technique is used in chapters 3 and 5 of this thesis to generate shape input sequences for the bottom left fill layout process.

3.5.2.3. Simulated Annealing

Annealing is a physical process, often used in the metal cutting industry, to change material properties such as hardness or remove internal stresses. These stresses can often be caused by the heat of the cutting process, especially on thicker materials. The technique involves heating and cooling the material on a predefined schedule to reorganise the internal crystalline structures until they will display the required properties. The simulated annealing search process takes inspiration from this physical process using the concept of energy (heat in metal annealing) and a cooling schedule to control the acceptability of neighbourhood solutions during the search process. The publication of the ideas that later gave rise to the technique can be found in (Metropolis et al., 1953) however it was not until 1983 that Kirkpatrick, Gelatt and Veechs introduced the concept of using the technique to control a combinatorial search process (Kirkpatrick et al., 1983).

Like hill climbing the process generates one solution at a time and automatically adopts it if it is better than the current solution. However in order to allow escape from local optima a worse solution may be adopted according to some probability. This probability is controlled by two factors, the difference between the solution quality of the current best solutions and the generated candidate solution (change in energy, called ΔE) and the length of the search time remaining (equivalent to the temperature T at a given point of the cooling process in the physical process). This probability is usually expressed in the formula $e^{(-\Delta E/T)}$.

The cooling schedule is often linear or geometric in shape and the temperature reduction is made after a certain number of iterations at a given temperature. This means that the probability of selecting a worse solution early in the schedule is higher as the temperature element allows the system to accept higher energy difference solutions. As the schedule progresses and the systems temperature reduces the probability of selecting a worst solution is lowered and when the temperature reaches zero, if it is allowed to, the search will act as a hill climb search only accepting better solutions.

The use of simulated annealing as a search mechanism in the cutting and packing literature is quite extensive, examples of its use can be found in (Jain et al., 1990), (Marques et al., 1991), (Oliveira & Ferreira, 1993), (Dowsland & Dowsland, 1993), (Han & Na, 1996), (Wu et al., 2003), (Gomes & Oliveira, 2006) and (Martins & Tsuzuki, 2010).

3.5.2.4. Beam Search

Beam search is a restricted best first search wherein a limited number of nodes will explored at each level of the graph using a breadth first search to build the candidate nodes. The node with the best partial evaluation will then be explored. The beam width controls the number of states stored at each level. Whilst this helps to restrict the memory overhead and reduce search time to generate solutions it does so at the expense of potentially not finding the optimal solution. The beam width can be fixed or variable, in the variable configuration the beam width is controlled by a threshold based on the current best solution.

This approach was used by (Bennell & Song, 2010) to generate good quality solutions in fast execution times. Beam search was originally used for speech recognition systems in (Lowerre, 1976) and has subsequently been applied to numerous optimisation problems.

3.5.3. Ant Algorithms

Ant algorithms are based on the study of ant behaviour, although ants are almost entirely blind they are able to navigate between their nest and food sources in the surrounding area using the shortest route, this is achieved using pheromone trails which are strengthened by repeated use of a path and degrade over time.

The use of ant algorithms to generate solutions to the travelling salesman problem (TSP) was first attempted in (Dorigo et al., 1996). The authors outlined how this is achieve by use of the following example. Imaging an initial case in which the ants need only travel in a straight line from their nest to a food source. We then imagine that an obstacle is placed in the way of this direct path. When the first ant arrives at the obstacle the ant must decide to turn left or right and without a pheromone trail to follow the there is an equal probability that it will turn left or right. If we assume that the left path is shorter than the right path then any ant taking the left will path will get to the food source faster and therefore return along that route sooner, strengthening the pheromone levels on that path. The stronger pheromone path is more attractive to other food gathering ants and therefore via a positive feedback loop the shortest path is considerably more attractive to ants gathering in the future.

The Dorigo et al. report that the ant algorithm outperformed general purpose heuristics like tabu search and simulated annealing on the literature TSP benchmarks. Furthermore the ant algorithm approach produced solutions of quality competitive to those of specialised TSP approaches although the ant algorithm took longer to find the solution.

Ant algorithms have been applied to cutting and packing problems in (Burke & Kendall, 1999), (Kendall, 2000), (Burke & Kendall, 2002) and (Levine & Ducatelle, 2004).

3.5.4. Genetic algorithms

First introduced by Fraser in (Fraser, 1957) and Bremermann in (Bremermann, 1958) then later seminally covered in (Holland, 1975) genetic algorithms are population based search methods inspired by evolutionary processes found in nature.

A genetic algorithm holds a population of solutions known as chromosomes; each chromosome is composed of several sub-parts called genes. The value assigned to a gene is known as its alleles the combination of these genes and their values makes a chromosome a potential solution to the problem being tackled. In order to facilitate the genetic algorithm process it is important that each solution in the population is evaluated, this evaluation is the population member's fitness.

The process of search using the genetic algorithm involves developing new generations of solutions from the existing population. The new population generation process is performed two types of operator which are designed to mimic evolutionary processes. The crossover operator is akin to the process of breeding in natural systems, two individuals in the population are selected using some defined strategy and exchange elements of their genes in order to produce child members of the new population. Most selection strategies tend to focus on the selection of the fitter members of the population as candidates for breeding selection, the hope being that they will combine their strengths in order to produce an improved child for the next generation. Additionally the genetic algorithm uses a mutation operator which can randomly change members of the population in order to maintain population diversity and increase the possibility of reaching different areas of the solution space. Generally the specific strategies and operators used in implementations of the genetic algorithm search technique are specially designed for the problem being tackled.

Detailed analysis is of genetic algorithms is available in (Goldberg, 1989) and the original papers by Bremermann and Fraser are revisited in (Fogel & Anderson, 2000) (Fogel & Fraser, 2000).

During the early stages of the CASE / TCS project the use of genetic algorithms generating sequences for packing with a bottom left fill placement strategy was explored. However due to the memory overhead and speed of result generation the implementation was unacceptable to the industrial partner and was therefore discounted.

The use of genetic algorithms in the cutting and packing literature is extensive examples can be found in the following publications:

(Fujita et al., 1993), (Dighe & Jakiela, 1996), (Jakobs, 1996), (Bounsaythip & Maouche, 1997) , (Cheng & Rao, 1997), (Anand et al., 1999), (Lui & Teng, 1999), (Ramesh Babu & Ramesh Babu, 1999), (Hopper, 2000), (Ramesh Babu & Ramesh Babu, 2001), (Tay et al., 2002), (Crispin et al., 2003), (Yeung & Tang, 2003), (Hifi & M'Hallah, 2004), (Wong et al., 2009), (Yang & Lin, 2009)

The majority of the approaches noted above use the genetic algorithm to solve the combinatorial optimisation element of their nesting approaches. In order to do this a chromosome generally represents a full sequence of shapes for placement, i.e. each gene is a shape to be placed. The sequence in the chromosome is often used to drive a sequence based placement algorithm where the quality of the generated solution is used to determine the fitness of an individual. In these examples the chromosomes are manipulated by numerous different sequence manipulating operators in order to generate varied members of the population, following the normal genetic algorithm approaches.

A smaller number of the noted authors have encoded more placement related information into their chromosomes such as placement angles and other factors such as edge distance for placement (Tay et al., 2002) or degree of allowed deviation from an area of required quality in the case of (Crispin et al., 2003). Only in (Dighe & Jakiela, 1996) do the authors use a hybrid genetic algorithm approach that is capable of producing full layouts. Many of the approaches outlined above are discussed in more detail in chapter two.

3.5.5. Artificial Neural Networks

Artificial neural networks have not been widely used in the field of irregular cutting and packing problems although they are marginally better represented in the orthogonal variants of the problem. Where they are represented in the cutting and packing literature they are usually used in combination with other techniques.

A neural network is formed from a set of simple neurons. When operating the neurons receive a weighted input signal and depending upon an activation level threshold the neuron may output a value, if the output occurs this is known as the neuron firing. Inputs can be in two forms, excitatory or inhibitory, the excitatory inputs contribute to the activation level threshold whilst inhibitory inputs will reduce the threshold. By building networks out of neurons it is possible to create complex networks of behaviour which is presented to the outside world through output nodes. Input nodes receive information from the outside world and these signals propagate through the system firing various neurons which will in turn fire further neurons.

The seminal study of neural networks can be found in (McCulloch & Pitts, 1943) a useful guide to the field can be found in (Hassoun, 1995).

The following papers in the cutting and packing literature utilise an artificial neural network:

(Dagli, 1990) uses a neocognitron neural network paradigm to generate data for assessing the degree of match between two irregular patterns. The information generated through the feature recognition network is passed to an energy function, and the optimal configuration of patterns is computed using a simulated annealing algorithm. Similar approaches are explored in (Poshyanonda & Dagli, 1992) and (Poshyanonda & Bahrami, 1992).

In (Han & Na, 1996) an artificial neural network model is used to generate rectangular pattern configurations. The patterns generated by the neural network are represented as decision variables of a mathematical programming model.

CHAPTER FOUR

4. A New Bottom-Left-Fill Heuristic Algorithm for the Two-Dimensional Irregular Packing Problem

4.1. Introduction

This chapter presents a new heuristic algorithm for the two-dimensional irregular stock cutting problem, which generates significantly better results than the previous state of the art on a wide range of established benchmark problems. The developed algorithm is able to pack shapes with a traditional line representation, and it can also pack shapes that incorporate circular arcs and holes. This in itself represents a significant improvement upon the state of the art. By utilising hill climbing and tabu local search methods the proposed technique produces 25 new best solutions for 26 previously reported benchmark problems drawn from over 20 years of cutting and packing research. These solutions are obtained using reasonable time frames, the majority of problems being solved to new benchmarks within 5 minutes.

In addition to this, the work also introduces 10 new benchmark problems which involve both circular arcs and holes. These are provided because of a shortage of realistic industrial style benchmark problems within the literature and to encourage further research and greater comparison between this and future methods.

The speed of the solution generation of this technique and its applicability to industrially complex shapes of any possible cut valid configuration made this approach a valuable technique in the context of the CASE/TCS projects. As the technique it was able to keep pace with Esprit's existing techniques whilst increasing the layout accuracy using the geometry of the shapes in the problem this work proved to be a valuable first step in the research programme, furthermore it required the generation of a robust geometry library which Esprit had not previously had access to. This work was published in (Burke et al., 2006).

4.2. The Proposed Approach

The new method for implementing a bottom-left fill packing strategy utilises new shape primitive overlap resolution techniques. These techniques allow the fast generation of high quality solutions by eliminating grid based inaccuracy in the vertical axis of the packing sheet. Furthermore, the proposed approach allows problems containing circular arcs and shapes with holes to be nested. By combining the new techniques with both hill climbing and tabu local search methods the algorithm is assessed against the 26 existing problems from the literature discussed in section 4.8 then introduces and set benchmarks for 10 new problems, 5 of which contain circular arcs and three that include shapes with holes.

There have been many different approaches for producing solutions for the irregular two-dimensional stock cutting problem. In general, the approaches that have achieved the best known results have used a no-fit polygon based technique to generate potential placement positions and/or test for overlaps e.g. (Gomes & Oliveira, 2002), (Gomes & Oliveira, 2006), (Egeblad et al., 2007)and (Imamichi et al., 2009). While the no-fit polygon is a powerful geometric technique, there are several issues that make it limited in scalability for industrial applications. No-fit polygon techniques are notorious for the large quantity of degenerate cases that must be handled in order to be completely robust. There are several well-known cases for which no-fit polygon algorithms *can* fail which include the following: hole filling, shapes with concavities and also *jigsaw* type shapes where one shape fits exactly into a concavity from another shape. Later chapters of this thesis cover the formulation of complete and robust no-fit polygon generation techniques for line arc polygons however, at the point this work was generated, there was not yet any implementations of the no-fit polygon that could successfully handle true arc geometry.

Without the ability to generate no-fit polygons including arcs most methods and benchmark problems have approximated arcs using a sequence of lines. The problem with approximations is that there is an *accuracy to time* trade-off. That is to say, that if we model the arc with fewer lines then we reduce the accuracy of the approximation but the shape is less complex and if we increase the quantity of lines we then improve accuracy but make the resultant shape more complex. When performing translation or rotation operations, obviously shapes with a larger number of lines will take longer to manipulate than a shape with fewer lines. Ultimately, there will always be inaccuracy when modelling arcs as a series of lines and in adding more lines to our approximation we increase the time required to operate on that approximation.

The geometry implementation used throughout this work is able to model shapes composed of both lines and non-approximated arcs and is able to conduct fast shape intersection operations to an accuracy of 10^{-7} metres, some of the considerations and practicalities of developing such a library are discussed in more detail in section 3.1.

In real world industrial settings, and especially for profiling (sheet metal cutting) within the steel cutting industry, it is imperative that we are able to handle arcs, concavities and holes. Traditionally, industry has used a bottom-left-fill approach based on an iterative grid approximation in which arcs are represented by approximated lines. The grid approach aims to reduce the infinite number of potential positions (due to the continuous nature of space) to a fixed set of potential locations. The algorithm works as follows: when placing a shape on the sheet we try the first grid location (bottom-left point) and then check for intersection with shapes already assigned to the sheet. If there are no intersecting shapes then the shape is assigned to the sheet in its current position and we start from the first grid location with a new shape. However, if the current shape does intersect with another shape on the sheet then it is moved to the next grid position and tested for intersection again, continuing the process until a valid position is found. It is obvious that this also means that using a lower resolution grid will, in the general case, adversely affect the quality of the solution because shapes are placed in later positions than they could otherwise be placed if a higher resolution grid was used, this causes an *accuracy to time* trade-off.

The approach that is described in the following sections is not restricted to moving shapes by a fixed translation when intersecting with another shape, unlike the iterative grid approach. This is achieved by using the underlying geometric primitives, i.e. line segments and circular arc segments, of intersecting shapes to resolve the overlap. This has two main advantages in that we can resolve intersecting shapes so that they touch exactly and, secondly, that this accuracy is achieved in a smaller number of steps than the iterative grid mechanism. Although the grid method and our proposed approach follow a similar conceptual procedure when placing shapes, our proposed approach has both a faster and more accurate method of producing solutions.

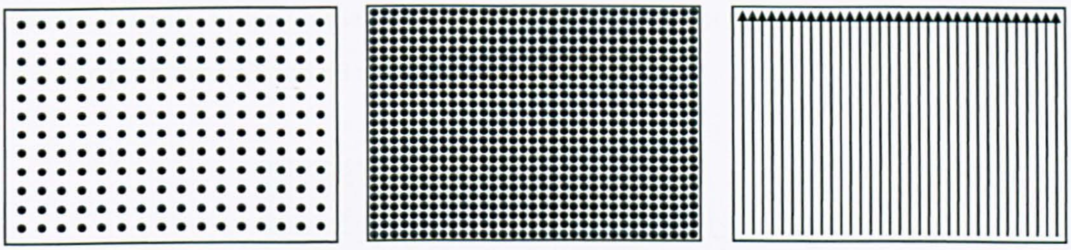


Figure 14 **a)** low resolution grid approach **b)** high resolution grid approach **c)** variable shift approach

Figure 14 shows the potential locations for two iterative grid approaches with differing resolutions and also our proposed overlap resolution method. In the two iterative grid approaches there are a finite number of locations for which shapes may be placed. The iterative grid approach of Figure 14b has a higher resolution than that in Figure 14a and therefore has more potential placement locations and should result in more compact packings. The variable shift approach of Figure 14c has an infinite number of potential locations due to its continuous y axis property and therefore has more chance of producing compact packings.

4.3. Geometrical Definitions

In order to illustrate the new packing method it is necessary to firstly define what constitutes a "primitive", "loop" and "shape". For the purposes of this discussion, a "primitive" is defined as either an arc or a line. A line is represented by its start and end points whereas an arc is circular and is represented through a centre point, radius, start angle and offset angle. A "loop" is defined as an anti-clockwise closed list of primitives where each primitive's end point is the start point of the next primitive. The technique does not allow for non-simple polygons, as defined in (O'Rourke, 1998), given the industrial nature of the problems being tackled only shapes that can be physically cut from a sheet of material are of relevance. A shape is defined as one outer loop and 0….*n* internal loops which can be thought of as holes in the shape. Most of the problems from the current literature do not include shapes with either arcs or holes and numerous examples only contain convex shapes, where it is easier to detect overlaps. Furthermore, it is necessary for these algorithms to cope with floating-point data in order to establish high accuracy and realism on real world problems. The geometry library used can cope with these extra complications. Figure 15 demonstrates an instance of overlap between two shapes, A and B. It is clear that the arc primitive a2 intersects with line primitive b4 and that the line primitives a3 and b3 also intersect.

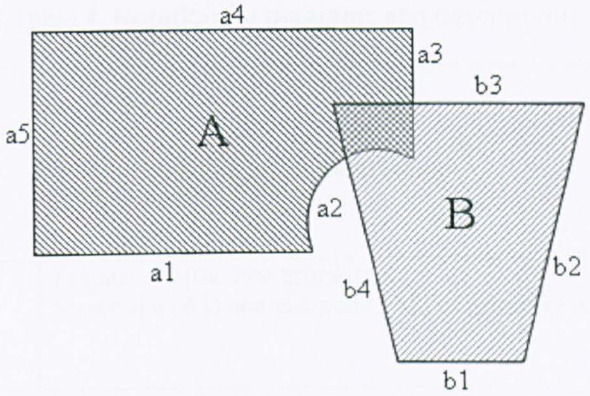


Figure 15. Overlapping Shapes

4.4. Resolving Overlapping Primitives

There are four possible cases that must be handled to resolve intersecting primitives. One of these primitives is part of the shape that we are trying to place, termed the "free shape", and the other is part of a shape that has already been placed on the sheet, named the "locked shape". The techniques described in the following sections involve calculating the positive vertical distance required to translate the free shape such that the two primitives no longer intersect. Whilst this resolves the overlap between the two intersecting primitives, the two shapes may still not be fully resolved in that other primitives belonging to the shapes may also intersect or the free shape may be entirely contained in the locked shape (discussed in section 4.5). However, repeated application of these techniques will always resolve overlapping shapes with the smallest positive vertical distance required. There are four intersection cases which must be handled: i) two lines, ii) line and arc, iii) arc and line, iv) two arcs. It should be noted that throughout all of the cases, the locked shape's intersecting primitive, called the "locked primitive" throughout this chapter, has already been assigned to the sheet and its position may not change whereas the intersecting primitive belonging to the shape that is being placed is termed the "free primitive". The *x* span of a primitive can be thought of as the horizontal span of its bounding rectangle. Another concept used is an "infinite vertical line". This is a vertical line that spans from negative infinity to positive infinity along the *y* axis. The notations used within the diagrams and descriptions of the following subsections are presented in Table 4.

Table 4. Notation for diagrams and descriptions

Symbol	**Description**
A1→A2 **A1, A2**	Primitive A (the free primitive). Start point (A1) and end point (A2) of primitive A
B1→B2 B1, B2	Primitive B (the locked primitive). Start point (B1) and end point (B2) of primitive B
CP	Centre point of an arc primitive
c1…cn	Intersection points
t1, t2	Tangent points of a line on an arc
⋮	Infinite vertical line (short-dashed vertical line)
– – – – – –	Perpendicular to a line primitive (long-dashed line)
↑	Translation used to resolve overlap (bolded vertical arrow)

4.4.1. Line & Line (Free Line moving through Locked Line)

In order to resolve any two intersecting lines it is necessary to the find the end points of each line, A and B, that are within the *x* span of each other, these points are known as the points in range (*pir*). By passing an infinite vertical line through each of the *pir* originating from line A and find the intersection points of these lines with line B. The distance between each *pir* from line A and its corresponding intersection point on line B is calculated using formula [1].

$$\mathit{distancePirA} = \mathit{intersectionPointB.y} - \mathit{pirA.y} \qquad \mathbf{[1]}$$

In order to pass infinite vertical lines through each of the *pir* originating from line B to find their intersection points with line A a different formula is required. Formula [2], is used to calculate the distances when the *pir* originates from the locked primitive (line B).

$$distancePirB = pirB.y - intersectionPointA.y \qquad \textbf{[2]}$$

For the resolution method, the overlap must always be resolved by translating line A in the positive vertical direction. The distance formulae, given in formulas [1] & [2], may yield negative results, therefore, these results are not valid positive vertical movements and are eliminated. These distance formulae also form part of the strategy for resolving other cases. For intersecting lines, there always exists one valid positive result which can be used to vertically translate line A, thus resolving the overlap. An example of this approach is shown in Figure 16.

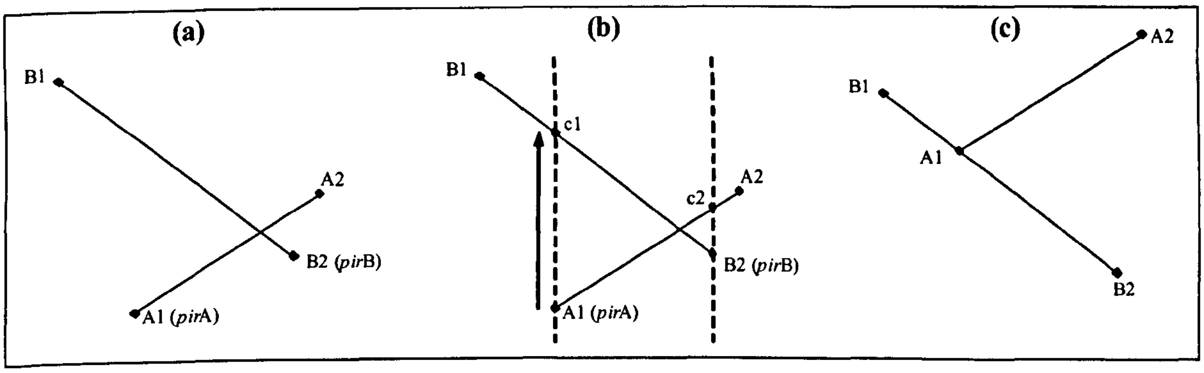


Figure 16. Resolution of intersecting line primitives

Here, point A1 is within the *x* span of B1 → B2 and B2 is within the *x* span of A1 → A2. These points are labelled *pir*A and *pir*B respectively (see Figure 16a). If an infinite vertical line is passed through *pir*A to create intersection point c1 and through *pir*B to create intersection point c2 (shown in Figure 16b). The distance between *pir*A and c1 is calculated using formula **[1]** and the distance between *pir*B and c2 is calculated using formula **[2]**. In this example, the first distance yields a positive result whilst the second is negative. Formula **[3]** shows how the technique combines the distance formulae 1 & 2 into a "max" function call:

$$yTranslation = \max(c1.y - pirA.y, pirB.y - c2.y) \quad \textbf{[3]}$$

The result of formula [3] is shown pictorially as the bolded arrow in Figure 16b. Figure 16c shows how the overlap has been resolved by vertically translating line A by this positive translation. In practice, all of the primitives of shape A are translated, not just the line involved in the overlap.

4.4.2. Line & Arc (Free Line moving through Locked Arc)

In this case, where a line is intersecting with an arc, it is necessary to find the positive vertical translation with which the line should be translated in order to completely resolve its intersection with the arc. As with the Line & Line case (section 4.4.1), it is possible to utilise the points in range of each primitive. However, because an arc is involved, it may be required that tangent points are used between the arc and line primitives. Figure 17 shows an example where applying the points in range method alone is not sufficient to resolve the overlap.

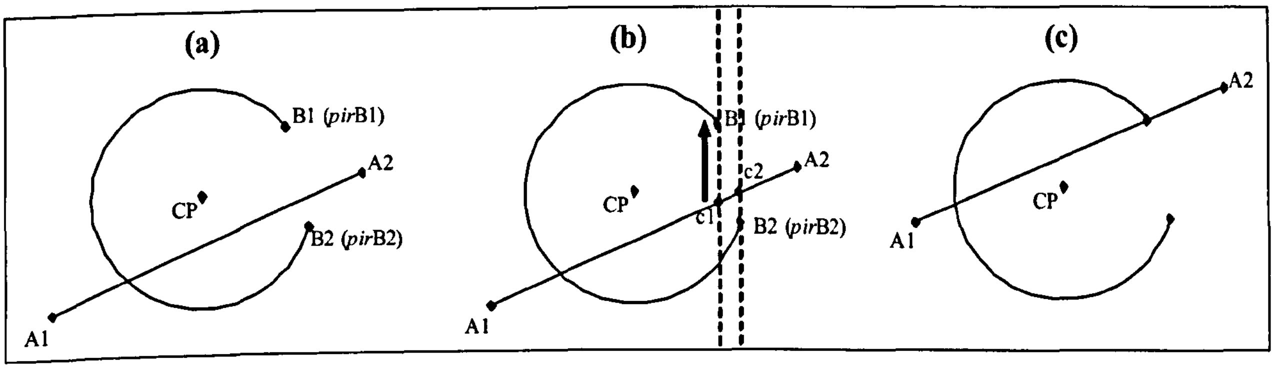


Figure 17. A line arc example where points in range are insufficient to resolve overlap

Figure 17a shows that the only points within range are B1 (*pir*B1) and B2 (*pir*B2) from the arc (both end points of the line, A1 and A2, are outside the *x* span of the arc and, therefore, are not in range). Once again, an infinite vertical line is passed through each *pir* and is intersected with the line A1→A2. This creates intersection points c1 from *pir*B1 and c2 from *pir*B2 (see Figure 17b). The distance between each *pir* and its respective intersection point on the other primitive is calculated using the distance formulae [1] & [2]. In this example, both *pir* originate from the locked primitive (arc B) and therefore both distances are calculated using formula [2]. This yields one positive result that is shown by the bolded arrow in Figure 17b. Figure 17c shows that this vertical translation is not sufficient to resolve the overlap. Figure 18 shows how to resort to the tangent points to fully resolve the overlap.

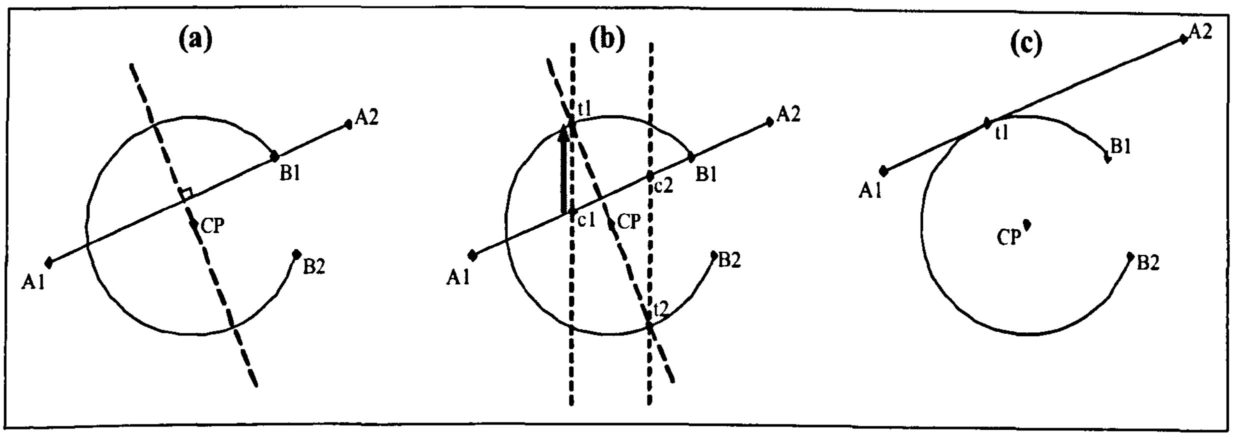


Figure 18. Using tangent points to resolve overlap with the line / arc case

The tangent points can be found by translating the perpendicular (or "*normal*") of the line such that it passes through the centre point of the arc, CP, as shown in figure 5a. The intersection of the perpendicular with the arc gives the tangent point(s), t1 and t2 (see Figure 18b). These tangent points are then used in a similar manner to the point in range technique in that infinite vertical lines are passed through each tangent point and intersected with line A1→A2 to give points c1 and c2. The translation distances can then be calculated by substituting each tangent point, t1 and t2, for *pir*B into formula **[2]**. This gives formula **[4]**.

$$distanceTangentB = tangentB.y - intersectionPointA.y \quad \textbf{[4]}$$

In the example, it can be seen that t1 would yield a positive translation whereas t2 would give a negative translation distance. Therefore, translating the line A1→A2 by the distance given by t1 will resolve the overlap (see Figure 18c). It should be noted that if the intersection of the perpendicular line with the arc yields no tangent points or the tangent points result in negative translation distances using formula **[4]** then the point in range technique *must* be able to resolve the overlapping primitives.

4.4.3. Arc & Line (Free Arc moving through Locked Line)

This case, an arc moving through a locked line, involves a similar approach to the free line & locked arc case. Once again, the same technique for points in range applies and, therefore, will not be repeated here. However, because the arc is now the free primitive (arc A1→A2) and the line is now the locked primitive (line B1→B2), we must substitute calculated tangent points and their intersections into formula [1] instead of formula [2] (as was the previous case in section 4.4.2). An example of this is shown in Figure 19.

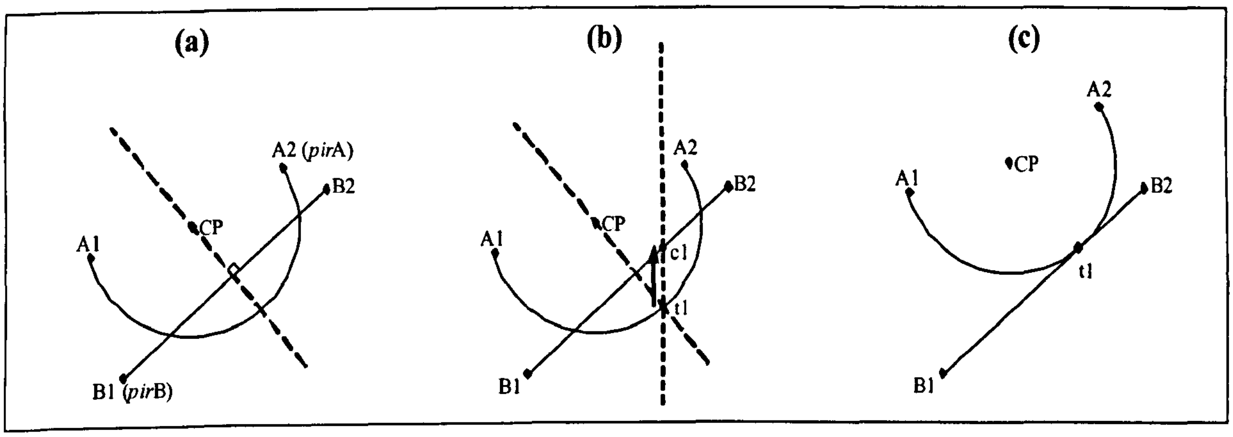


Figure 19. Resolving overlap using the tangent point method with the arc / line case

Figure 19a shows that points A2 and B1 are the points in range, *pir*A and *pir*B. However, both of the *pir* produce negative translations (using formulae [1] and [2]) thus they cannot be used to resolve the intersection therefore the tangent points method is utilised once again. In the example, only one tangent point is found because the perpendicular line only intersects the arc in only one place. Figure 19b shows that an infinite vertical line is passed through the tangent point, t1, and is intersected with line B1→B2 to produce point c1. The translation distances can then be calculated by substituting tangent points for *pir*A in formula [1]. This gives formula [5].

$$distanceTangentA = intersectionPointB.y - tangentA.y \quad [5]$$

Using formula [5] in our example yields a positive translation distance as shown by the bolded arrow in Figure 19b. The intersection is resolved by translating the arc, A1→A2, by this vertical distance as shown in Figure 19c. Once again, if the tangents method does not find tangent points or does not yield a valid positive result then the points in range method *will* be able to resolve fully.

4.4.4. Arc & Arc (Free Arc moving through Locked Arc)

The arc through arc case initially uses the identical point in range technique. For the situation where the point in range method is unable to resolve the intersection between the two arc primitives, the use of two circle tangent methods that utilise the radii of the arcs and the Pythagorean theorem allow resolution. An example of intersecting arcs in opposite orientations is shown in Figure 20.

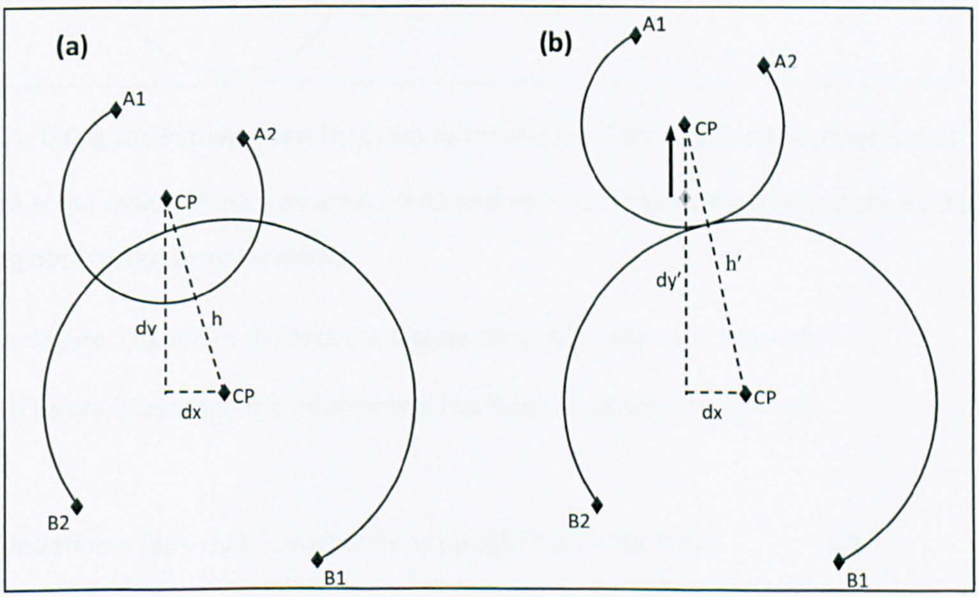


Figure 20. Using the Pythagorean Theorem to resolve arc / arc intersections (method 1)

Given that rA is the radius of the free arc A1→A2 and rB is the radius of the locked arc B1→B2 it is possible to make the following observations:

From Figure 20a, when the arcs are intersecting: $(rB - rA) < h < (rA + rB)$

From Figure 20b, when the intersection has been resolved: $h' = (rA + rB)$

Therefore:

$$\mathit{yTranslation} = (dy' - dy) \quad : \text{where } dy' = \text{sqrt}((h' * h') - (dx * dx)) \quad \textbf{[6]}$$

This intersection can then be resolved by translating arc A by the result of formula **[6].**

A further arc through arc case that may require solution involves two arcs of similar orientation as shown in Figure 21.

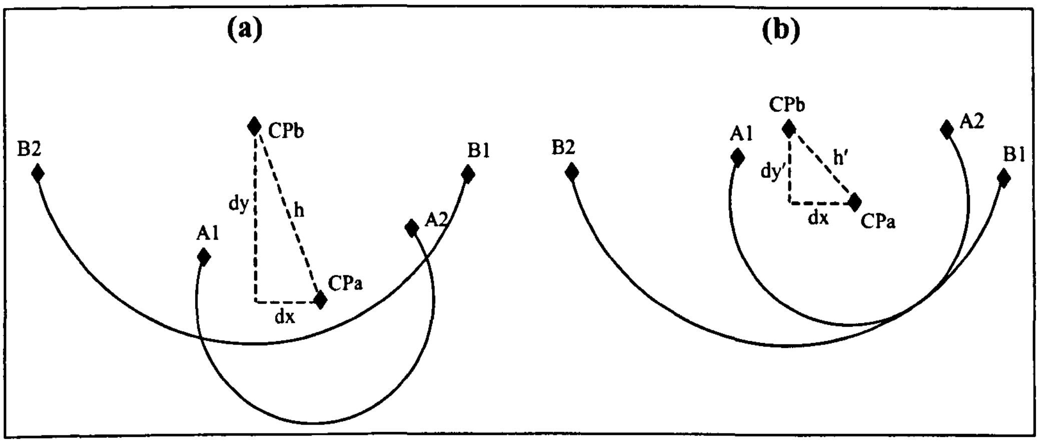


Figure 21. Using the Pythagorean Theorem to resolve arc / arc intersections (method 2)

Given that rA is the radius of the free arc A1→A2 and rB is the radius of the locked arc B1→B2 the following observations can be made:

From Figure 21a, when the arcs are intersecting: $(rB - rA) < h < (rA + rB)$

From Figure 21b, when the intersection has been resolved: $h' = (rB - rA)$

Therefore:

$$yTranslation = (dy - dy') \quad : \text{where } dy' = \text{sqrt}((h' * h') - (dx * dx)) \qquad [7]$$

If the result of formula [7] is positive, applying this vertical translation to arc A will resolve the overlap. It can be seen that, whereas the first circle tangent resolution method translates the free arc to the exterior of the locked arc circle, the second method translates the free arc to be inside of the arc circle. This is imperative for the correct manipulation of both convex and concave arcs.

4.4.5. Intersection Resolution Summary

The four possible intersection cases outlined above have shown that each case can be resolved by using the points in range method by using formulae [1] and [2]. This will always resolve the intersection where two lines are involved (see formula [3]). If arcs are involved, the point in range method may not be sufficient to resolve intersections fully and supplementary tangent based techniques may be employed. When an arc and line are intersecting by using perpendicular of the line primitive and displacing it through the arc's centre point it can be used at its intersection with the arc to find tangent points. Where the arc is the "locked primitive" and the line is the "free primitive" the use formula [4] will allow for the calculation the vertical translation required. Formula [5] is utilised when the arc is the "free primitive" and the line is locked. The final case, where two arcs are intersecting, introduced two circle tangent methods that can resolve intersections. The first method is calculated by formula [6] and results in the respective arc's parent circles being separate. The second method results in the respective free arc's parent circle being contained by the locked arc's parent circle (formula [7]). The least expensive of the cases is where two lines are involved, as no extra tangent calculations are required. This presents optimisation possibilities (which are included in the implementation) whereby, if there are many intersecting pairs of primitives between two shapes, the line only cases are resolved first. Although repeated applications of these techniques may be necessary to fully resolve the overlap between two shapes, this is more efficient and accurate than the iterative grid method outlined in section 4.2.

4.5. Contained Shapes

There now remains one special invalid case in which one shape is completely contained within another. This requires another resolution strategy as there are no intersecting primitives. During the nesting process it is possible that a shape may become entirely contained within other already assigned shapes. In this circumstance there are no intersecting primitives to resolve so another approach is required. As the free shape A is contained by locked shape B an infinite vertical line is cast up from the lowest point on shape A, lpA, through shape B. The resulting intersection of the infinite vertical line and shape B gives us points c1 ... cn. The translation we perform is defined by formula [8]:

$$\mathit{yTranslation} = \min(c1.y - \mathit{lp}A.y, \ldots\ldots\ cn.y - \mathit{lp}A.y)\ \text{ where : } (ci.y - \mathit{lp}A.y) > 0,\ i = 1\ldots\ n$$

[8]

Figure 22 shows an instance where employing this technique does not fully resolve overlap between the shapes. However, shape A is no longer contained by shape B and there are intersecting primitives once more allowing the techniques for resolving primitive intersection to be employed once more. This process continues until the shape intersection is completely resolved.

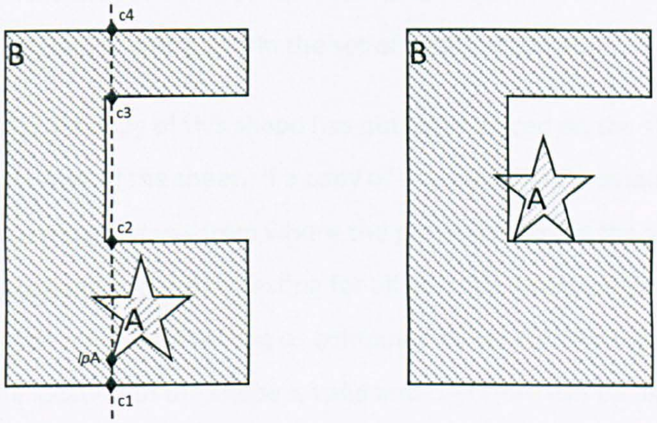


Figure 22. Contained shape where overlap is not fully resolved

In resolving shape intersections, the vertical translations employed may cause the free shape to intersect with other already assigned shapes. These intersections must also be resolved until the shape does not intersect and can be assigned to the sheet or until the shape has moved off the top of the sheet. In the latter case, the shape must be translated back to the bottom of the sheet in the next *x* coordinate and the process continues until all of the shapes have been placed.

4.6. Summary – The Bottom-Left-Fill Algorithm

This bottom-left-fill placement algorithm takes a sheet size and an input sequence of shapes and their allowable rotations. The algorithm progresses packing by placing the first shape in the lower left corner of the sheet in its most efficient orientation (the orientation that yields the smallest bounding rectangle width within the set of rotation criteria).

With subsequent shapes, if a copy of this shape has not been placed on the sheet, the shape starts at the lower left corner of the sheet. If a copy of the shape has previously been assigned to the sheet, then the new copy starts from where the previous copy of the shape was placed. A valid location for placement is found by testing for all possible intersections and containment. If the shape is not intersecting or contained by (or containing) other already placed shapes, then the location of the shape is valid and therefore can be assigned to the sheet.

When a shape is in a position that intersects with already assigned shapes, the resolution techniques described earlier in this section are used to resolve the overlap in a positive vertical direction (up the y axis of the sheet). As all possible intersections are computed the primitves that will produce the largest vertical resolution are selected. If resolving overlap results in the shape moving off the top of the sheet, then it is returned to the bottom of the sheet and is incremented along the positive x axis by a certain value (known as the resolution). The process continues as before with overlap/intersection tests and resolution until the shape does not intersect and can be placed. Packing is completed when all shapes have been assigned to the sheet and the solution can be returned to the user. Shapes are always packed in the order they appear in the input sequence.

The following pseudo code describes the bottom-left-fill process:

Algorithm 5. The bottom-left fill process

```
Input Sequence : shape[1..m][1..n] where   m = number of different shapes,
        n = number of rotations for given shape[m]
sheetshape[1..q] where array holds placed shapes on the sheet
q = total number of shapes assigned to the sheet
x = current x position
resolution = x increment step value

Begin

q = 1;
Place shape[1][1] at bottom left of sheet (0,0);   // place first shape
sheetshape[q] = shape[1][1];
q++;                                                        // increment shapes added counter

for (i = 2; i <= m; i++)                                    // start remaining shape placement
{
        for (j = 1; j <= n; j++)                            // pack each allowable rotation, j
        {
                place shape[i][j] at bottom left of sheet;

                while (Overlap(shapes[i][j], sheetshape[1..q]))     // find feasible position
                {
                        Get Intersecting Primitives of shape[i][j] and sheetshape[1..q];

                        Resolve Overlapping primitives;

                        if (shape[i][j] off top of sheet)
                        {
                                x = x + resolution;
                                place shape[i][j] at (x,0);
                        }

                }

                if (shape i in orientation j is least costly orientation seen so far)
                {
                        bestorientation = j;            // record best orientation seen so far
                }
        }

        sheetshape[q] = shape[i][bestorientation];  // assign shape i in best orientation to sheet
        q++;
}

Return Evaluation (total length of packing);

End
```

4.7. Local Search

It is usual to apply some sorting criteria to the shapes of a given problem before packing, often by decreasing length or area. Although these often yield solutions of reasonable quality, further improvements can be found if a local search mechanism is applied to generate new input orderings. This approach has been used during the experiments of section 4.12 by applying hill climbing and tabu search for both area and length pre-sorted arrangements. For additional information on the hill climbing and tabu search techniques see section 3.5.

A standard hill climbing algorithm is used during the experiments which simply applies operators to the current solution in order to find a neighbour of increased quality. For each cycle of the algorithm a single neighbour solution is generated for examination and comparison against the current best solution. If an improved neighbour is found it is adopted as the current solution and the search continues. If the neighbour is not an improvement on the current solution it is discarded and the search continues with other neighbours. The best solution is returned at the end of the search.

The tabu search mechanism implemented for the experiments is similar in nature to that described in (Blazewicz et al., 1993). The process generates a given number of neighbours, using the selected operator, and moves to the best solution in this subset of the neighbourhood. This best solution is then used to generate the next set of neighbours and the cycle continues. The previously generated shape sequences are held in the tabu list, this means that the algorithm will not revisit recently seen sequences within a given list length.

For the experiments in this chapter the tabu search uses a neighbourhood size of 5 solutions and a tabu list length of 200 solutions. These values were chosen from a set of possible values during initial experiments.

The operators used throughout both search techniques are 1 Opt, 2 Opt, 3 Opt, 4 Opt and *N* Opt. 1 Opt removes a randomly chosen shape and inserts it at a random location in the sequence. 2 Opt swaps two randomly chosen shapes in the order, although not two of the same type as this would produce the same result, this is extended up to 4 Opt where four randomly selected shapes are swapped. *N* Opt selects a random number of shapes to swap and is likely to produce a radically different solution, and thus diversify the search.

The solution operator is chosen by means of an independently generated random number selected, within bounds, that is then compared to a weighted scale, which gives the particular operator to be used. Each operator has the following chance of selection:

1 Opt = 40%, 2 Opt = 30%, 3 Opt = 15%, 4 Opt = 10%, *N* Opt = 5%

This is because the less radical operators allow for the concentration of the search and the highly radical operators, e.g. N Opt, allow the search to escape local optima.

No statistical sensitivity analysis was performed on these input parameters and operators as introduction of the meta-heuristic search was a major step forward for the industrial partner's nesting technology, it had previously only been able to attempt the shape area descending and shape length descending heuristics used as starting points in the following experiments. Consequently when some reasonably well performing settings were arrived at the focus of the CASE and TCS projects was directed towards generating additional useful geometric techniques, these additional techniques are the focus of the later chapters of this thesis.

The following pseudocode shows how both of the local searches interface with bottom-left-fill:

Algorithm 6. The Local Search Process using Hill Climbing and Tabu Search

```
INPUT:   Problem Shapes, Quantities and Allowable Rotations, Sheet Size
current.ordering = Sort Ordering(decreasing area, decreasing length)

Begin

current.evaluation = Bottom-Left-Fill(current.ordering);
best = current;

while (!Stopping Criteria)     // either max number of iterations or time based
{
        opt = Select Operator (1,2,3,4,n Opt);

        if (TABU)
        {
        for (i = 0; i < neighbourhood size; i++)
        {
                neighbour[i].ordering = Generate NotTabu Neighbour(current ordering, opt);
                neighbour[i].evaluation = Bottom-Left-Fill(neighbour[i].ordering);
        }
        current = GetBestNeighbour(neighbour[]);
        }
        else if (HILL CLIMBING)
        {
        neighbour.ordering = Generate Neighbour(current.ordering, opt);
        neighbour.evaluation = Bottom-Left-Fill(neighbour.ordering);
        if (neighbour.evaluation <= current.evaluation) { current = neighbour; }
        }

        if (current.evaluation < best.evaluation) { best = current; }

}

return best;

End
```

4.8. Benchmark Problems from the Literature

To test the effectiveness of the new Bottom-Left-Fill algorithm the experiments are run on the relevant test problems from the literature. Some of these problems were first collected by Hopper in (Hopper, 2000) and now feature on the EURO Special Interest Group on Cutting and Packing (ESICUP) website (http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=2).

Additionally (Hopper, 2000) introduced 10 new problems, 9 of which were randomly generated and consist of varying quantities of similar polygonal shapes (note: for these 9 "Poly" problems, it is necessary to rotate the shapes into their minimum bounding rectangle orientation before applying the problems' rotation criteria). In (Oliveira et al., 2000) the authors introduce 5 new problems, 3 of which are drawn directly from the textile industry. (Blazewicz et al., 1993), (Jakobs, 1996) and (Dighe & Jakiela, 1996) introduce two problems each to the collection. The remaining problems have each been contributed by different practitioners. Table 5 shows 21 problems and provides the best known results using a length based evaluation at the time that this algorithm was published. Table 6 contains 5 problems and the results for the best known solutions for the problems evaluated by density measures at the point this work was published. A full list of all subsequent benchmark improvements can be found in Table 3 of section 2.8.

Table 5. Length evaluated benchmark problems from the literature

Original Author	Problem Name	No. of Shapes	Rotational Constraints	Sheet Width	Best Length	Best Available Result Reference
Blazewicz, Hawryluk & Walkowiak (1993)	Blasz1	28	0, 180 Absolute	15	27.3	Gomes & Oliveira (2002) [called SHAPES2]
Ratanapan & Dagli (1997)	Dagli	30	90 Incremental	60	65.6	Hopper (2000) [using NestLib]
Fujita, Akagi & Kirokawa (1993)	Fu	12	90 Incremental	38	34	Fujita & Akagi (1993)
Jakobs (1996)	Jakobs1	25	90 Incremental	40	13.2	Hopper (2000) [using SigmaNest]
Jakobs (1996)	Jakobs2	25	90 Incremental	70	28.2	Hopper (2000) [S. Annealing]
Marques, Bispo & Sentieiro (1991)	Marques	24	90 Incremental	104	83.6	Hopper (2000) [Naïve Evolution]
Hopper (2000)	Poly1A	15	90 Incremental	40	14.7	Hopper (2000) [using NestLib]
Hopper (2000)	Poly2A	30	90 Incremental	40	30.1	Hopper (2000) [using NestLib]
Hopper (2000)	Poly3A	45	90 Incremental	40	40.4	Hopper (2000) [using NestLib]
Hopper (2000)	Poly4A	60	90 Incremental	40	56.9	Hopper (2000) [using NestLib]
Hopper (2000)	Poly5A	75	90 Incremental	40	71.6	Hopper (2000) [using NestLib]
Hopper (2000)	Poly2B	30	90 Incremental	40	33.1	Hopper (2000) [using SigmaNest]
Hopper (2000)	Poly3B	45	90 Incremental	40	41.8	Hopper (2000) [using NestLib]
Hopper (2000)	Poly4B	60	90 Incremental	40	52.9	Hopper (2000) [using NestLib]
Hopper (2000)	Poly5B	75	90 Incremental	40	63.4	Hopper (2000) [using NestLib]
Hopper (2000)	SHAPES	43	90 Incremental	40	63	Hopper (2000) [Simple Heuristic]
Oliveira & Ferreira (1993)	SHAPES0	43	0 Absolute	40	63	Dowsland & Dowsland (1993)
Oliveira & Ferreira (1993)	SHAPES1	43	0, 180 Absolute	40	59	Gomes & Oliveira (2002)
Oliveira & Ferreira (1993)	SHIRTS	99	0, 180 Absolute	40	63.13	Gomes & Oliveira (2002)
Oliveira, Gomes & Ferreira (2000)	SWIM	48	0, 180 Absolute	5752	6568	Hopper (2000) [using NestLib]
Oliveira, Gomes & Ferreira (2000)	TROUSERS	64	0, 180 Absolute	79	245.75	Gomes & Oliveira (2002)

Table 6. Density evaluated benchmark problems from the literature

Original Author	Problem Name	No. of Shapes	Rotational Constraints	Sheet Width	Best Density	Best Available Result Reference
Albano & Sappupo (1980)	Albano	24	90 Incremental	4900	86%	Hopper (2000) [S. Annealing]
Blazewicz, Hawryluk & Walkowiak (1993)	Blasz2	20	90 Incremental	15	68.6%	Blaszewicz, Hawryluk & Walkowiak(1993)
Dighe & Jakiela (1996)	Dighe1	16	90 Incremental	100	72.4%	Hopper (2000) [using NestLib]
Dighe & Jakiela (1996)	Dighe2	10	90 Incremental	100	74.6%	Hopper (2000) [Gen. Algorithm]
Bounsaythip & Maouche (1997)	Mao	20	90 Incremental	2550	71.6%	Hopper (2000) [Gen. Algorithm]

4.9. New Benchmark Problems

In addition to the literature problems the algorithm is tested on ten new benchmark problems for the irregular stock cutting problem. Table 7 gives the details of these new problems. The new benchmark data can be found in the appendices of this thesis

Table 7. New Benchmark Problems

Problem Name	No. of Shapes	Rotational Constraints	Sheet Width	Optimal Known
Profiles1	32	90 Incremental	2000	No
Profiles2	50	90 Incremental	2500	No
Profiles3	46	45 Incremental	2500	No
Profiles4	54	90 Incremental	500	No
Profiles5	50	15 Incremental	4000	No
Profiles6	69	90 Incremental	5000	No
Profiles7	9	90 Incremental	500	Yes
Profiles8	18	90 Incremental	1000	Yes
Profiles9	57	90 Incremental	1500	No
Profiles10	91	0 Absolute	3000	No

4.10. Line & Arc - Profiles1 to Profiles5

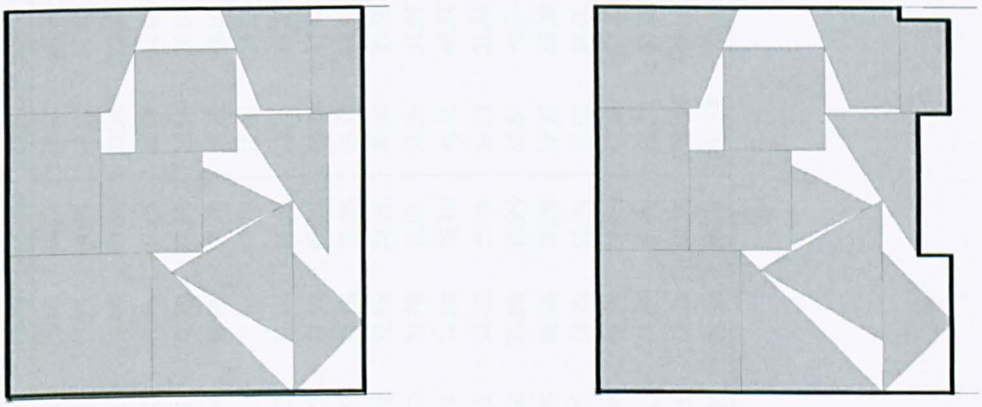
These new problems introduce arcs and holes to the set of benchmark problems. Some of these shapes, in particular Profiles1 and Profiles2, have been chosen from a library of standard shapes within the sheet metal profiling industry. All of these problems contain at least one shape consisting of one or more arcs and their optimal solutions are not known.

4.11. Line Only - Profiles6 to Profiles10

A further five problems have been created, some involving shapes with holes, which can be tackled by non-arc implementations. The optimal solutions are not known with the exception of Profiles7 and Profiles8 which are 'jigsaw' problems where the optimal length is 1000 units for both problems. Profiles9 is a novelty dataset involving a subset of letters from the English alphabet. Profiles10 combines polygons from numerous literature benchmark problems.

4.12. Experiments

For the experiments the proposed Bottom-Left-Fill placement algorithm (section 4.6) and local search techniques (section 4.7) to generate input shape orderings. During the search process, the generated solutions are evaluated by the total length of the packing. Additionally the evaluation function records two different density measures, the first is a simple straight line density (Density1) whilst the second density measure, used by Hopper (2000), is based on the union of all individual shape bounding rectangles; this allows us to use a non-rectangular final density measurement (Density2). Both methods are pictured in figure 10, where the thicker line represents the containing area.



Straight Line Density Measure (Density1)

Hopper (2000) Density Measure (Density2)

Figure 23. Density Measures

Density can then calculated by: Density = Total Shape Area / Containing Area

All of the experiments conducted were performed on a PC with a 2GHz Intel Pentium 4 processor and 256MB RAM.

4.13. Experiments on Literature Benchmark Problems

The results of the literature problem experiments are divided into two groups, those for which the best known result is measured using overall packing length and those which have been evaluated using density measures. Each problem was run 10 times using 100 iteration runs, for both length and area initial orderings (resulting in a total of 40 runs for each problem). The results of the experiments are shown in the following tables

Table 8. Experiments on length evaluated literature benchmark problems (100 iteration runs, best results shown in bold)

		Hill Climbing								Tabu Search							
		Length Ordered				Area Ordered				Length Ordered				Area Ordered			
Problem	Best	Average	Best Result			Average	Best Result			Average	Best Result			Average	Best Result		
	Literature	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2
Blasz1	27.30	28.72	28.60	75.5%	78.0%	28.57	28.40	76.1%	**81.0%**	28.42	28.00	77.1%	78.5%	28.64	**27.80**	**77.7%**	79.9%
Dagli	65.60	62.57	61.82	82.0%	84.1%	62.28	61.67	82.2%	**89.0%**	62.15	61.10	83.0%	84.2%	61.98	**60.57**	**83.7%**	84.3%
Fu	34.00	33.57	33.00	86.4%	90.1%	33.90	33.00	86.4%	90.3%	33.58	**32.80**	**86.9%**	90.0%	33.67	33.00	86.4%	**90.8%**
Jakobs1	13.20	12.40	12.00	81.7%	89.1%	12.02	12.00	81.7%	87.8%	12.08	12.00	81.7%	**92.6%**	11.95	**11.86**	**82.6%**	89.5%
Jakobs2	28.20	26.40	26.00	74.2%	**83.3%**	26.62	26.00	74.2%	77.8%	26.53	26.00	74.2%	80.8%	26.37	**25.80**	**74.8%**	83.1%
Marques	83.60	82.76	81.00	85.4%	88.6%	81.97	80.84	85.6%	88.2%	84.20	83.00	83.3%	86.0%	81.10	**80.00**	**86.5%**	**89.3%**
Poly1A	14.70	14.73	14.27	71.8%	76.0%	14.40	14.03	73.1%	77.5%	14.80	14.56	70.4%	76.0%	14.46	**14.00**	**73.2%**	**78.2%**
Poly2A	30.10	28.82	28.41	72.1%	**77.5%**	28.68	28.43	72.1%	75.3%	28.76	28.26	72.5%	76.3%	28.42	**28.17**	**72.8%**	75.9%
Poly3A	40.40	43.14	42.65	72.1%	74.3%	42.49	41.77	73.6%	**75.5%**	43.24	42.48	72.4%	74.0%	42.61	**41.65**	**73.8%**	74.8%
Poly4A	56.90	56.89	56.30	72.8%	74.6%	56.93	55.32	74.1%	75.8%	56.14	**54.93**	**74.6%**	**75.9%**	56.07	55.51	73.9%	75.4%
Poly5A	71.60	70.64	**69.37**	**73.9%**	**75.7%**	71.25	70.72	72.5%	74.0%	70.96	70.69	72.5%	73.4%	70.98	70.55	72.6%	73.8%
Poly2B	33.10	31.14	30.98	73.0%	77.1%	31.15	30.70	73.7%	77.0%	31.26	31.09	72.7%	76.2%	31.05	**30.00**	**75.4%**	**77.5%**
Poly3B	41.80	41.33	40.91	74.5%	**77.1%**	41.13	40.77	74.8%	76.4%	41.16	**40.74**	**74.9%**	76.5%	41.31	40.88	74.6%	76.4%
Poly4B	52.90	52.21	51.91	74.5%	76.5%	52.56	52.12	74.2%	75.5%	52.11	**51.73**	**74.8%**	**77.4%**	52.05	51.78	74.7%	76.5%
Poly5B	63.40	61.97	60.83	75.5%	**77.2%**	61.19	**60.54**	**75.8%**	76.9%	61.87	61.04	75.2%	76.6%	61.43	61.43	74.7%	75.3%
SHAPES	63.00	60.00	59.20	67.4%	68.4%	60.22	60.00	66.5%	**69.1%**	59.72	**59.00**	**67.6%**	69.0%	60.00	60.00	66.5%	**69.1%**
SHAPES0	63.00	66.50	**66.00**	**60.5%**	**62.6%**	67.26	66.70	59.8%	62.0%	66.60	**66.00**	**60.5%**	62.3%	67.22	67.00	59.6%	61.6%
SHAPES1	59.00	63.38	62.10	64.3%	65.5%	61.32	**60.00**	**66.5%**	**68.9%**	62.36	61.00	65.4%	68.1%	62.18	62.00	64.4%	66.9%
SHIRTS	63.13	64.18	64.00	84.4%	85.7%	65.06	64.30	84.0%	86.4%	64.24	**63.80**	**84.6%**	86.0%	64.99	64.04	84.3%	**87.3%**
SWIM	6568.00	6610.98	**6462.40**	**68.4%**	**71.6%**	6713.08	6601.80	67.0%	69.2%	6551.88	6489.80	68.2%	69.6%	6804.80	6723.70	65.8%	70.2%
TROUSERS	245.75	251.39	248.67	87.7%	89.0%	248.88	246.98	88.3%	**90.1%**	251.54	248.38	87.8%	88.6%	249.56	**246.57**	**88.5%**	89.7%

Table 9. Experiments on density evaluated literature benchmark problems (100 iteration runs, best results shown in bold)

		Hill Climbing								Tabu Search							
		Length Ordered				Area Ordered				Length Ordered				Area Ordered			
Problem	Best	Average	Best Result			Average	Best Result			Average	Best Result			Average	Best Result		
	Literature	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2
Albano	86.0%(D2)	10465.16	10354.80	84.1%	85.9%	10505.64	**10292.90**	**84.6%**	86.2%	10426.76	10315.10	84.4%	**86.5%**	10406.74	10298.60	84.5%	85.8%
Blasz2	68.6%(D1)	25.85	25.60	73.6%	79.3%	25.87	**25.28**	**74.5%**	78.4%	25.86	25.38	74.2%	**79.9%**	25.70	25.38	74.2%	78.2%
Dighe1	72.4%(D2)	1351.10	1320.00	75.8%	76.9%	1342.74	1324.60	75.5%	77.8%	1327.72	1307.20	76.5%	78.2%	1332.98	**1292.30**	**77.4%**	**78.9%**
Dighe2	74.6%(D2)	1302.20	**1260.00**	**79.4%**	83.3%	1303.36	1270.70	78.7%	81.5%	1279.38	1270.70	78.7%	81.5%	1295.14	**1260.00**	**79.4%**	**84.3%**
Mao	71.6%(D2)	1875.52	**1854.30**	**79.5%**	82.0%	1889.32	1863.40	79.1%	82.5%	1875.46	**1854.30**	**79.5%**	82.0%	1890.28	1867.00	78.9%	**82.9%**

A summary of the best results from these 40 runs is presented in Table 10 detailing the average times per nest, time taken to find the best solution and the percentage improvement on the best known result from the literature. For problems where the best solution found was an improvement upon the best known solution at the point the research was carried out, 2002, the percentage improvement is marked in a bold typeface.

Table 10. Summary of best results using 100 iteration runs

Problem	Best	Best Result			Time/Nest	Time to best	Percentage
	Literature	Length	Density1	Density2	(s)	(s)	Improvement
Blasz1	27.3	27.80	77.7%	81.0%	0.32	21	-1.83%
Dagli	65.6	60.57	83.7%	89.0%	2.04	188.8	**7.66%**
Fu	34	32.80	86.9%	90.8%	0.24	20.78	**3.53%**
Jakobs1	13.2	11.86	82.6%	92.6%	0.74	43.49	**10.17%**
Jakobs2	28.2	25.80	74.8%	83.3%	2.13	81.41	**8.51%**
Marques	83.6	80.00	86.5%	89.3%	0.25	4.87	**4.31%**
Poly1A	14.7	14.00	73.2%	78.2%	0.36	12.48	**4.76%**
Poly2A	30.1	28.17	72.8%	77.5%	1.24	120.56	**6.42%**
Poly3A	40.4	41.65	73.8%	75.5%	2.01	210.07	-3.10%
Poly4A	56.9	54.93	74.6%	75.9%	2.43	203.17	**3.47%**
Poly5A	71.6	69.37	73.9%	75.7%	5.04	475.63	**3.12%**
Poly2B	33.1	30.00	75.4%	77.5%	2.50	179.86	**9.36%**
Poly3B	41.8	40.74	74.9%	77.1%	4.26	417.67	**2.54%**
Poly4B	52.9	51.73	74.8%	77.4%	8.24	95.66	**2.20%**
Poly5B	63.4	60.54	75.8%	77.2%	14.70	676.61	**4.51%**
SHAPES	63	59.00	67.6%	69.1%	0.60	31.36	**6.35%**
SHAPES0	63	66.00	60.5%	62.6%	0.93	21.33	-4.76%
SHAPES1	59	60.00	66.5%	68.9%	0.82	2.19	-1.69%
SHIRTS	63.13	63.80	84.6%	87.3%	4.99	58.36	-1.06%
SWIM	6568	6462.40	68.4%	71.6%	12.39	607.37	**1.61%**
TROUSERS	245.75	246.57	88.5%	90.1%	7.89	756.15	-0.33%
Albano	86.0%	10292.90	84.6%	86.5%	1.18	93.39	**0.5%**
Blasz2	68.6%	25.28	74.5%	79.9%	0.16	10.94	**11.3%**
Dighe1	72.4%	1292.30	77.4%	78.9%	0.22	8.87	**6.5%**
Dighe2	74.6%	1260.00	79.4%	84.3%	0.10	7.12	**9.7%**
Mao	71.6%	1854.30	79.5%	82.9%	0.38	29.74	**11.3%**

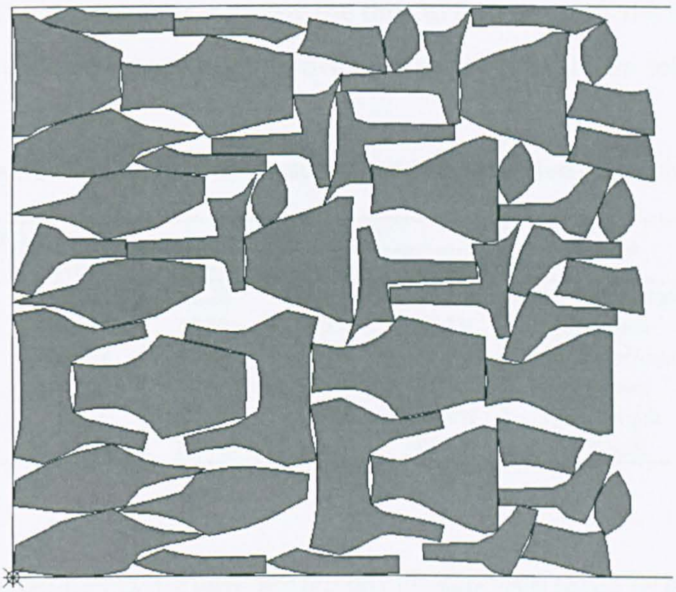


Figure 24. "SWIM" 100 iteration solution (6462.4 units, 607 seconds)

The initial round of experiments shows that the proposed approach has improved upon 20 of the 26 available literature problems. The best solution for dataset SWIM was obtained in 607 seconds and is shown in Figure 24. The average overall improvement over the 26 problems is slightly over 4% and is nearer 6% for the 20 problems where the new best solution has been found. On average the best solutions have been obtained within a few minutes execution time, although for many of the problems only a few seconds are necessary due to the high performance of the presented algorithm. The average time per nest compares favourably with other literature approaches utilising the no-fit polygon. For example Gomes and Oliveira (2002) state that a solution for the problem SHAPES1 can be generated within 6.18 seconds, on a 450MHz CPU, in comparison with 0.82 seconds for a solution generated by this new nesting method.

The experiments where then extended on the problems for which new benchmarks had not been set, Blasz1, SHAPES1, SHIRTS and TROUSERS. For these extended runs the durations were extended to 551.73s, 2019.77s, 6367.57s and 13613.67s respectively. These upper run limits are line with those used by the authors who presented the then relevant benchmarks for these problems (Gomes & Oliveira, 2002). For the problems SHAPES and Poly3A the experiment was simply extended to 1000 iterations per run, as guidance on search times is not available in the literature. These extension experiments for performed for an additional 10 runs. Table 11 shows the best results of these runs including the time to best solution, the method that generated that solution and percentage improvement on the best known solution.

Table 11. Summary of the results from the extended experiments

Problem	Best Literature (2002)	Average Length	Best Result			Method	Time to best (s)	Percentage Improvement
			Length	Density1	Density2			
Blasz1	27.3	27.47	27.20	79.4%	80.7%	Hill Climb/Length	501.91	**0.37%**
Poly3A	40.4	41.45	40.33	76.2%	77.3%	Tabu/Area	1515.49	**0.18%**
SHAPES0	63	65.68	65.00	61.4%	63.6%	Hill Climb/Area	332.39	-3.17%
SHAPES1	59	60.53	58.40	68.3%	71.5%	Tabu/Area	1810.14	**1.02%**
SHIRTS	63.13	63.66	63.00	85.7%	88.1%	Tabu/Length	806.5	**0.21%**
TROUSERS	245.75	246.40	243.40	89.6%	91.1%	Tabu/Area	3611.99	**0.96%**

Of the 6 extended experiments the new bottom-left fill algorithm sets 5 new benchmarks for Blasz1, Poly3A, SHAPES1, SHIRTS and TROUSERS. For the remaining problem, SHAPES0, the solution matches the work of (Gomes & Oliveira, 2002) (length of 65 units) but has not reached the best known solution of 63 units of (Dowsland & Dowsland, 1993). Using the new nesting algorithm on the 26 problems from the literature has produced 25 new best solutions. This was the first time any research paper in this area had attempted such a broad range of problems with such success. The solution obtained for TROUSERS is shown in Figure 25. All best solutions obtained for the literature problems are shown in the appendices.

Figure 25. "TROUSERS" extended run solution (243.4 units, 3612 seconds)

4.14. Experiments on New Benchmark Problems

The next set of experiments conducted involved setting benchmarks for the new problems, Profiles1 –10. Each problem was run 10 times for 30 minute durations with both hill climbing and tabu search (length and area initial orderings) resulting in a total of 40 runs. Table 13 is summary of the best results from these 40 runs. Figure 26 shows the best solutions for datasets Profiles1 and Profiles9. All best solutions obtained for these new problems are available in the appendices of the thesis.

Table 12. Experiments on new benchmark problems (30 minutes per run, best results shown in bold)

Problem	Hill Climbing								Tabu Search							
	Length Ordered				Area Ordered				Length Ordered				Area Ordered			
	Average	Best Result			Average	Best Result			Average	Best Result			Average	Best Result		
	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2
Profiles1	1417.50	**1377.74**	**82.0%**	**85.2%**	1443.21	1436.20	78.4%	80.4%	1431.45	1405.64	80.3%	84.1%	1453.12	1427.64	79.1%	82.3%
Profiles2	3285.67	3261.00	49.3%	50.5%	3291.01	3225.54	49.8%	**51.3%**	3291.86	**3216.10**	**50.0%**	50.8%	3344.81	3280.00	49.0%	50.0%
Profiles3	8311.23	8228.55	50.7%	51.1%	8252.02	**8193.89**	**50.9%**	**52.6%**	8323.90	8231.14	50.7%	51.5%	8392.79	8290.49	50.3%	51.7%
Profiles4	2489.54	2480.96	74.3%	75.1%	2492.88	2479.30	74.3%	74.8%	2488.86	2460.60	74.9%	**75.7%**	2486.56	**2453.12**	**75.1%**	75.7%
Profiles5	3427.47	3360.57	69.6%	73.1%	3381.38	**3332.70**	**70.2%**	71.1%	3405.22	3352.90	69.7%	**73.6%**	3383.31	3360.94	69.6%	71.3%
Profiles6	3180.16	3170.25	73.9%	76.7%	3177.91	3146.82	74.4%	77.5%	3147.11	**3097.86**	**75.6%**	**77.8%**	3162.74	3145.76	74.5%	77.0%
Profiles7	1326.71	**1296.30**	**77.1%**	78.3%	1326.57	1309.10	76.4%	**80.2%**	1324.38	1309.10	76.4%	79.6%	1339.99	1325.40	75.4%	78.3%
Profiles8	1330.88	**1318.70**	**75.8%**	77.2%	1361.02	1341.92	74.5%	77.7%	1337.57	1322.55	75.6%	**78.0%**	1355.04	1338.47	74.7%	76.4%
Profiles9	1341.70	1332.98	51.4%	53.9%	1323.44	1314.97	52.1%	54.3%	1350.47	1318.68	51.9%	54.2%	1305.70	**1290.67**	**53.1%**	**54.9%**
Profiles10	11496.12	11410.60	64.7%	65.5%	11758.50	11555.60	63.9%	64.6%	11401.28	**11160.10**	**66.2%**	**66.8%**	11657.30	11552.30	63.9%	64.7%

Table 13. Summary of best results for new benchmark problems

Problem	Best Result		
	Length	Density1	Density2
Profiles1	1377.74	82.0%	85.2%
Profiles2	3216.10	50.0%	51.3%
Profiles3	8193.89	50.9%	52.6%
Profiles4	2453.12	75.1%	75.7%
Profiles5	3332.70	70.2%	73.6%
Profiles6	3097.86	75.6%	77.8%
Profiles7	1296.30	77.1%	80.2%
Profiles8	1318.70	75.8%	77.2%
Profiles9	1290.67	53.1%	54.9%
Profiles10	11160.10	66.2%	66.8%

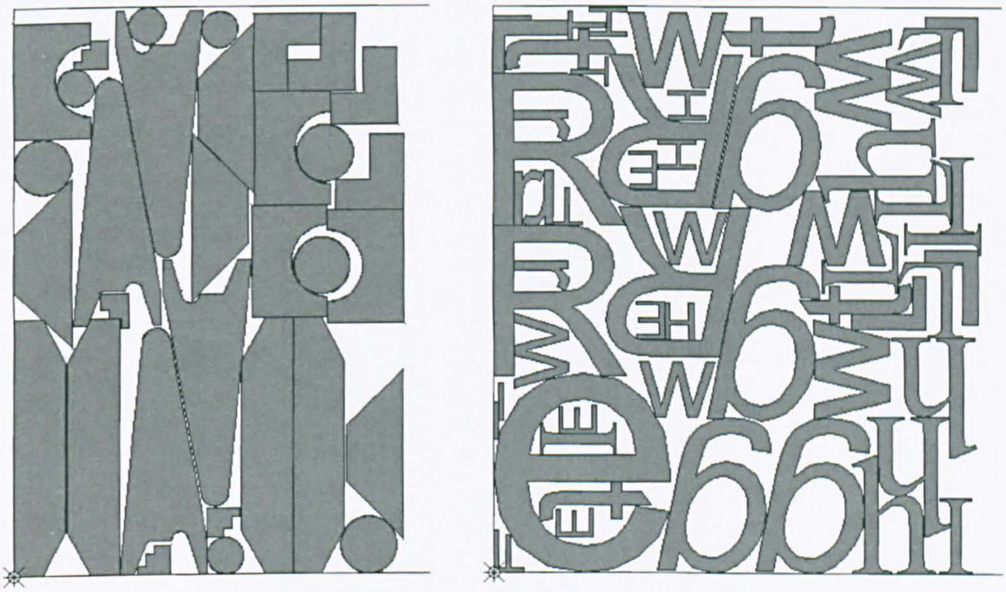


Figure 26. Best solutions for "Profiles1" (1377.74 units) & "Profiles9" (1290.67 units)

4.15. Summary

This chapter introduced a new technique of primitive overlap resolution. It then outlined the steps required for implementing an efficient bottom left fill nesting algorithm that is able to handle profiles with both circular arcs and holes. This algorithm produces nests for realistic problems both quickly and to a level of accuracy that makes it a strong candidate for industrial application. In the course of applying the new algorithm to 26 literature problems the proposed algorithm, using only simple local search mechanisms, was able to produce the best-known results for 25 of the 26 problems. The majority of these best solutions were generated within 100 search iterations and yielded an average improvement of 5% over the existing best solutions from the literature.

Upon publication this work was the first in the field of two dimensional irregular cutting and packing that had attempted such a broad range of problems from the literature. Furthermore the work introduces and sets initial benchmarks for 10 new problems, some of which involve profiles with both arcs and holes that have not previously been represented within the literature.

CHAPTER FIVE

5. No-Fit Polygon

5.1. Introduction

The no-fit polygon is a construct that can be used between pairs of shapes to allow for fast and efficient handling of geometry within irregular two-dimensional stock cutting problems. Previously, the no-fit polygon (NFP) has not been widely applied because of the perception that it is difficult to implement and the lack of generic approaches that can cope with all problem cases without specific case-by-case handling. Indeed in (Bennell & Oliveira, 2008), which cites the algorithm described in this chapter via (Burke et al., 2007), the authors note that the perceived difficulty in developing a robust no-fit polygon generation technique is regarded as a barrier to the study of irregular packing problems.

The focus of this chapter is to introduce an improved orbital method for the generation of no-fit polygons that is both *complete* and *robust* and therefore does not suffer from the typical problem cases found in other approaches from the literature. Furthermore, the algorithm only involves two simple geometric stages that work for all shape instances without special handling and therefore it is easily understood and implemented. We show several examples to demonstrate how the proposed approach can handle the known degenerate cases such as holes, interlocking concavities and jigsaw type pieces. At the point of publication this was the first no-fit polygon implementation that could successfully cope with such features. Chapter five of this thesis extended the work presented here to include arcs and shows via experimental results that the no-fit polygon represents a considerably more efficient solution for overlap detections than standard trigonometric techniques.

Whilst the generation of the no-fit polygon is challenging, it is a 'tool' and not a 'solution' and this is perhaps one of the reasons that there are many publications in the literature which state that the no-fit polygon is used but which provide relatively little or no details on its implementation. The section 3.4 concentrates specifically on the no-fit polygon and provides an overview of the previous techniques that have been used for its creation.

This chapter outlines the full implementation details for a new robust orbital approach that can cope with the traditional problem cases that many other approaches cannot handle.

5.2. The New No-Fit Polygon Construction Algorithm

The new approach can be divided into two logical stages. Section 5.3 describes the procedure for the first of these where one polygon slides around another to create the outer path of the no-fit polygon of the two shapes. This part follows an approach that is logically similar to that of Mahadevan's algorithm although a modified implementation is proposed. The second section introduces the notion and identification of starting positions that allow the algorithm to find the remaining paths of the no-fit polygon (see section 5.4). These paths will be internal holes of the no-fit polygon (i.e. contained by the outer path) and are not found using Mahadevan's algorithm alone. For this section the assumption is made that the algorithm is operating upon two anticlockwise oriented polygons, A and B, that exist somewhere within two-dimensional space in order to produce the no-fit polygon NFP_{AB} (orbiting polygon B around polygon A). The first operation that must be performed is to translate polygon B so that it is touching, but not intersecting, polygon A. The new approach maintains the same approach as used by Mahadevan whereby polygon B is translated so that its largest y-coordinate is placed at the lowest y-coordinate of the polygon, A (see Figure 27).

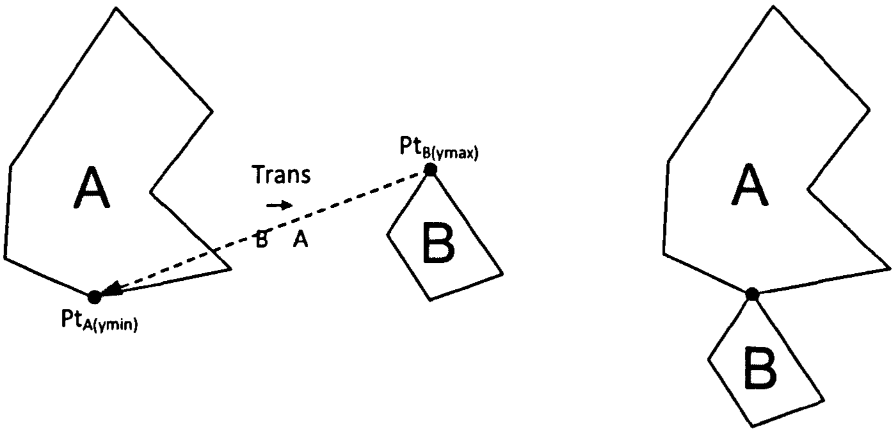


Figure 27. Initial translation of the orbiting polygon to touch the stationary polygon

Using these two vertices for alignment guarantees that A and B will be non-intersecting and touching. The translation that results in polygon B touching polygon A is shown in the formula **[1]**:

$$\text{Trans } B \rightarrow A = Pt_{A(ymin)} - Pt_{B(ymax)} \qquad \textbf{[1]}$$

In reality, any starting position may be used providing that it results in polygons A and B touching and not intersecting. The orbital approach can now commence to generate the outer path of the no-fit polygon by orbiting polygon B in an anticlockwise direction.

5.3. Orbiting / Sliding

The main aim of the orbiting (or sliding) section of the algorithm is to detect the correct movements that B must make to traverse around A in order to return to its original position. This is an iterative procedure with each translation step creating an edge of the no-fit polygon. This can be further broken down into the following subparts which are discussed in the following sections:

- Section 5.3.1- Detection of Touching Edges
- Section 5.3.2- Creation of Potential Translation Vectors
- Section 5.3.3- Finding a Feasible Translation
- Section 5.3.4- Trimming the Feasible Translation
- Section 5.3.5- Applying the Feasible Translation

5.3.1. Detection of Touching Edges

The ability to correctly detect touching and intersecting edges is of upmost importance in the course of the algorithm. This is achieved by testing each edge of polygon A against each edge of polygon B. Each pair of edges that touch (one from polygon A and one from polygon B) is stored along with the position of the touching vertex. Figure 28 shows the resulting set of touching edge pairs. This is in contrast to the approach of Mahadevan who performs calculations using all edges at a touching vertex. For example on Figure 28, Mahadevan would present edges a2, a3, b1 and b4 on the same diagram. Experience gathered whilst developing this algorithm has shown that this makes the required calculations longwinded and more difficult to explain. These edge-pair diagrams will be used in the next stages to create the potential translation vectors for polygon B and to eliminate the non-viable translation vectors.

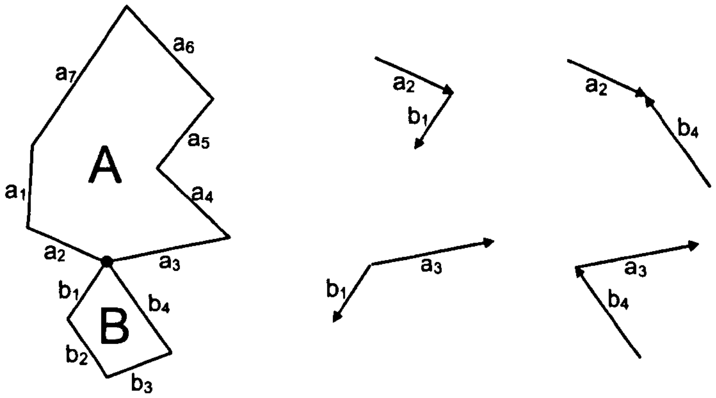


Figure 28. Identification of touching edge pairs

5.3.2. Creation of Potential Translation Vectors

The vector with which polygon B must be translated to orbit polygon A must either be derived from an edge of polygon A or from polygon B depending on the situation. Figure 29 shows an example of each case.

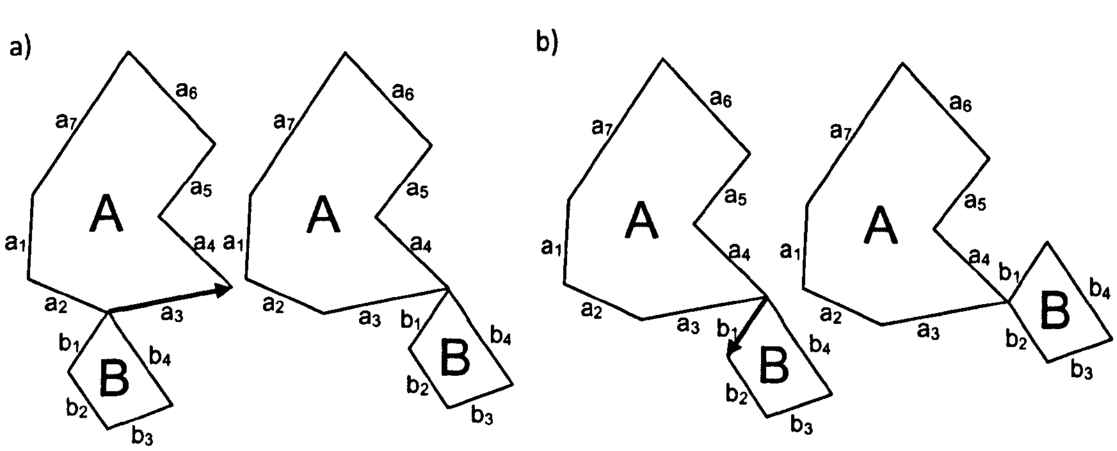


Figure 29. Translation vector: a) derived from edge a_3, b) derived from edge b_1

Note that in the second case (Figure 29b), because polygon B slides along one of its own edges, the translation vector is found by reversing the edge. This is further examined through the relative movement of polygon A with respect to B. Polygon A is relatively moved along the edge b_1. In reality, polygon A must remain fixed and B must be translated and therefore, the translation vector is reversed in this situation.

It is possible to obtain the set of potential translation vectors by using the touching edge pairs. There are three possibilities: i) both edges touch at a vertex, ii) a vertex of orbiting edge touches the middle of the stationary edge, or iii) a vertex of the stationary edge touches the middle of the orbiting edge. These cases are depicted in Figure 30.

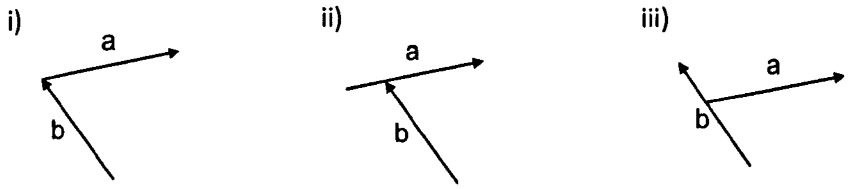


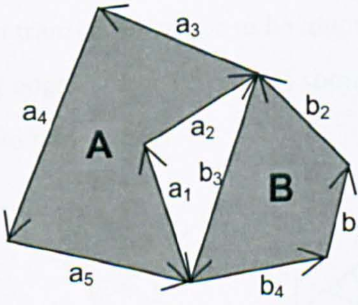
Figure 30. Touching edge-pair types

Each pair of touching edges yields one potential translation vector. In case (ii), the translation vector is simply defined using the point at which the two edges touch and the stationary edge's end vertex. In case (iii), a similar process is required except that the end vertex of the orbiting edge is used and the vector direction is also reversed. Case (i) requires the correct identification of whether the potential translation vector is derived from the stationary or orbiting edge. This can be identified by the following set of rules based on the touching vertices and a test for whether the orbiting edge, b, is left or right of the stationary edge, a. Table 14 shows the different possibilities and the edge from which a potential translation vector is derived under each circumstance.

Table 14. Deriving the potential translation when both edges touch at vertices

Case	Touching Edge Vertex		Relative Position of the Orbiting Edge to the Stationary Edge	Translation Derived From
	Stationary	Orbiting		
1	Start	Start	Left	Orbiting Edge
2	Start	Start	Right	Stationary Edge
3	Start	End	Left	-
4	Start	End	Right	Stationary Edge
5	End	Start	Left	-
6	End	Start	Right	Orbiting Edge
7	End	End		-
8			Parallel	Either Edge

Some potential translations can be eliminated in certain cases of the table (an '-' entry in the table indicates that no translation is derived). This is possible as when a translation is derived from an edge, the resultant vector is defined by touching point → end vertex (and then the vector is reversed for an orbiting edge). This allows the elimination of case 7 of the table because both the stationary and orbiting edges touch on the end vertices and this would yield a null translation vector. In cases 3 and 4, the orbiting edge touches at its end vertex thus a translation cannot be derived from the orbiting edge. In cases 5 and 6, the stationary edge cannot be used as it would produce a null vector. Figure 31 shows each of the cases using two polygons that touch at two separate places.



Case	Edges	Derived From
1	(a_3,b_3)	b_3
2	(a_1,b_4)	a_1
3	(a_3,b_2)	-
4	(a_1,b_3)	a_1
5	(a_5,b_4)	-
6	(a_2,b_3)	b_3
7	(a_2,b_2) , (a_5,b_3)	-

Figure 31. Two polygons touching at two separate positions and vertex-vertex touch cases

In cases 3 and 5, no translation can be derived because the edge of the orbiting polygon is to the left of the edge from the stationary polygon. For example, edges a_3 and b_2 touch on their start and end vertices respectively. As a null translation is produced using edge b_2 (due to touching with the end vertex), the only possibility is to derive a translation vector from the stationary edge, a_3. However, edge b_2 is left of edge a_3 and, if this translation vector were used, edge b_2 would slide along the inside of edge a_3. The positional "left test" eliminates the translations where polygons would slide along the inside of an edge rather than the outside of the edge. When edges are parallel, either the stationary or the orbiting edge may be used.

5.3.3. Finding a Feasible Translation Vector

Once the potential translation vectors have been produced, the next stage is to select a translation vector that does not result in an immediate intersection. For example, if we return to Figure 31 we have generated two potential translation vectors, a_1 and $-b_3$. It can easily be seen that the orbiting polygon B must be translated using vector $-b_3$ in order to move anticlockwise around polygon A. If we were to translate polygon B along vector a_1 instead, this would result in an immediate intersection between edges a_2 and b_3 (and also a_3 and b_3). Once again the set of touching edge-pairs, generated from section 5.3.1, contains all of the information needed to determine a feasible translation vector.

The process here is simplified due to our proposed representation of touching edges and involves taking each of the potential translation vectors in turn, identified in section 5.3.2, and placing them on the touching position at each touching edge-pair. We can define positional relationships between the stationary and orbiting edge based on the union of left/right regions that indicate whether a particular potential translation will be suitable for those edges. For a potential translation vector to be identified as feasible, it must be suitable for every pair of touching edges. Figure 32 shows some examples of touching edge-pairs and the angular range of feasible translations.

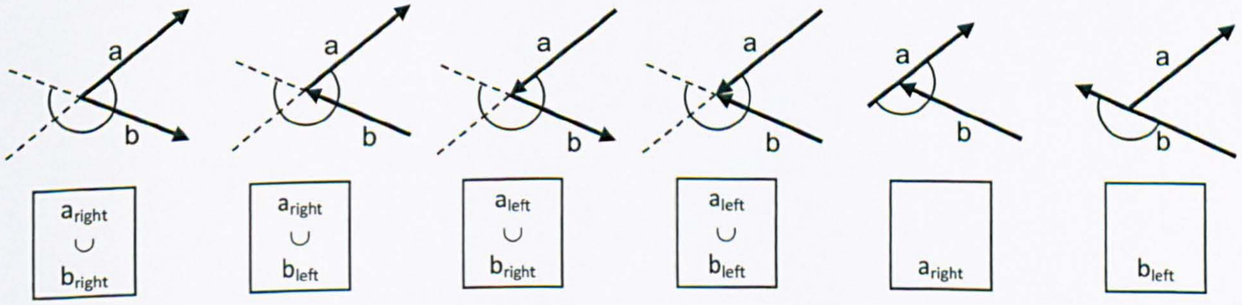

a
b
$a_{right} \cup b_{right}$
a
b
$a_{right} \cup b_{left}$
a
b
$a_{left} \cup b_{right}$
a
b
$a_{left} \cup b_{left}$
a
b
a_{right}
a
b
b_{left}

Figure 32. Identifying the feasible angular range of translations (indicated by an arc)

Whilst more possibilities can obviously occur the examples shown in Figure 32 should provide enough information to derive the omitted relationships. Figure 31 shows the calculation of two potential translation vectors, $-b_3$ and a_1, for two touching polygons. Figure 33 uses the same example to demonstrate how the approach eliminates translation vector a_1. In reality, once a potential translation fails on one of these tests, it is infeasible and can be eliminated without further testing.

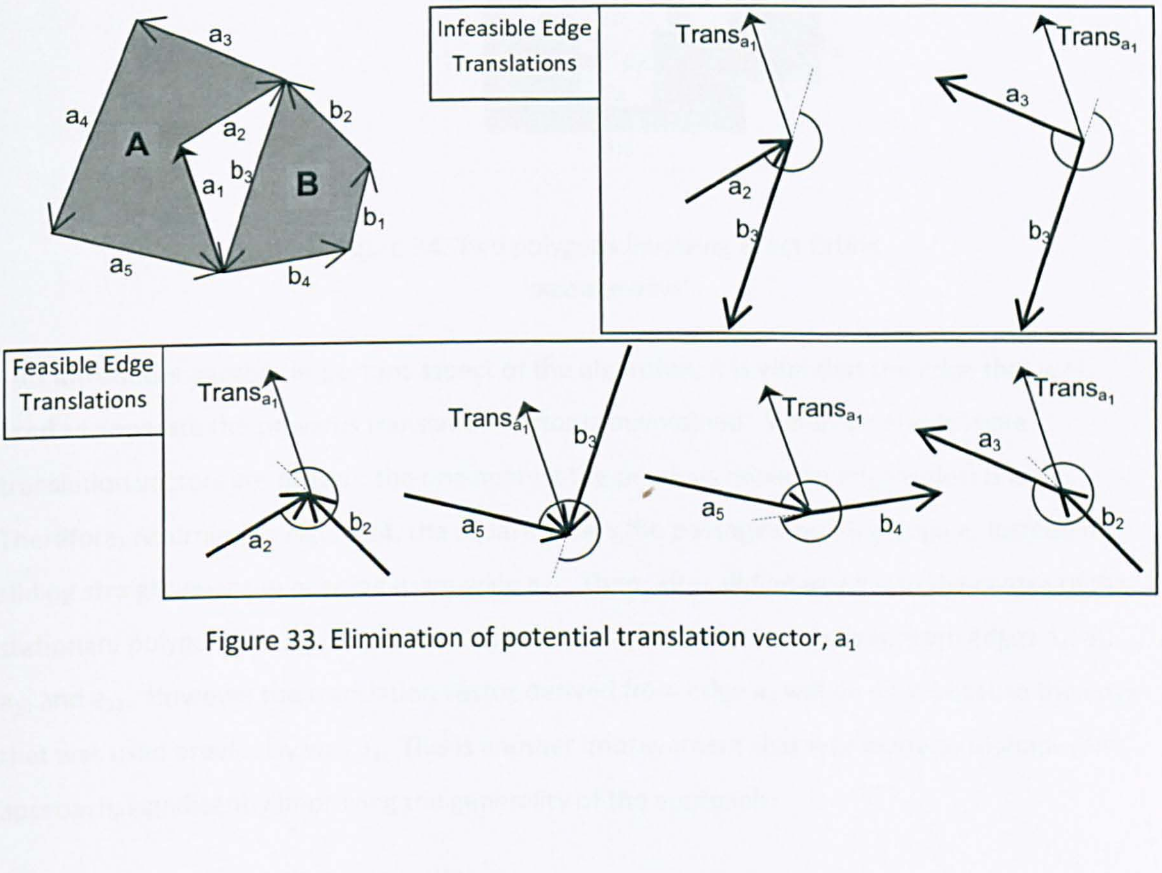


Figure 33. Elimination of potential translation vector, a_1

At this stage one or more feasible translation vectors and been found. Usually there will only be one translation vector but several may be present under the special circumstances that are illustrated in Figure 34 which is an actual screenshot taken from the implementation. The centre of the orbiting square has been set as the reference point for clarity.

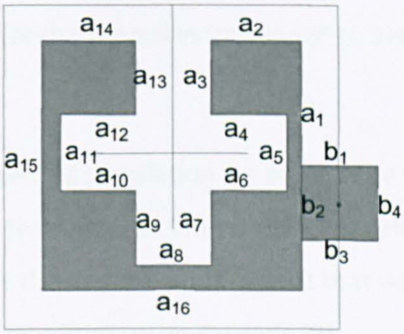


Figure 34. Two polygons involving exact fitting 'passageways'

This introduces another important aspect of the algorithm; it is vital that the edge that was used to generate the previous translation vector is maintained. When several feasible translation vectors are present the one nearest the previous move (in edge order) is used. Therefore, returning to Figure 34, the square enters the passageway using edge a_3 instead of sliding straight over the opening using edge a_{14}. Then, after sliding along into the centre of the stationary polygon, the square has four feasible translation vectors derived from edges a_4, a_7, a_{10} and a_{13}. However the translation vector derived from edge a_4 will be used because the edge that was used previously was a_3. This is another improvement that was made to Mahadevan's approach, significantly improving the generality of the approach.

5.3.4. Trimming the Feasible Translation Vector

The last step before the translation of the polygon B is to trim the translation vector. This is important because there may be other edges that can interfere with the translation of the orbiting polygon. Figure 35a provides an example where the application of the entire translation vector results in the two shapes intersecting. In order to prevent the orbiting polygon entering the body of the stationary polygon, the feasible translation, a_7, must be trimmed to edge a_1 (see Figure 35b).

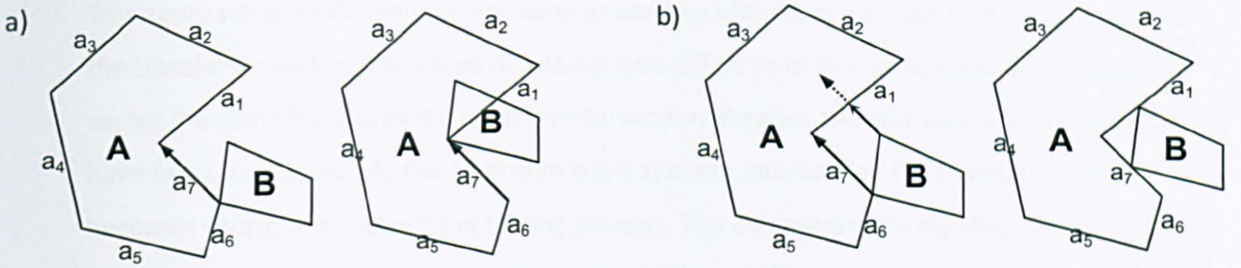


Figure 35. A translation vector than requires 'trimming' to avoid intersection

In order to find the correct non-intersecting translation we project the translation vector at each of the vertices of polygon B and test them for intersection with all edges of polygon A. This ensures that we correctly identify the vertices of polygon B that will cross into polygon A and reduce the translation distance accordingly using formula **[2]**.

$$\text{New Translation} = \text{IntersectionPt} - \text{Translation}_{\text{StartPt}} \qquad \textbf{[2]}$$

To avoid any possible intersection it is necessary to project the translation vector back from all vertices of polygon A (by translating the end vertex of the translation vector onto each vertex) and perform intersection testing with all edges of polygon B to test if any will cross into polygon B. This is depicted in Figure 36 using the same example but a different orbiting polygon.

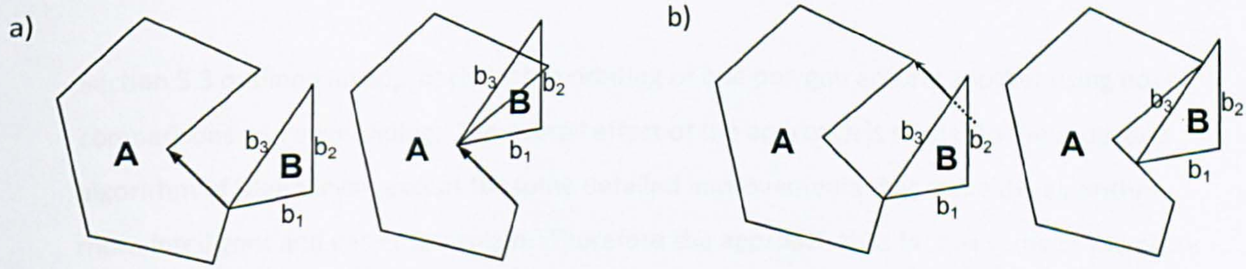


Figure 36. Trimming with projections from polygon A

Once again, if there is an intersection, the translation distance is reduced to reflect this using formula **[3]**.

$$\text{New Translation} = \text{Translation}_{\text{EndPt}} - \text{IntersectionPt} \qquad \textbf{[3]}$$

This approach is fundamentally the same as used by Mahadevan except that here the trim of the translation vector at the time of intersection differs as Mahadevan keeps the translation vector the same but stores the minimal intersection distance and trims once all projections have been completed. As the algorithm trims at every intersection the translation vector becomes shorter throughout the testing process. This can reduce the number of intersections that occur due to fast elimination tests through the use of bounding boxes, which are much faster than standard line intersections. The approach of Mahadevan may need to calculate several more full intersection tests because the entire translation vector is used at each projection and could potentially require more computation.

5.3.5. Applying the Feasible Translation Vector

The final step is to append the trimmed translation vector to the end of the partial no-fit polygon created so far and polygon B is translated by the trimmed translation vector and the traversed or partially traversed edge is flagged to avoid later start point processing, see next section. This process has moved the polygon to the next 'decision' point and the process can restart from the detection of touching edges (section 5.3.1). The only additional check to this process that is required is a test to detect if the reference point of polygon B has returned to its initial starting position.

5.4. Start Points

Section 5.3 outlined an approach for the orbiting of one polygon around another using edge comparisons and edge sliding. The overall effect of the approach is similar to the proposed algorithm of Mahadevan except for some detailed improvements that make the algorithm more intelligent and easier to explain. Therefore the approach thus far has some of the same limitations that were present in (Mahadevan, 1984) such as the inability to generate complete no-fit polygons for shapes involving interlocking concavities, jigsaw pieces or holes (see section 5.6). Figure 37a shows two polygons for which the sliding algorithm alone does not produce the complete no-fit polygon. The polygons have concavities that can interlock with each other to create a further no-fit polygon region but this is never found because of the narrow entrance to the stationary polygon's concavity (see Figure 37b). Another approach is needed to identify such possibilities.

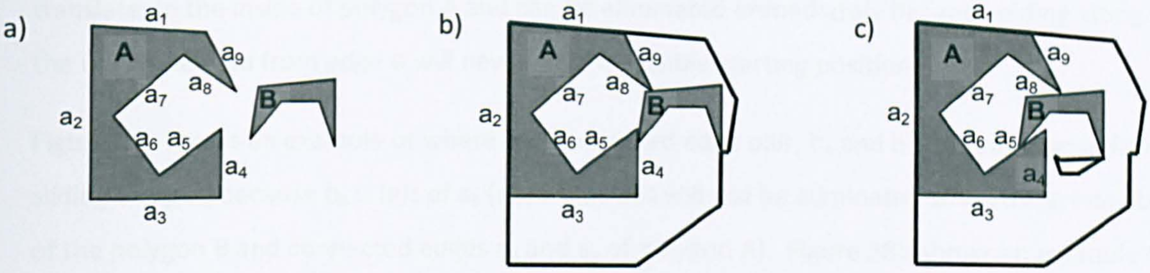


Figure 37. Interlocking concavities: a) polygons, b) no-fit polygon using sliding alone, c) the complete no-fit polygon

In the following section an approach based on the identification of feasible touching but non-intersecting start positions is described. Using the previously described sliding technique it is possible to generate the remaining inner loops of the no-fit polygon construct. For example, if polygon B can be placed in the location shown in Figure 37c, then the sliding algorithm can be employed (without change) to generate the internal no-fit polygon region (note that this loop will be a clockwise loop indicating that it is an inner-fit polygon region). Feasible starting positions are found through a modification to the approach described in section 5.3 and once again the process involves sliding.

In order to explain the process it is useful to examine an example of finding the start positions along one edge of polygon A. This process can then easily be reproduced for each edge of polygon A and polygon B to create all feasible starting positions to allow polygon B to orbit around polygon A in order to create the entire no-fit polygon.

Given an edge, e, of polygon A, we can detect the potential starting positions of an orbiting polygon B along e. The process involves translating polygon B such that each of its vertices, in turn, are aligned to the start vertex of e. For each position the following steps are performed. If the polygons do not intersect in this position, then this is noted as a feasible start position for the two polygons. If polygon B intersects with polygon A then a further test is required and the edge sliding techniques is employed to traverse along edge e until a non-intersecting position is found or the end of the edge is reached. In such an instance, the first test involves examining whether the two connected edges of polygon B (joined at the touching vertex) are both right of or parallel to, but not both parallel to, the edge e. This test is performed by checking the edges of polygon B against the edge e, the left, right parallel decision being made with respect to direction of edge e. If either of polygon B's connected edges are left of e then they will translate on the inside of polygon A and can be eliminated immediately because sliding along the vector derived from edge e will never yield a feasible starting position.

Figure 38a shows an example of where the connected edge pair, b_3 and b_4, are eliminated from sliding along a_5 because b_4 is left of a_5 (note that this will not be eliminated when using edge b_4 of the polygon B and connected edges a_4 and a_5 of polygon A). Figure 38b shows an example of where both connected edges, b_1 and b_2, are right of edge a_5.

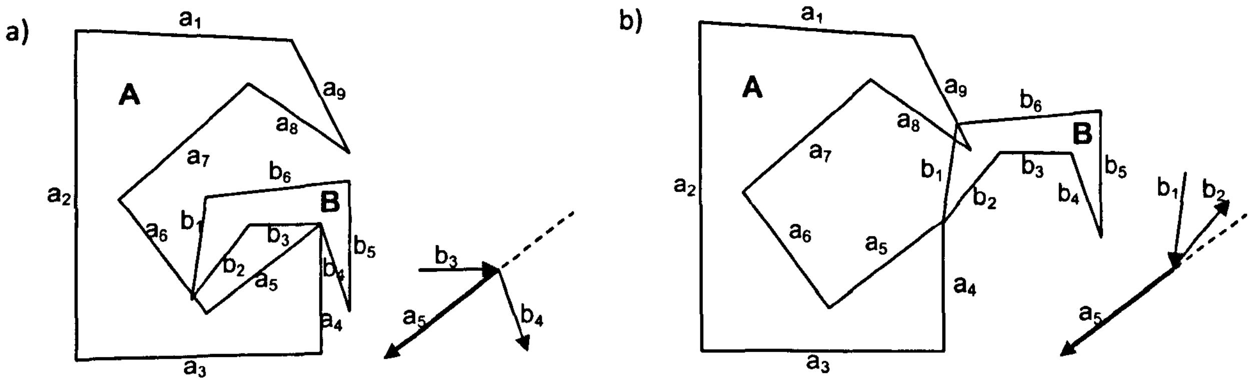


Figure 38. Vertex alignment of polygon B to polygon A using edge a_5: a) invalid alignment, b) valid alignment

Assuming both connected edges are right of edge e and polygons A and B intersect it is possible to attempt to resolve the overlap by translating the orbiting polygon along the translation vector defined by e. As with the previous section vector trimming can now be employed (the procedure is identical). The resultant translation vector can then be applied to slide polygon B along edge e and then another intersection test is performed. If the two polygons still intersect then the process repeats with the translation vector derived from the touching point to the end vertex of edge e. If the intersection has been resolved then the reference point of polygon B is a potential start position. If the entirety of edge e is traversed and there is still an intersection then no feasible starting position can be found along edge e and the aligned vertex / connected edges of polygon B. The next vertex of polygon B is then considered and so on until all edge vertex possibilities have been examined. Note it is also important to examine the edges of polygon B with the vertices from polygon A but as this is an identical procedure, this will not be discussed. Figure 39 shows this process on the example presented in Figure 38b. When aligned using a_5 and the vertex connecting edges b_1 and b_2, the two polygons are in an initially intersecting position so we need to resolve the intersection along the vector derived from edge a_5.

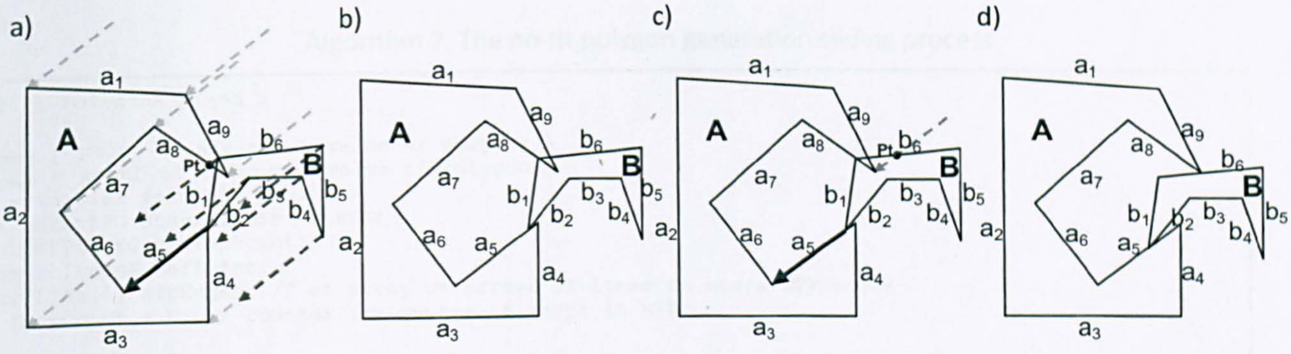


Figure 39. Start point generation process

Figure 39a shows the trimming process and identification of the closest intersection point, Pt. Polygon B is then translated by the trimmed translation vector, resulting in smallest feasible translation. The resulting position is shown in Figure 39b. The polygons are still intersecting so the procedure must repeat. The translation vector is calculated from the touching point to the end vertex of edge a_5 and is then trimmed as shown in Figure 39c (only the important trim intersection is shown). After applying this translation to polygon B, the polygons are no longer intersecting and therefore a feasible start point has been found and the standard orbital approach can be employed once more to generate the no-fit polygon loop.

In this new technique the outer no-fit polygon loop is first generated using the approach in section 5.3 and *then* the starting position procedure applied to the untraversed / unflagged edges from both polygon A and polygon B in turn. During the process, as soon as a feasible start position is found, the inner loop of the no-fit polygon is calculated and its edges are flagged edges as they are traversed as before. This ensures that the algorithm is fast because more edges become flagged as new no-fit polygon loops are produced, thus reducing the computational requirements for generating new feasible start positions. The procedure repeats until all edges have been seen or no more feasible start positions are available and the complete no-fit polygon is returned.

5.5. Summary - The No-Fit Polygon Algorithm

Given the functionality outlined in the previous sections we can describe no-fit polygon generation using the following pseudo code:

Algorithm 7. The no-fit polygon generation sliding process

```
Input : Polygons A and B

$Pt_{A(ymin)}$ = point of minimum x value of polygon A
$Pt_{B(ymax)}$ = point of maximum x value of polygon B
IFP = initial feasible position
Bool bStartPointAvailable = true;
Point NFPLoopStartRefPoint;
Point PolygonB_RefPoint;
Array[Line[]] nfpEdges; // an array of arrays of lines to store NFP edges
Int loopCount = 0; // counter for number of loops in NFP

Begin

Place polygons in IFP using translation $Pt_{A(ymin)}$ - $Pt_{B(ymax)}$
NFPLoopStartRefPoint = $Pt_{B(ymax)}$;
PolygonB_RefPoint = $Pt_{B(ymax)}$;

While(bStartPointAvailable)
{
        bStartPointAvailable = false;
        // find touching points & segments touching those points, generate touching structures

        Touchers[] toucherStructures = FindTouchers(A, B);

        // Eliminate non feasible touchers, ones that cause immediate intersection
        Touchers[] feasibleTouchers = CanMove(A, B, toucherStructures);

        // Trim feasible translations against polygon A and B
        Touchers[] trimmedTouchers = Trim(feasibleTouchers, A, B);

        // Sort trimmed translations by length
        Touchers[] lengthSortedTouchers = Sort(trimmedTouchers);

        //Translate polygon B along longest feasible translation vector
        B.Translate(lengthSortedTouchers[0].Translation);

        // Add translation to nfpEdges & mark traversed edge on static
        nfpEdges[loopCount].Add(lengthSortedTranslations[0].Translation);
        A.MarkEdge(lengthSortedTranslations[0].StaticEdgeID);

        If(NFPLoopStartRefPoint == PolygonB_RefPoint) // completed an NFP loop
        {
                Point nextStartPoint;
                // find next feasible start point - reset PolygonB_RefPoint to relevant point
                bStartPointAvailable= FindNextStartPoint(A, B, &nextStartPoint, &PolygonB_RefPoint);

                if(bStartPointAvailable)
                {
                        //Translate polygon B to nextStartPoint
                        B.Translate(PolygonB_RefPoint - nextStartPoint);

                        NFPLoopStartRefPoint = nextStartPoint;
                        loopCount++;
                }
        }
        else
        {
                bStartPointAvailable = true; // allow edge traversal to continue
        }
}
$NFP_{AB}$ = Complete(nfpEdges); //Reconstitute NFP from nfpEdges

End
```

Pseudo code for **FindNextStartPoint**:

Algorithm 8. Technique to find next start point in no-fit polygon generation

```
Input : PolygonA, PolygonB, reference to Point nextStartPoint, reference to Point
PolygonB_RefPoint

Int A_EdgeCount = PolygonA.EdgeCount;
Int B_EdgeCount = PolygonB.EdgeCount;
Edge StaticEdge;
Edge movingEdge;


Begin

for(int i = 0; i < A_EdgeCount; i++)
{
        if(PolygonA.IsEdgeMarked(i))
        {
                continue;
        }
        else
        {
                staticEdge = PolygonA.GetEdge(i);
        }

        for(int j = 0; j < B_EdgeCount; j++)
        {
                movingEdge = PolygonB.GetEdge(j);

                // translate the PolygonB so that movingEdge start in on start of the static edge
                PolygonB.Translate(movingEdge.Start - staticEdge.Start);

                Bool bFinishedEdge = false;
                Bool bIntersects = PolygonB.IntersectsWith(PolygonA);

                while(bIntersects AND !FinishedEdge)
                {
                        //Edge slide until not intersecting or end of staticEdge reached
                        Toucher currentToucher = MakeToucher(staticEdge.Start);
                        Toucher trimmedToucher = Trim(currentToucher, PolygonA, PolygonB);
                        PolygonB.Translate(trimmedToucher.Translation);

                        bIntersects = PolygonB.IntersectsWith(PolygonA);

                        if(bIntersects)
                        {
                                If(movingEdge.Start == staticEdge.End)
                                        bFinishedEdge = true;
                        }
                }
                // mark the traversed edge as seen (whether edge start point found or not)
                staticEdge.Mark(true);

                if(!bIntersects)
                {
                        // set the references to the points passed in to be the nextStartPoint
                        // and return true;
                        nextStartPoint = movingEdge.Start;
                        PolygonB_RefPoint = movingEdge.Start;

                        return true;
                }
        }
}
// all edges on moving tried on all unmarked edges on static, nothing found - no more starts
return false;

End
```

5.6. Problem Cases

This section includes discussion of each of the problem cases that cause difficulties with other methods and show why the presented approach is able to handle them without specific case-by-case implementations. In each figure in this section the stationary and orbiting polygons are shown in dark and light grey shading respectively and the reference point of the orbiting polygon is denoted by a black dot. This is used to trace the loops of the no-fit polygon (which are numbered).

5.6.1. Interlocking Concavities

The interlocking of concavities is a typical problem case for the previous orbital sliding approaches in the literature such as (Mahadevan, 1984). Figure 40 shows a screenshot of our implementation where we generate the complete no-fit polygon of identical shapes with one rotated through 180°. The shapes in these orientations involve many different interactions of the concavities and, therefore, multiple feasible start points. The no-fit polygon of these two shapes results in *six* loops within the no-fit polygon (one outer loop, 1, and five internal loops, 2-6).

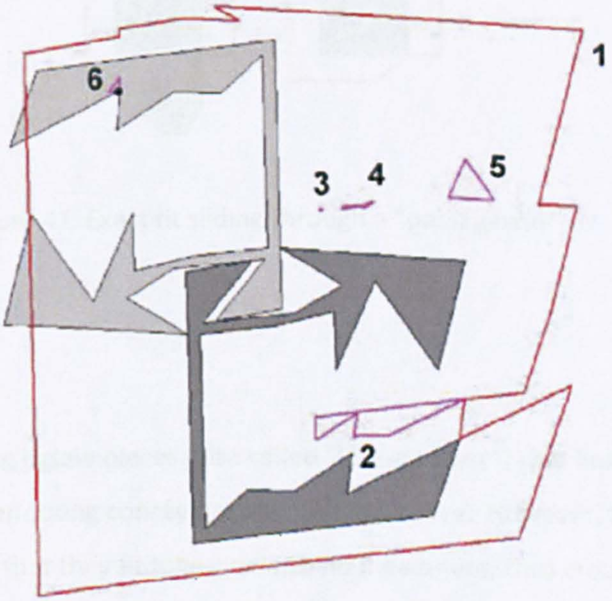


Figure 40. Multiple interlocking positions

5.6.2. Exact Fit: Sliding

The next case involves sliding through exactly fitting "passageways" which could not be handled by any contemporaneous approach at the time this approach was published, notably the Minkowski sum approach of (Bennell et al., 2001) or the orbital approach given by Mahadevan. This shortcoming was later overcome by an updated Minkowski sum approach presented in (Bennell & Song, 2008)

The extension to Mahadevan's algorithm that allows this sliding technique to deal with such exact fit problems requires the maintenance of the previously traversed edge and, thus, enabling consecutive edges of the stationary polygon to be used in the next translation iteration. We previously discussed another example of this case in Figure 34.

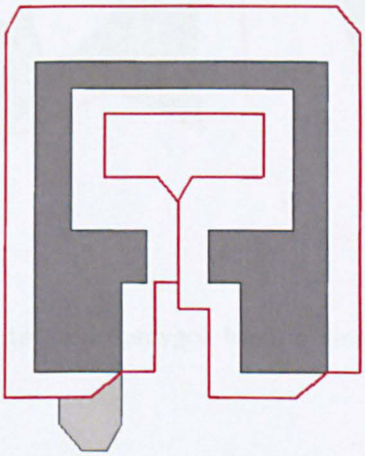


Figure 41. Exact fit sliding through a "passageway"

5.6.3. Exact Fit: Jigsaw Pieces

The problem case involving jigsaw pieces (also called "lock-and-key") that link exactly together is another form of the interlocking concavity case outlined above. However, the distinguishing feature of jigsaw pieces is that they fit together with no movement, thus creating a singular feasible point within the no-fit polygon (rather than an internal loop in the interlocking concavity case). In our approach, this position is found by the generation of feasible starting positions. Of course, the algorithm will still try to slide the orbiting polygon along edges but, in this case, there is no feasible translation vector therefore distinguishing that the position is just a singular feasible point location (see Figure 42). Most of the previous approaches have suffered from such degeneracy including: the Minkowski sum approach where no boundary exists to identify lock-and-key positions within the no-fit polygon, digitisation which suffers from loss of accuracy, convex decomposition whereby it is not possible to find such positions through recombination alone and the sliding algorithm of Mahadevan (identical difficulties as with interlocking concavities).

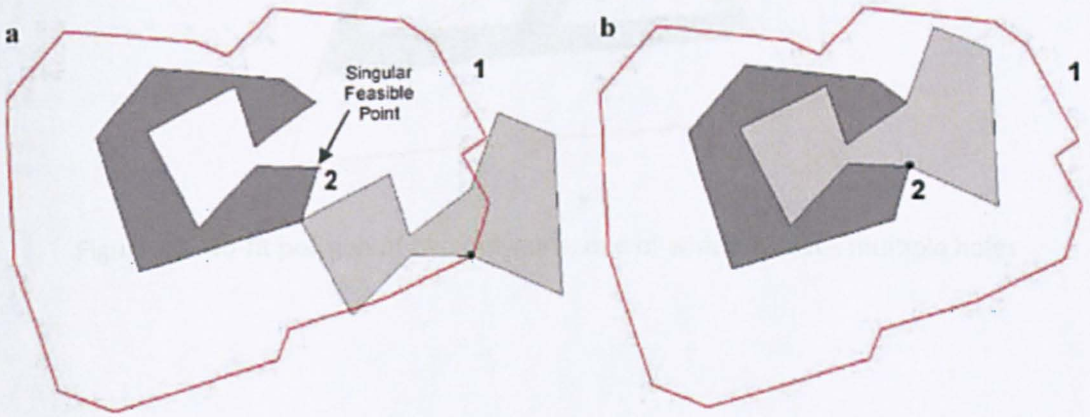


Figure 42. Jigsaw pieces: a) outer no-fit polygon loop, b) singular feasible internal position

5.6.4. Holes

There have been few approaches within the literature that can operate effectively on polygonal shapes containing holes. This could be due to the difficulty of the generation process for no-fit polygons involving holes. However, through the detection of feasible starting positions, the no-fit polygon can be generated easily and, more importantly, completely. Figure 43 shows an example involving a shape with two holes and a shape that can be placed within several distinct regions of the holes. The figure shows a mixture cases involving holes and interlocking concavities: loop 1 is the outer no-fit polygon loop, loops 2,3,5,6,7 are all hole cases, and loops 4 and 8 are cases whereby the concavity of the smaller orbiting shape interacts with the narrow gaps within the larger hole of the stationary shape.

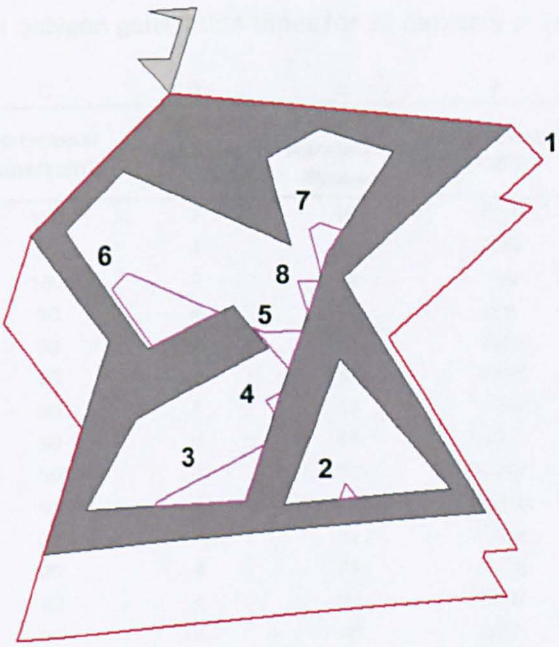


Figure 43. No-fit polygon of two polygons, one of which involves multiple holes

5.7. Generation Times on the Benchmark Problems

In order to demonstrate the speed and capabilities of the new no-fit polygon procedure, the following section reports the generation times for 32 benchmark problems from the literature. These datasets can be found on the EURO Special Interest Group on Cutting and Packing (ESICUP) website. For each problem the total generation time and number of no-fit polygons generated per second is reported (see Table 15). The "Logical Total Number of Shapes" (column E) is found by E = B * D and the "Total Number of NFPs" (column F) is calculated by F = E2 (i.e. the NFP for every logical shape against every other logical shape is calculated). All experiments were conducted on a Pentium 4 2GHz processor with 256MB RAM.

Table 15. No-fit polygon generation times for 32 datasets of the literature

A	B	C	D	E	F	G	H
Dataset	Number of Different Shapes	Rotational Constraints	Number of Rotations per Shape	Logical Total Number of Shapes	Total Number of NFPs	Total Generation Time (s)	NFPs Per Second
Albano180	8	180	2	16	256	0.32	800
Albano90	8	90	4	32	1024	0.71	1442
Blasz1	7	180	2	14	196	0.21	933
Blasz2	4	90	4	16	256	0.19	1347
Dagli	10	90	4	40	1600	0.93	1720
Dighe1	16	90	4	64	4096	1.28	3200
Dighe2	10	90	4	40	1600	0.62	2581
Fu	12	90	4	48	2304	0.52	4431
Jakobs1	25	90	4	100	10000	5.57	1795
Jakobs2	25	90	4	100	10000	5.07	1972
Mao	9	90	4	36	1296	1.41	919
Marques	8	90	4	32	1024	0.79	1296
Poly1a	15	90	4	60	3600	1.37	2628
Poly2a	15	90	4	60	3600	1.37	2628
Poly3a	15	90	4	60	3600	1.37	2628
Poly4a	15	90	4	60	3600	1.37	2628
Poly5a	15	90	4	60	3600	1.37	2628
Poly2b	30	90	4	120	14400	7.54	1910
Poly3b	45	90	4	180	32400	27.14	1194
Poly4b	60	90	4	240	57600	68.45	841
Poly5b	75	90	4	300	90000	141.90	634
Shapes	4	90	4	16	256	0.38	674
Shapes0	4	0	1	4	16	0.11	145
Shapes1	4	180	2	8	64	0.19	337
Shirts	8	180	2	16	256	0.33	776
Swim	10	180	2	20	400	6.08	66
Trousers	17	180	2	34	1156	0.73	1584
Profiles6	9	90	4	36	1296	0.86	1507
Profiles7	9	90	4	36	1296	0.58	2234
Profiles8	9	90	4	36	1296	0.56	2314
Profiles9	16	90	4	64	4096	44.30	92
Profiles10	13	0	1	13	169	0.55	307

Table 15 shows that the no-fit polygon constructs can be generated within reasonable time frames on most of the benchmark data. Typically the algorithm can generate about 1000 no-fit polygons per second although this can vary drastically depending on the number of line segments in each problem. For example, the two problems with the lowest no-fit polygons generated per second, "Swim" and "Profiles9", involve pieces with many line segments (up to 67) due to approximation of arcs and, further to this, "Profiles9" contains several shapes with holes (see Figure 44).

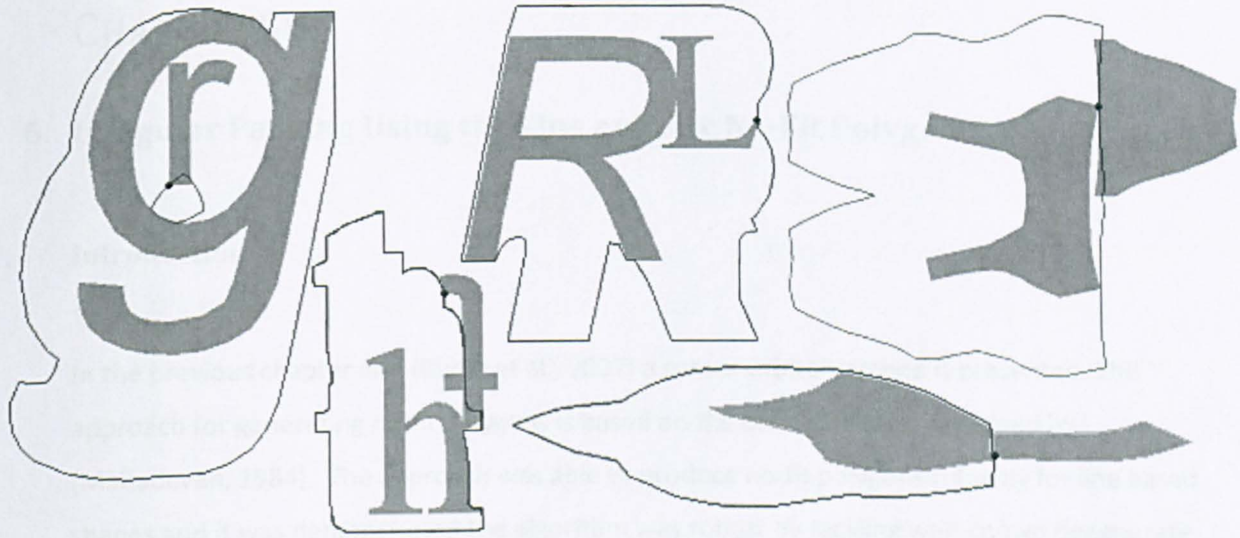


Figure 44. A selection of pieces and no-fit polygons from the "Profiles 9" (letters) and "Swim" datasets

The slowest overall generation times occur within the Poly4b and Poly5b problems and are due to the large number of no-fit polygons to generate. Other reported generation times for available benchmark problems of the literature can be found in (Bennell et al., 2001)

Whilst very large problems may require a few minutes to generate all of no-fit polygons, such as the Poly4b and Poly5b test problems (requiring 68 and 142 seconds respectively), for repeated automatic nesting it is likely that the layout algorithm will more than recover this overhead by being able to use the faster no-fit polygon based intersection testing.

5.8. Summary

This chapter has shown that the no-fit polygon is an important construct that can be used for the development of faster automated packing algorithms as opposed to the traditional trigonometric based approaches. A complete and robust implementation of the orbital method of the no-fit polygon has been presented. This approach can deal with degenerate cases that the previous methods cannot resolve easily. At the point of publication this was also the first time that an implementation of the no-fit polygon that has been rigorously investigated for robustness with the known degenerate cases and on such a wide range of benchmark data.

Chapter Six

6. Irregular Packing Using the Line and Arc No-Fit Polygon

6.1. Introduction

In the previous chapter and (Burke et al., 2007) a robust orbital method is presented. This approach for generating no-fit polygons is based on the orbital method proposed by (Mahadevan, 1984). The approach was able to produce no-fit polygons robustly for line based shapes and it was demonstrated the algorithm was robust by tackling well-known degenerate cases that had been identified as causing difficulties for previously published approaches. Figure 45 shows some of the degenerate cases that were handled using this procedure.

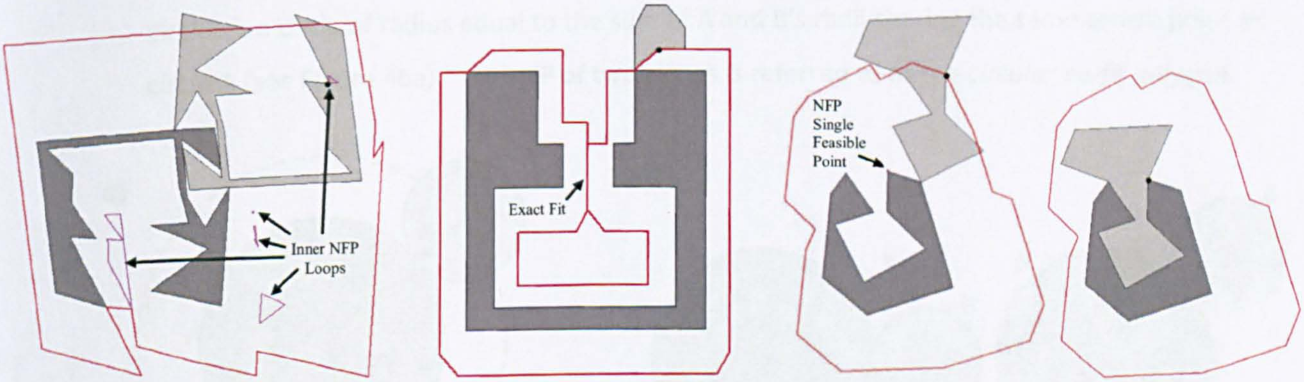


Figure 45. No-Fit Polygons generated using the approach presented by Burke et al. (2007)

This chapter describes the further extension of the approach presented in chapter five which allows for the generation of the line arc no-fit polygon for polygons containing both lines and arcs like those introduced in the new benchmark problems, profiles1-10, introduced in chapter four to test the effectiveness of the new bottom-left fill placement algorithm. This work was published in (Burke et al., 2010).

The introduction of circular arcs into the no-fit polygon generation problem was a requirement for the CASE and TCS as projects as the industrial partner's customers often cut parts incorporating arcs. Without adding support for line arc no-fit polygons it was determined that only a small percentage of problems could be tackled using the no-fit polygon approach. Furthermore the ability to generate spaced shapes, using circles in a no-fit polygon generation (see section 8.2), was a useful addition to the existing spacing techniques used by the industrial partner.

6.2. A Simplified Case involving Circles

In order to describe the arc modifications needed for no-fit polygon generation, it is beneficial to examine a case involving two circles. These concepts are then generalised to arcs in the remainder of this section. The no-fit polygon of two circles, A and B (the convention of shape B moving around shape A is maintained), is defined as the path followed by the reference point of B (the circle's centre point) when circle B orbits circle A. The no-fit polygon, NFB_{AB}, of two circles is a circle of radius equal to the sum of A and B's radii sharing the same centre point as circle A (see Figure 46a). The NFP of two circles is referred to as the *circular no-fit polygon*.

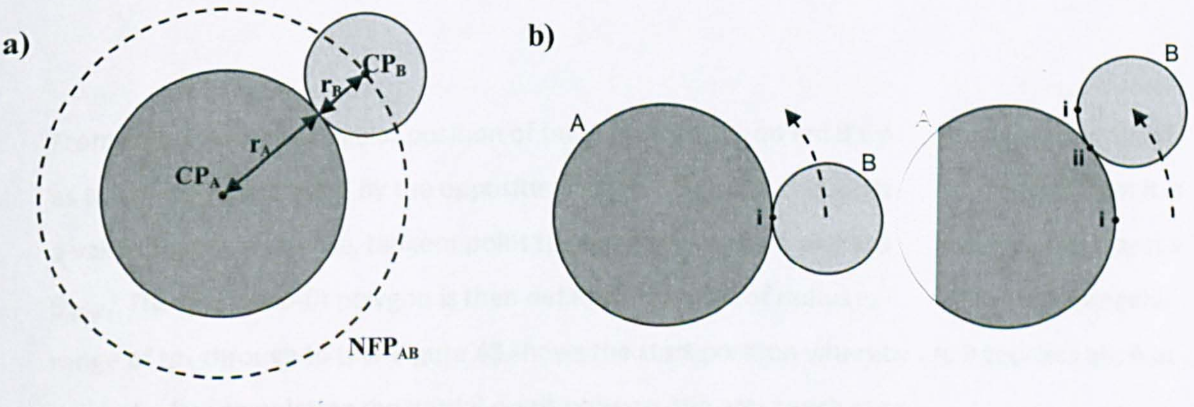


Figure 46. a) No-Fit Polygon of Two Circles, b) Movement of the Touch Position

The edges within line-only no-fit polygons are derived from the same edges as the two input shapes A and B. However, this is no longer the case with arcs as the no-fit polygon does not take the form of the shape edges from which it was derived but rather assuming properties from *both* interacting edges (namely the radii). This is because the touching position moves on *both* arcs as B moves around A (see Figure 46).

6.3. Generalisation to Arcs

The circular no-fit polygon example presented in Figure 46 can be generalised to circular convex arcs by eliminating the segment of the no-fit polygon for which the arcs will not touch on their radial extents. This can be created by examining the start and end angles of the two arcs (see Figure 47a).

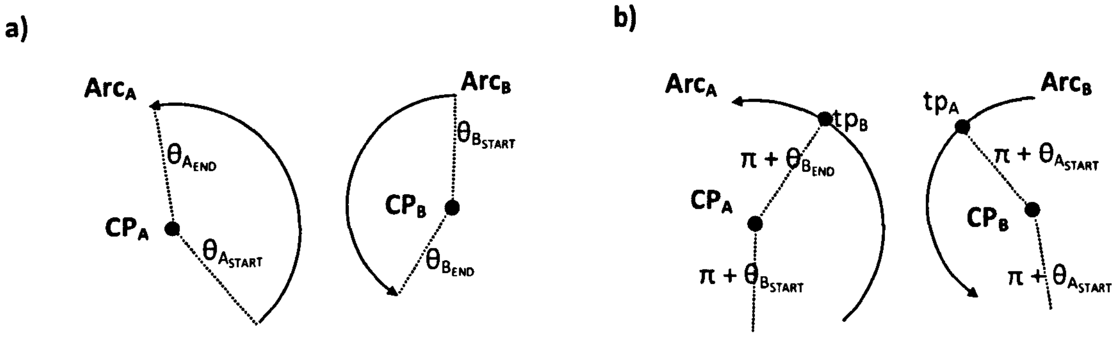


Figure 47. Finding circle tangents (A and B are convex)

From Figure 47b, the angular position of tangent point tp_A on arc B's parent circle is identified as the angle represented by the opposite angle, $\pi + \theta_{AStart}$. If this angle exists on arc B then it is a valid tangent. Likewise, tangent point tp_B's angular position on parent circle A is given by $\pi + \theta_{BEnd}$. The partial no-fit polygon is then defined as the arc of radius $r_A + r_B$ through the angular range of tp_A through to tp_B. Figure 48 shows the start position whereby arc B touches arc A at tp_A and, after completing the partial no-fit polygon, the arcs touch at tp_B.

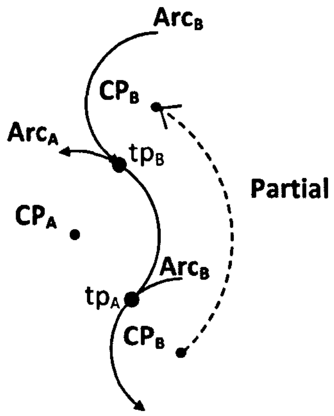


Figure 48. Creating the partial no-fit polygon circle from the arc tangents

If there are no tangent points, then the two arcs cannot interact to create any of the circular no-fit polygon (see Figure 49).

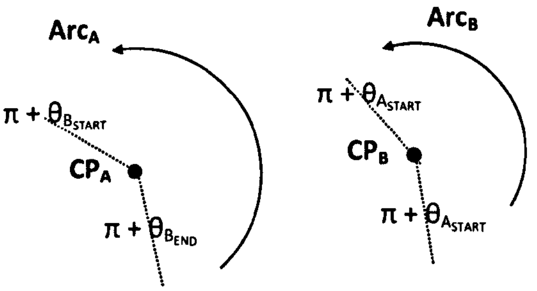

Arc$_A$
$\pi + \theta_{B_{START}}$
CP$_A$
$\pi + \theta_{B_{END}}$
Arc$_B$
$\pi + \theta_{A_{START}}$
CP$_B$
$\pi + \theta_{A_{START}}$

Figure 49. No arc tangent points available

A further example must be examined whereby one of the circles exists in its concave form (i.e. a hole) as this will allow for generalisations to be made for concave arcs. Figure 50 shows an example of the no-fit polygon produced when a circle travels around the inside edge of another circle. In Figure 50a, the no-fit polygon is a circle once again, but this time it has a radius equal to $r_A - r_B$. Its centre point is the same as A's centre point. In order to create the partial circular no-fit polygon in this instance, the same tangent point procedure can be repeated except that the start/end angles are not modified by π. The partial circular no-fit polygon is defined by the arc of radius $r_A - r_B$ and start and end angles defined by the tangent point angles, tp$_B$ and tp$_A$ respectively (see Figure 50b).

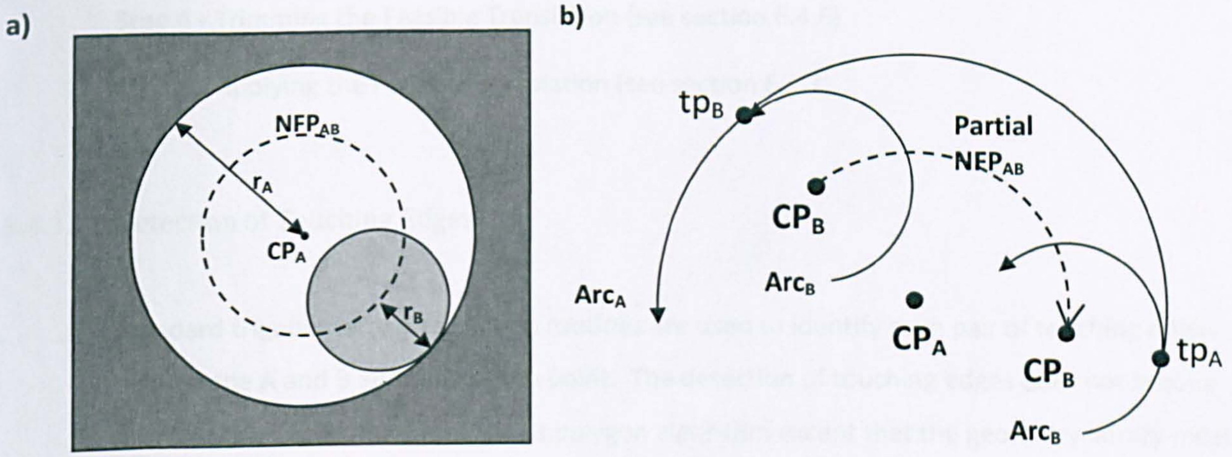


Figure 50. a) No-fit polygon of two circles in the concave case, b) Creating the partial no-fit polygon where arc A is concave and arc B is convex

In the case where arc A is convex and B is concave, the partial circular no-fit polygon can be obtained by the same procedure as above. For this, arc A and B must be switched such that arc B is now the convex arc and A is concave. Now by rotating the resultant partial circular no-fit polygon by 180°, the correct arc section for the no-fit polygon is produced. This is possible because, as B moves in one direction, A can also be thought of as moving relatively in the opposite direction. If one concave and one convex arc are involved then a partial circular no-fit polygon is only possible if the concave arc has a larger radius than the convex arc.

It is not physically possible for two concave arcs to orbit each other via the circular no-fit polygon as the shapes with which they belong would have to be intersecting.

6.4. Modifying the Orbital No-Fit Polygon Approach

With the circular no-fit polygon described and its generalisation to arcs, the modifications that are required to extend the orbital no-fit polygon generation algorithm are presented in the following sections. The arc modifications are described using the algorithmic steps defined in the previous chapter and in (Burke et al., 2007). The steps are as follows:

Step 1 - Detection of Touching Edges (see section 6.4.1)

Step 2 - Creation of Potential Translation Vectors (see section 6.4.2)

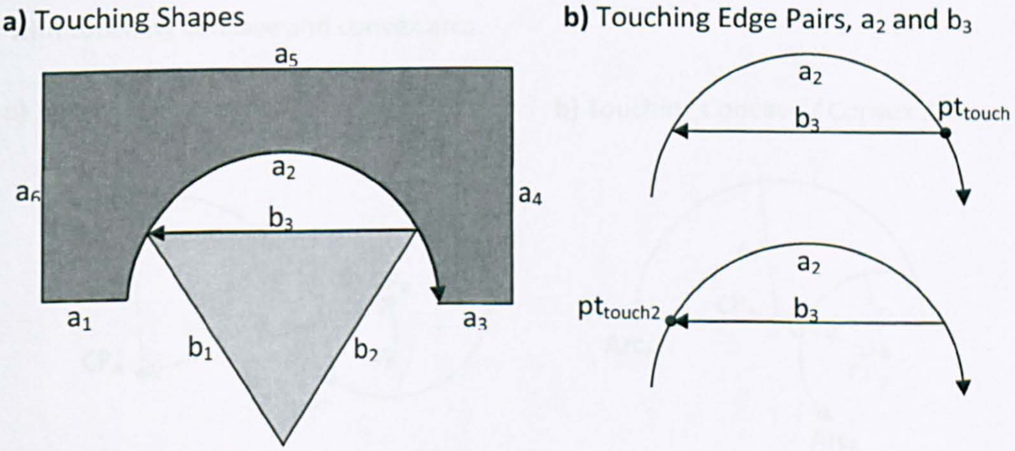
Step 3 - Finding a Feasible Translation (see section 6.4.3)

Step 4 - Trimming the Feasible Translation (see section 6.4.6)

Step 5 - Applying the Feasible Translation (see section 6.4.7)

6.4.1. Detection of Touching Edges

Standard trigonometry intersection routines are used to identify each pair of touching edges from shape A and B and their touch point. The detection of touching edges does not require modification of the line-based no-fit polygon algorithm except that the geometry library must be able to detect that lines are touching arcs or that arcs are touching other arcs. When concave arcs are involved, there is a possibility of two edges touching at two separate points. However, this complication can be eliminated by storing two separate touching edge pair structures for each of the touching points as shown in Figure 51.


a) Touching Shapes

b) Touching Edge Pairs, a_2 and b_3

Figure 51. Touching edges with two distinct touch positions (edge a_2 and b_3)

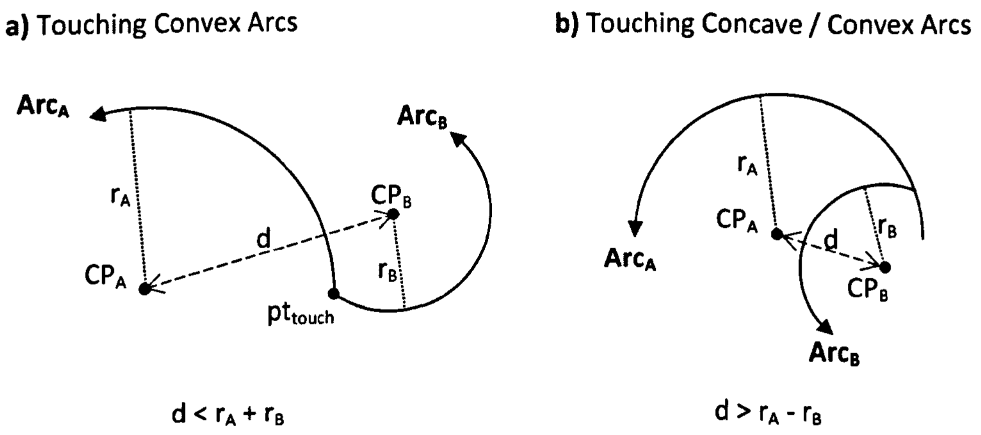
In Figure 51a, both the start and end points of line, b_3, touch the concave arc edge, a_2. Therefore, two touching entries are stored in the touching edge pair list to reflect this and are shown by Figure 51b. No further modifications are required for this stage of the algorithm.

6.4.2. Creation of Potential Translation Vectors

Step 2 involves the creation of potential translation vectors using the touching edge pairs that were identified in step 1. In this stage the inclusion of arc edges requires a number of modifications based on the notion of the partial circular no-fit polygon (described in section 6.3). These modifications are outlined in the next three subsections in a case by case manner.

6.4.3. Potential Translation Vectors for Touching Pairs Involving Two Arcs

As shown previously the circular no-fit polygon can easily be created when convex arcs are touching at their tangent points (see section 6.3). This is when the distance between the two arc centres is equal to the sum of the two radii (the maximal distance where the arcs can touch). In the case of touching concave and convex arcs, the concave radius must be greater than the convex radius and the distance between the two arc centres must be equal to the convex radius subtracted from the concave radius. In these situations, the potential translation is simply the partial circular no-fit polygon. However, arcs may also touch such that the distance between the centre points does not allow the partial circular no-fit polygon to be used. Examples of this situation are shown in Figure 52a with touching convex arcs and Figure 52b with touching concave and convex arcs.


a) Touching Convex Arcs

b) Touching Concave / Convex Arcs

$d < r_A + r_B$

$d > r_A - r_B$

Figure 52. Non-tangential touching arcs

In such situations two potential translations can be derived from firstly the stationary arc, A, and then the orbiting arc, B.

There are three possible cases:

i) The two arcs both touch on start/end points

ii) Arc B's start/end point touches the between the ends of arc A

iii) Arc A's start/end point touches the between the ends of arc B.

Cases (ii) and (iii) will be examined first as these are the most simple within the line case. In section 5.3.2, given two touching lines, A and B, the potential translation vector for case (ii) was the line defined by the touching point to the end point of line A (stationary line). The same principle is used for the arc case (ii), the potential translation vector is the arc defined by the touch point to the end point of arc A (stationary arc) which also has the same centre point and radius of arc A. Figure 53 shows an example of a potential translation vector for case (ii).

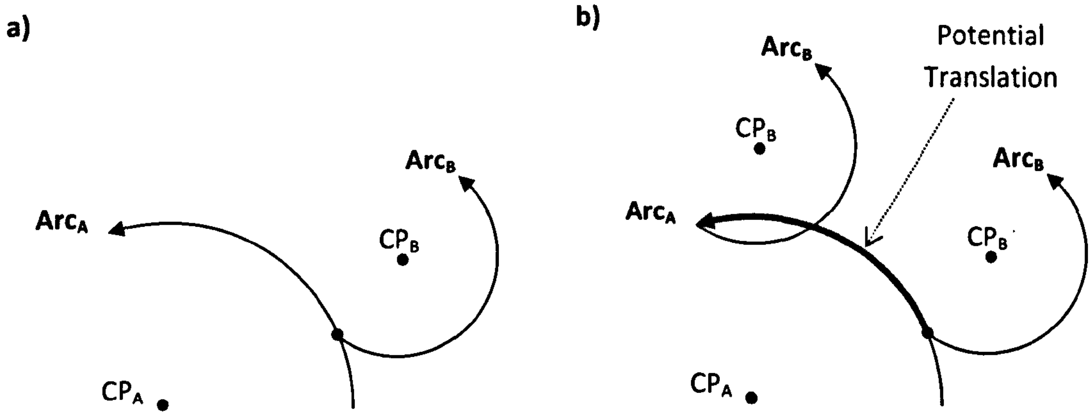


Figure 53. Potential translation vector derived in case (ii): arc B's start/end point touches the middle of arc A.

However, as shown in Figure 53b, this potential translation could result in arc B intersecting with arc A. On further examination, this can be explained because a tangent point exists before the end point of arc A (see Figure 54a). Therefore, the potential translation should be trimmed to any tangent point that exists along the potential translation (Figure 54b). Assuming that the translation of Figure 54 is performed in full, the next potential translation can now be derived from the circular no-fit polygon of the two arcs (i.e. the distance between the arc centre points is equal to $r_A + r_B$ in the convex case and equal to $r_A - r_B$ in the concave case).

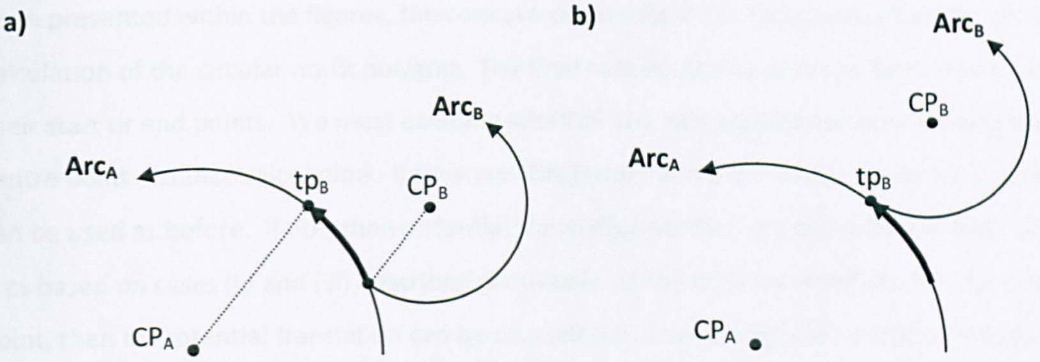


Figure 54. Potential translation vector derived in case (ii) when a tangent point is present: arc B's start/end point touches between the ends of arc A

The same principles found in case (ii) also apply for case (iii) whereby the start/end point of arc A touches between the ends of arc B (see Figure 55a). It is necessary to firstly examine whether a tangent point exists from the touching point towards the end of arc A. If this is the case then the partial circular no-fit polygon can be derived. If not then an arc is created from the touch point to the end point of arc B (with the same centre point and radius of B). However, this defines the potential translation vector with which arc A moves (relatively) so the translation vector must be rotated by 180° to obtain the correct translation vector for arc B (see Figure 55b).

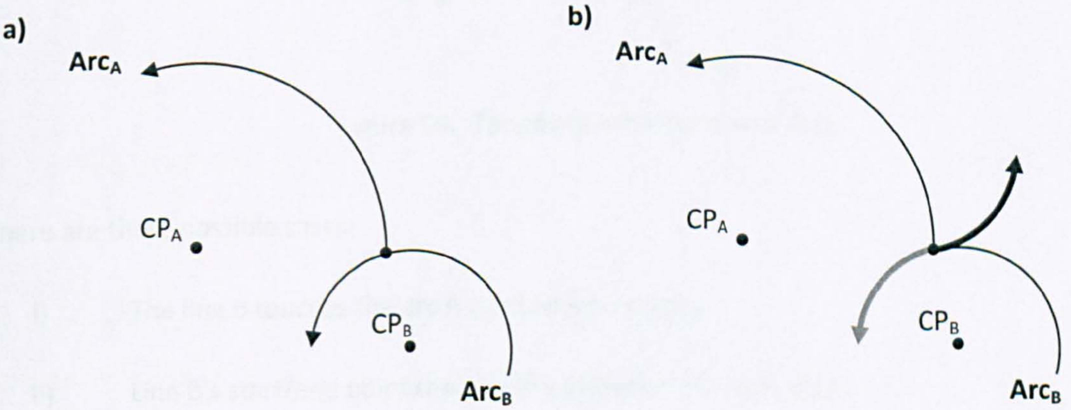


Figure 55. Potential translation vector derived in case (iii): arc A's start/end point touches the middle of arc B

From Figure 55b, it can be seen that when arc B is translated by the potential vector, the two arcs remain in contact, without intersecting, throughout arc B's translation. Once again, this only holds true if there are no tangential positions along the translation. If one exists, the translation should be trimmed (as with the previous case). Although only convex cases have been presented within the figures, the concave cases follow the same procedure except for the calculation of the circular no-fit polygon. The final case is case (i) whereby both arcs touch on their start or end points. We must examine whether the two arcs are tangential using the centre point distance calculation. If they are tangential then the partial circular no-fit polygon can be used as before. If not, then potential translation vectors are derived from both of the arcs based on cases (ii) and (iii) described previously. If the arc's touching point is the end point, then no potential translation can be created (i.e. a zero length arc translation would be created as the end point is also the touch point). Also, if both arcs touch on their start points, then two potential translation vectors will be produced (one from arc_A and one from arc_B).

6.4.4. Potential Translation Vectors for Touching Pairs Involving a Line and an Arc

Touching lines and arcs are less complicated than the touching arcs case but tangential points must still be identified (see Figure 56).

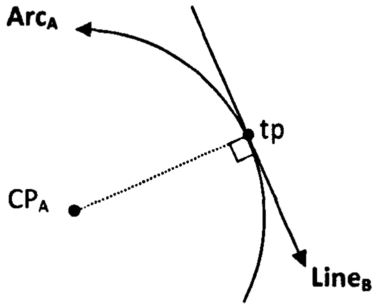


Figure 56. Tangents with lines and arcs

There are three possible cases:

i) The line B touches the arc A on start/end points

ii) Line B's start/end point touches the between the ends of arc A

iii) Arc A's start/end point touches the between the ends of line B.

A potential translation arc is created from the touch point to the end point of arc A in case (ii) where no tangent point exists, and from the touch point to the tangent point where one does exist. It is important to detect whether there is a tangent point to avoid translations that result in intersections. In case (iii) the potential translation vector is defined by the end point of line B to the touch point (tangent points are not required here). Finally, case (i) is handled by the techniques from cases (ii) and (iii). Once again, two potential translation vectors are produced (one derived from the arc and one from the line). Once again, no potential translation can be derived from a primitive whereby its end point is also the touching point.

6.4.5. Finding a Feasible Translation

The next stage is to find a feasible translation from the set of potential translations. A potential translation is only feasible if it is feasible for each of the touching edge pairs. In chapter four a set of rules was developed to indicate translation feasibility through the use of left/right line tests. Examples of these are reproduced in Figure 57 as they will also be used when handling arcs. The touching point can either be at the start, end or in the middle of an edge (the touching point for each edge is identified above each diagram in Figure 57).

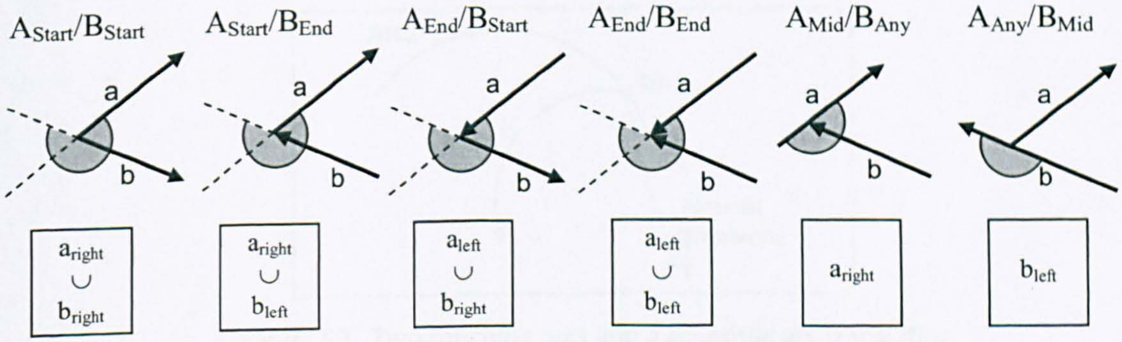


Figure 57. Identifying the feasible angular range of translations (indicated by the arc) for different touch categories

Now that arcs are involved, it initially seems that the problem is more complicated. However, the line tests shown in Figure 57 can be used providing that any arcs within the touching edge pairs and potential translations are reduced to a tangential line at the touch point. Figure 58 shows how the tangential line is created depending on the position of the touching point on the arc: i) at a touching start point, the tangential line must start at the touch point, ii) a mid-arc touch results in a tangential line with its midpoint on the touch point and iii) a touching end point requires that the tangential line also ends at the arc's end point.

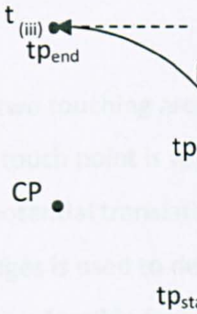


Figure 58. Conversion to tangential lines at the touch point of an arc

It is important that the directionality of the arc is maintained when creating the tangential line for the method of section 5.3.3 to be applicable (the length of the tangential line does not matter as it is only used for left/right tests). Figure 59, Figure 60 and Figure 61 demonstrate the procedure.

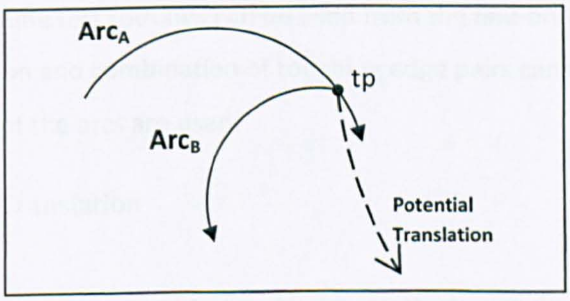


Figure 59. Two touching arcs and a potential arc translation

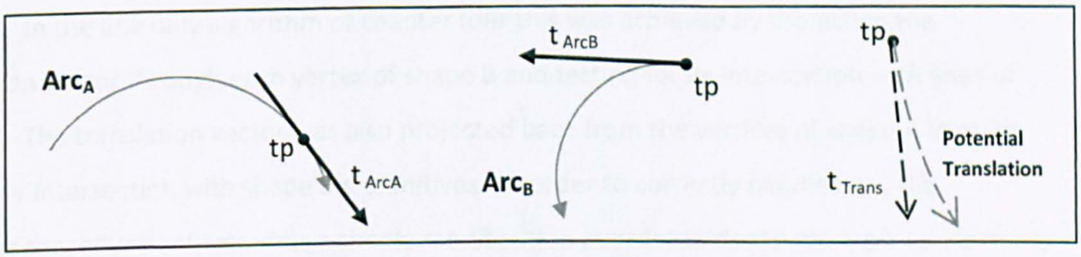


Figure 60. Creating the tangential lines

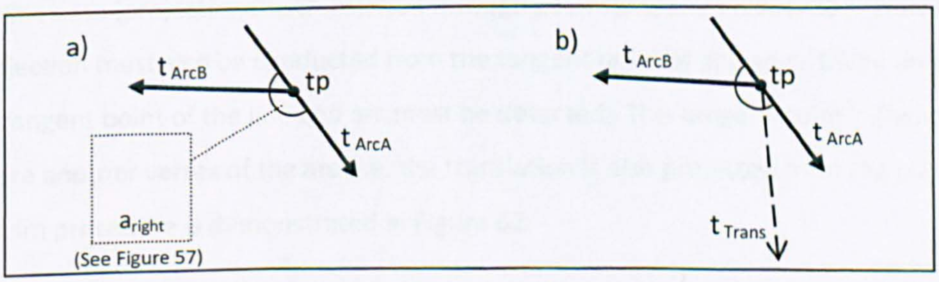


Figure 61. Revisiting the line method using tangential lines

Figure 59 shows an example of two touching arcs and a potential arc translation (arc_A is concave, arc_B is convex and the touch point is tp). In Figure 60, tangential lines are created for the two touching arcs and the potential translation arc. In Figure 61a, the feasibility test from chapter five for touching line edges is used to define the feasible region and Figure 61b shows that the potential translation arc is feasible for this pair of touching edges. In that providing that both the edges from polygon B (Arc B) and the potential translation (t_{Trans}) are right of the static primitive (Arc A) we can accept the translation as feasible.

In order to see why tangential lines can be used to determine the feasibility of the movements, we must examine what is being calculated. To show that a potential translation is feasible for a particular edge pair, we need only show that the translation will not immediately result in an intersection. As the tangential lines of arc primitives define the vector path of the arc at a particular position, the same test routines can be used from the line-only case. The feasibility of any potential translation and combination of touching edge pairs can be identified provided that the tangential lines of the arcs are used.

6.4.6. Trimming the Feasible Translation

The last step before polygon B can be translated is to trim the translation vector. This is important as there may be other edges that could interfere with the translation of the orbiting polygon. In the line only algorithm of chapter four this was achieved by projecting the translation vector through each vertex of shape B and testing for its intersection with lines of shape A. The translation vector was also projected back from the vertices of shape A in order to test for intersection with shape B's primitives. In order to correctly handle arcs, the trimming procedure only requires a simple modification in order to deal with: i) arc to arc trims, ii) line to arc trims and iii) arc to line trims. For each of these, the same approach is taken as with the line case (projecting the translation through each start/end vertex etc.). However, an extra projection must also be conducted from the tangent point of any arcs. Given an arc and a line, the tangent point of the line and arc must be detected. This tangent point is then treated as if it were another vertex of the arc (i.e. the translation is also projected from the point). The line/arc trim procedure is demonstrated in Figure 62.

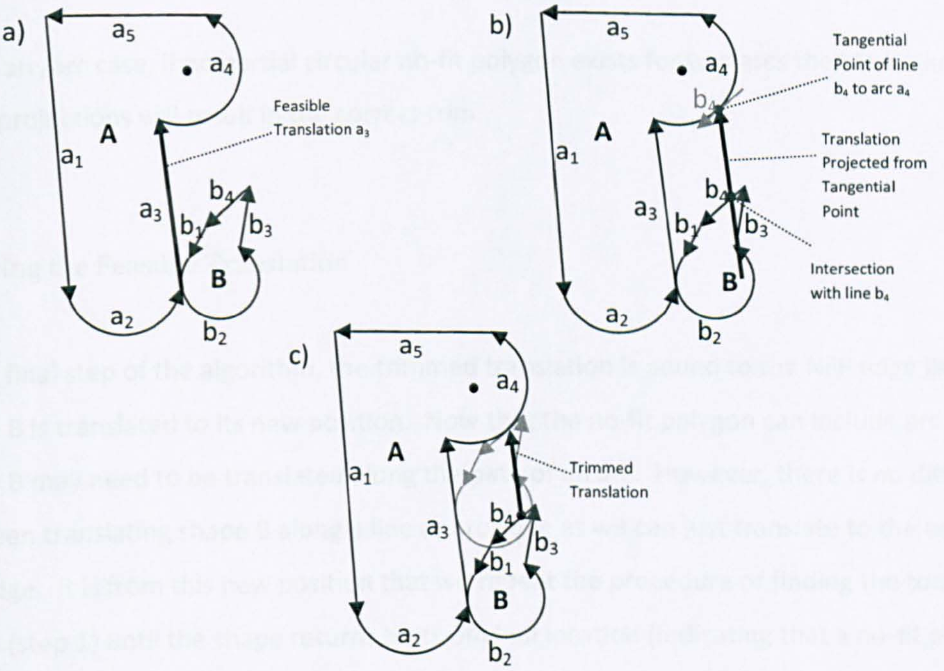


Figure 62. Tangential trimming of the feasible translation from an arc and a line, a_4 and b_4

When tri t and addi he partial circular no-fit polygon of the two arcs. If there is an intersection, then the translation is trimmed as normal. The process for an arc/arc trim is demonstrated in Figure 63.

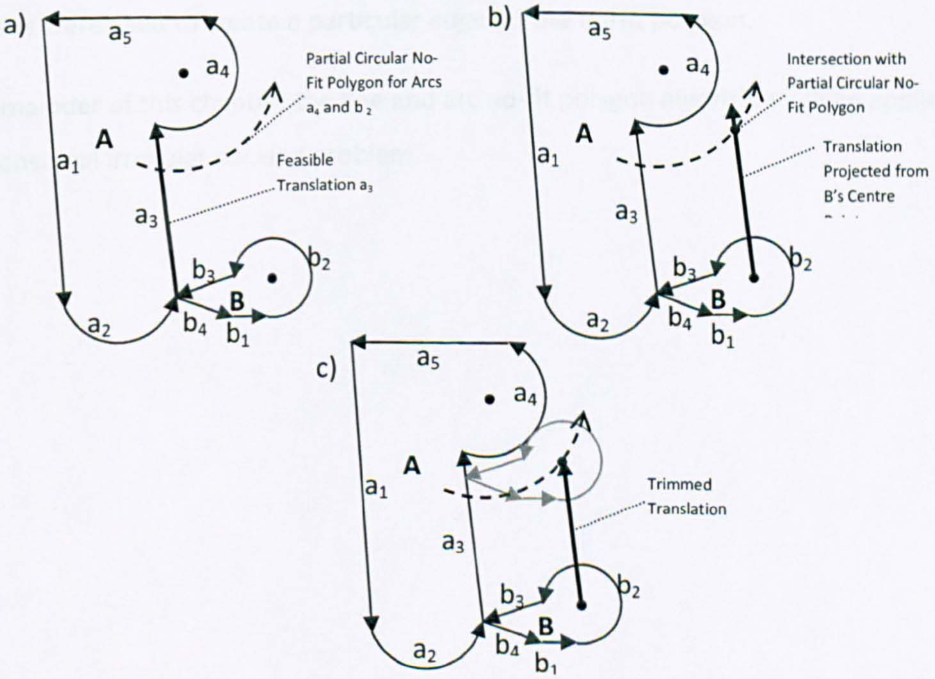

a)
a5
a4
Partial Circular No-Fit Polygon for Arcs a4 and b2
A
a1
Feasible Translation a3
a3
b3
b2
B
a2
b4
b1
b)
a5
a4
Intersection with Partial Circular No-Fit Polygon
A
a1
Translation Projected from B's Centre
a3
b3
b2
B
a2
b4
b1
c)
a5
a4
A
a1
Trimmed Translation
a3
b3
b2
B
a2
b4
b1

Figure 63. Tangential trimming of the feasible translation from two arcs

In the arc/arc case, if no partial circular no-fit polygon exists for two arcs then the start/end point projections will result in the correct trim.

6.4.7. Applying the Feasible Translation

In the final step of the algorithm, the trimmed translation is added to the NFP edge list and shape B is translated to its new position. Now that the no-fit polygon can include arc edges, shape B may need to be translated along the path of an arc. However, there is no difference between translating shape B along a line or arc edge as we can just translate to the end point of the edge. It is from this new position that we repeat the procedure of finding the touching edges (step 1) until the shape returns to its original location (indicating that a no-fit polygon loop has been created).

The previous sections identified the modifications required to allow the orbital approach of chapter four to be able to generate no-fit polygons for shapes that also contain arc edges. Figure 64 shows examples of no-fit polygons that have been generated using the proposed line and arc no-fit polygon algorithm. Included in Figure 64 are annotations that explain what procedures were used to create a particular edge on the no-fit polygon.

In the remainder of this chapter, the line and arc no-fit polygon algorithm will be applied to the two dimensional irregular packing problem.

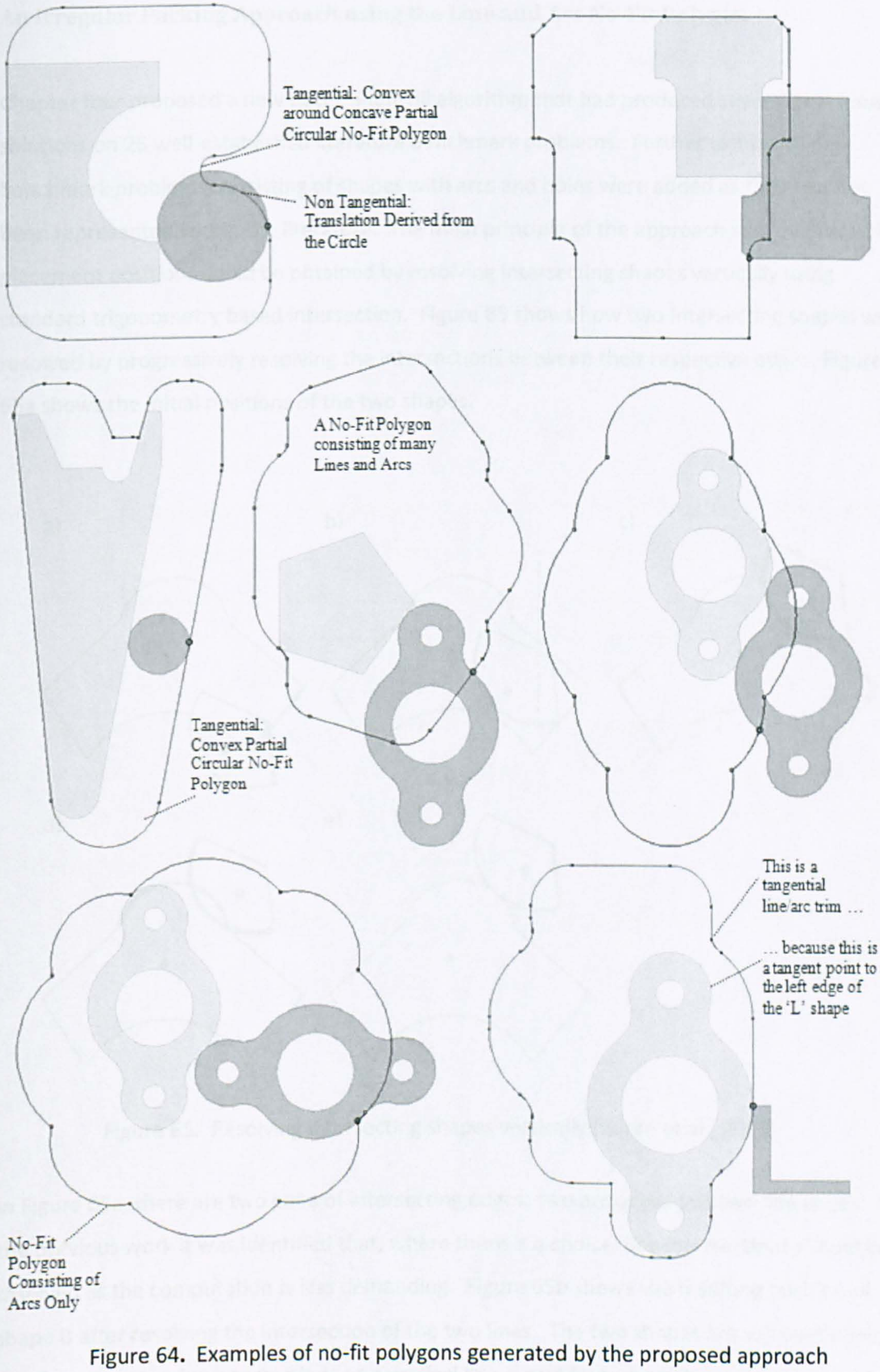

Tangential: Convex around Concave Partial Circular No-Fit Polygon
Non Tangential: Translation Derived from the Circle
A No-Fit Polygon consisting of many Lines and Arcs
Tangential: Convex Partial Circular No-Fit Polygon
This is a tangential line/arc trim ...
... because this is a tangent point to the left edge of the 'L' shape
No-Fit Polygon Consisting of Arcs Only

Figure 64. Examples of no-fit polygons generated by the proposed approach

6.5. An Irregular Packing Approach using the Line and Arc No-Fit Polygon

Chapter four proposed a new bottom-left-fill algorithm that had produced several best known solutions on 26 well-established literature benchmark problems. Further to this, 10 new benchmark problems consisting of shapes with arcs and holes were added as they had not been represented within the literature. The main principle of the approach was that feasible placement positions could be obtained by resolving intersecting shapes vertically using standard trigonometry based intersection. Figure 65 shows how two intersecting shapes were resolved by progressively resolving the intersections between their respective edges. Figure 65a shows the initial positions of the two shapes.

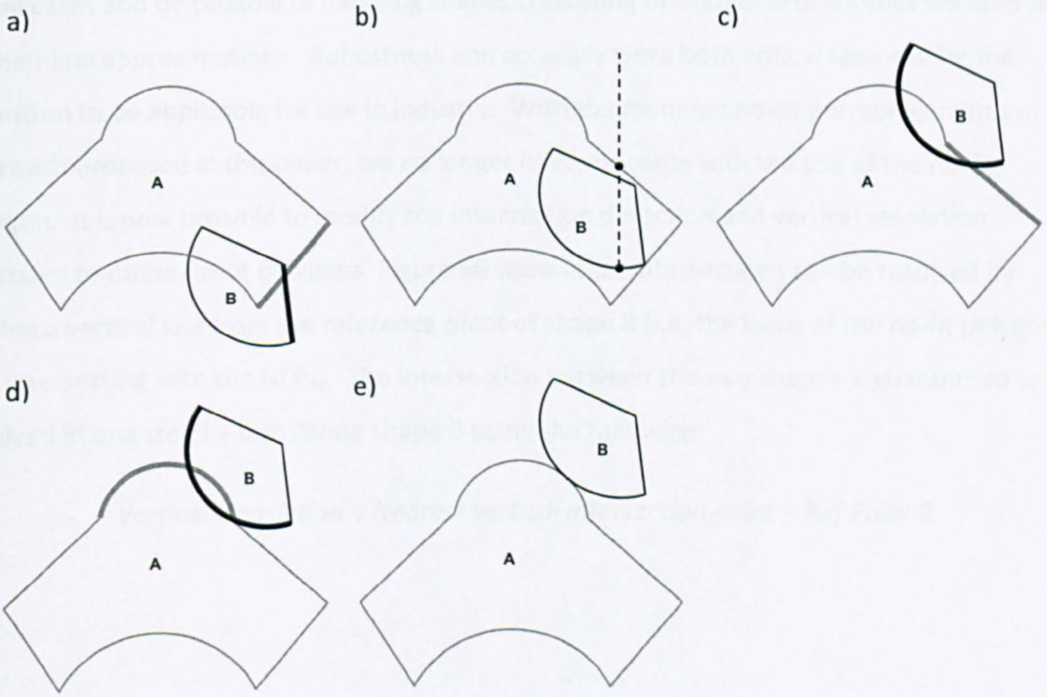


Figure 65. Resolving intersecting shapes vertically (Burke et al., 2006)

In Figure 65a, there are two pairs of intersecting edges: two arc edges and two line edges. In the previous work it was identified that, where there is a choice, line intersections should be resolved as the computation is less demanding. Figure 65b shows the resulting position of shape B after resolving the intersection of the two lines. The two shapes are still overlapping but there are no edge intersections so a vertical line is cast from the bottom-most point of shape B and tested for intersection with shape A. Shape B is translated vertically by the distance between the nearest intersection point and the bottom-most point. Figure 65c shows that this results in a new pair of edges intersecting (a line from shape A and an arc from shape B). Figure 65d shows the final arc-arc intersection and Figure 65egives the final non-intersecting configuration for these two shapes. For specific implementation details and pseudo code see chapter four.

In contrast to the standard trigonometry based intersection detection, no-fit polygons can be used to quickly identify the intersection state of two shapes by a simple point-in-polygon test. If the reference point of shape B is inside the no-fit polygon NFP_{AB}, then the two shapes are known to be intersecting. In chapter four standard trigonometry intersection detection was used as, at that time, there were no approaches that could generate no-fit polygons robustly for all cases and be capable of handling shapes consisting of circular arcs without decomposing to their line approximations. Robustness and accuracy were both critical features for the algorithm to be applicable for use in industry. With the modified no-fit polygon generation approach proposed in this paper, we no longer have concerns with the use of the no-fit polygon. It is now possible to modify the intersection detection and vertical resolution approach to utilise no-fit polygons. Figure 66 shows that intersections can be resolved by casting a vertical line from the reference point of shape B (i.e. the locus of the no-fit polygon) and intersecting with the $\mathrm{NFP}_{AB.}$ The intersection between the two shapes is guaranteed to be resolved in one step by translating shape B using the following:

Vertical translation = Nearest vertical intersection point – Ref Point B

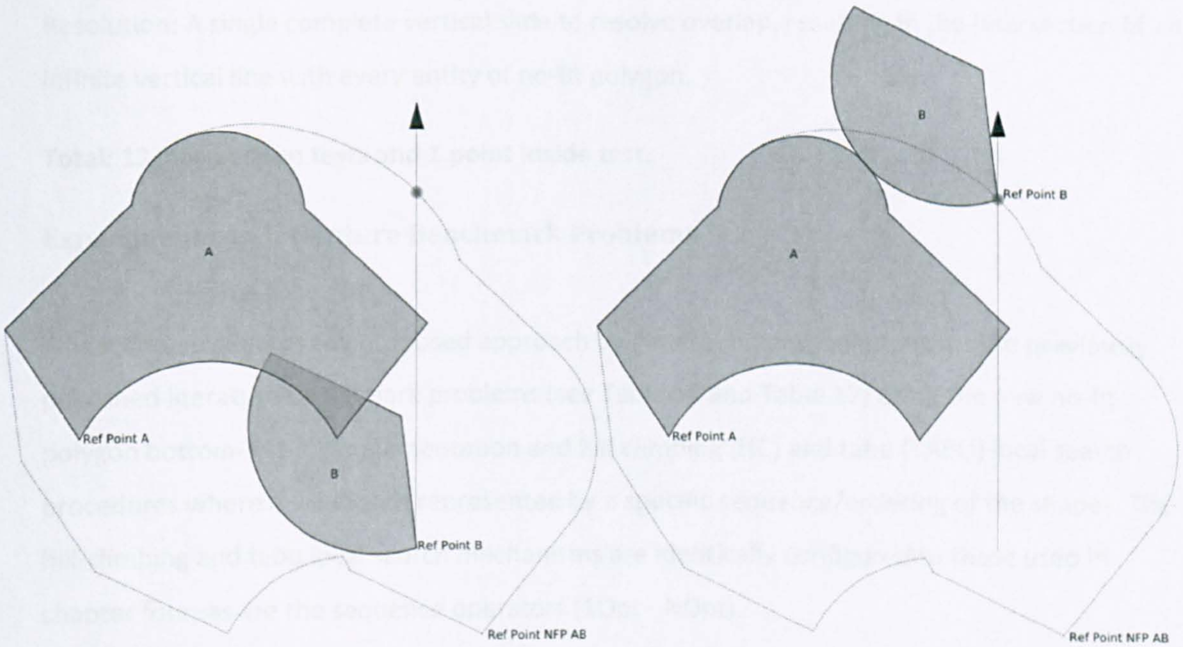


Figure 66. Resolving intersecting shapes with the arc and line no-fit polygon

The following concrete example demonstrates the relative efficiency of overlap resolution for an instance of two overlapping shapes using the entity by entity resolution approach used in the technique presented in chapter four and the same overlap resolution when using the no-fit polygon. For the shapes shown in Figure 65, using the stepwise overlap resolution, the detection of an invalid position and the vertical resolution requires:

Detection: Full entity by entity intersection test and if no intersection is found all points in moving shape to be tested for containment against static shape. In the worst case for our example shapes this would result in 18 intersection tests and 3 point inside tests.

Resolution: For each vertical resolution required to fully resolve the overlap the technique requires the intersection of an infinite vertical line with all entities in the static shape. Additionally at each step in the resolution the detection step must be performed to ascertain if the resolution is complete. For the example case this would result in 4 full resolution cycles, i.e. 24 intersection tests against an infinite vertical line and at most 4 times the full detection cycle.

Total: 76 intersection tests and 12 point inside tests.

For the same shapes shown in Figure 66, using the no-fit polygon:

Detection: Point inside test of moving shape's ref point against the in place no-fit polygon.

Resolution: A single complete vertical slide to resolve overlap, resulting in the intersection of an infinite vertical line with every entity of no-fit polygon.

Total: 13 intersection tests and 1 point inside test.

6.6. Experiments on Literature Benchmark Problems

This section evaluates the proposed approach by generating new solutions for the previously published literature benchmark problems (see Table 16 and Table 17) using the new no-fit polygon bottom-left-fill implementation and hill-climbing (HC) and tabu (TABU) local search procedures where a solution is represented by a specific sequence/ordering of the shapes. The hill-climbing and tabu local search mechanisms are identically configured to those used in chapter four, as are the sequence operators (1Opt - NOpt).

To provide a numerical evaluation of the shape sequence, we place each shape in the order given by the shape sequence using the bottom-left-fill placement heuristic discussed in chapter four and then we calculate the overall length of the layout and its density. The initial shape sequences are provided by the decreasing length or decreasing area sorted ordering of the input shapes.

Table 16. Length evaluated irregular benchmark problems from the literature

Original Author	Problem Name	Shapes	Rotational Constraints	Sheet Width	Best Length	Time To Best (s)	Best Available Result Reference
Blazewicz, Hawryluk & Walkowiak (1993)	Blasz1	28	0, 180 Absolute	15	26.57	5603	Bennell & Song (2007) [called SHAPES2]
Ratanapan & Dagli (1997)	Dagli	30	90 Incremental	60	57.64	17331	Bennell & Song (2007)
Fujita, Akagi & Kirokawa (1993)	Fu	12	90 Incremental	38	31.57	1192	Bennell & Song (2007)
Jakobs (1996)	Jakobs1	25	90 Incremental	40	11.4	2193	Bennell & Song (2007)
Jakobs (1996)	Jakobs2	25	90 Incremental	70	24.97	4540	Gomes & Oliveira (2006)
Marques, Bispo & Sentieiro (1991)	Marques	24	90 Incremental	104	77.79	10692	Bennell & Song (2007)
Hopper (2000)	Poly1A	15	90 Incremental	40	14.00	12	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly2A	30	90 Incremental	40	28.17	121	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly3A	45	90 Incremental	40	40.33	1515	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly4A	60	90 Incremental	40	54.93	203	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly5A	75	90 Incremental	40	69.37	476	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly2B	30	90 Incremental	40	30.00	180	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly3B	45	90 Incremental	40	40.74	418	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly4B	60	90 Incremental	40	51.73	96	Burke, Hellier, Kendall & Whitwell (2006)
Hopper (2000)	Poly5B	75	90 Incremental	40	57.53	52514	Bennell & Song (2007)
Hopper (2000)	SHAPES	43	90 Incremental	40	59	31	Burke, Hellier, Kendall & Whitwell (2006)
Oliveira & Ferreira (1993)	SHAPES0	43	0 Absolute	40	60	3914	Gomes & Oliveira (2006)
Oliveira & Ferreira (1993)	SHAPES1	43	0, 180 Absolute	40	55	398	Bennell & Song (2007)

Oliveira & Ferreira (1993)	SHIRTS	99	0, 180 Absolute	40	61.33	7033	Bennell & Song (2007)
Oliveira, Gomes & Ferreira (2000)	SWIM	48	0, 180 Absolute	5752	5895.17	15721	Bennell & Song (2007)
Oliveira, Gomes & Ferreira (2000)	TROUSERS	64	0, 180 Absolute	79	241	5988	Bennell & Song (2007)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles1	32	90 Incremental	2000	1377.74	189	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles2	50	90 Incremental	2500	3216.10	1264	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles3	46	45 Incremental	2500	8193.89	1759	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles4	54	90 Incremental	500	2453.12	1131	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles5	50	15 Incremental	4000	3332.70	1317	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles6	69	90 Incremental	5000	3097.86	813	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles7	9	90 Incremental	500	1296.30	680	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles8	18	90 Incremental	1000	1318.70	354	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles9	57	90 Incremental	1500	1290.67	1215	Burke, Hellier, Kendall & Whitwell (2006)
Burke, Hellier, Kendall & Whitwell (2006)	Profiles10	91	0 Absolute	3000	11160	111	Burke, Hellier, Kendall & Whitwell (2006)

Table 17 Density evaluated irregular benchmark problems from the literature

Original Author	Problem Name	Shapes	Rotational Constraints	Sheet Width	Best Density %	Time To Best (s)	Best Available Result Reference
Albano & Sappupo (1980)	Albano	24	90 Incremental	4900	87.88	5460	Bennell & Song (2007)
Blazewicz, Hawryluk & Walkowiak (1993)	Blasz2	20	90 Incremental	15	83.61	2257	Gomes & Oliveira (2006)
Dighe & Jakiela (1996)	Dighe1	16	90 Incremental	100	100	1.4	Gomes & Oliveira (2006), Bennell & Song (2007)
Dighe & Jakiela (1996)	Dighe2	10	90 Incremental	100	100	0.3	Gomes & Oliveira (2006), Bennell & Song (2007)
Bounsaythip & Maouche (1997)	Mao	20	90 Incremental	2550	81.01	667	Gomes & Oliveira (2006)

All of the experiments have been conducted on a 2GHz Intel Pentium 4 processor with 256MB RAM, identical equipment to that used for all previous experiments.

In the experiments of chapter four 100 iterations were allowed for each run. However, as both approaches will generate identical layouts when provided with the same shape input order, allowing only 100 iterations for this new approach would not allow us to determine the difference in the quality of layouts generated or the average quality of the solutions generated. For these experiments each problem is allowed 5 minutes run time which generally matches the duration that the trigonometric approach took to perform 100 iterations or most problems. This allows for a comparison of the speed benefits (average time taken per nest) and the likelihood of finding higher quality layouts, on average, using the no-fit polygon approach when compared with the trigonometric approach of chapter four.

In Table 18 and Table 19 show the best and average layout lengths obtained by the proposed approach for the length and density evaluated literature benchmark problems. We also show two different density measures: the first is a simple straight line density (Density1), while the second density measure, used by (Hopper, 2000), is based on the union of all individual shape bounding rectangles. This allows us to use a nonrectangular final density measurement (Density2).

For each dataset, we perform 10 runs (of 5 minutes each) using each of the four different combinations of initial sorting and local search procedure: i) HC + Area, ii) HC + Length, iii) TABU + Area, and iv) TABU + Length. Additional statistical analysis giving minimum, maximum, average layout length and standard deviation for each of the four combinations of sort and local search procedure separately are presented in Table 20. From Table 18, Table 19 and, it is clear that no sort/search combination dominates across the entire set of benchmarks. This may indicate that the combination of sort and search is somewhat dependent on the input data. For example, the SWIM dataset performs better when using a length sorted initial ordering for both hill-climbing and tabu local searches.

Table 18. Experiments on length evaluated literature benchmark problems

Problem	Hill Climbing								Tabu Search							
	Length Ordered				Area Ordered				Length Ordered				Area Ordered			
	Average Length	Best Result			Average Length	Best Result			Average Length	Best Result			Average Length	Best Result		
		Length	Density1	Density2		Length	Density1	Density2		Length	Density1	Density2		Length	Density1	Density2
Blasz1	27.54	**26.80**	**80.6%**	**82.8%**	27.73	27.20	79.4%	81.2%	27.33	27.10	79.7%	81.0%	27.51	27.00	80.0%	81.3%
Dagli	60.81	**59.94**	**84.6%**	**85.8%**	60.87	60.36	84.0%	85.1%	61.03	60.24	84.2%	84.8%	61.15	60.38	84.0%	84.8%
Fu	32.98	32.89	86.7%	89.1%	32.33	**31.57**	**90.2%**	**92.1%**	32.94	32.70	87.2%	87.7%	32.92	**31.57**	**90.2%**	**92.1%**
Jakobs1	11.90	**11.50**	**85.2%**	88.5%	12.00	12.00	81.7%	86.0%	11.97	11.86	82.6%	88.9%	11.79	**11.50**	**85.2%**	**90.3%**
Jakobs2	26.00	26.00	74.2%	77.5%	26.00	26.00	74.2%	78.4%	26.00	26.00	74.2%	80.8%	25.41	**24.70**	**78.1%**	**82.5%**
Marques	78.80	**78.00**	**88.7%**	89.6%	79.59	79.00	87.6%	89.1%	79.71	78.93	87.6%	**89.9%**	79.60	79.00	87.6%	88.0%
Poly1A	13.67	**13.30**	**77.1%**	**82.8%**	13.81	13.70	74.8%	78.5%	13.78	13.69	74.8%	79.3%	13.97	13.69	74.9%	77.1%
Poly2A	27.53	**27.09**	**75.7%**	77.7%	27.79	27.58	74.3%	75.2%	27.19	**27.09**	**75.7%**	**78.8%**	27.55	27.13	75.6%	77.3%
Poly3A	41.35	**41.07**	**74.9%**	76.5%	41.75	41.53	74.0%	75.7%	41.55	41.25	74.5%	**77.7%**	41.77	41.67	73.8%	75.4%
Poly4A	55.78	55.14	74.4%	76.0%	55.73	55.53	73.8%	75.7%	55.75	**54.60**	**75.1%**	**76.7%**	55.73	55.10	74.4%	75.7%
Poly5A	69.98	69.84	73.4%	74.6%	70.20	69.56	73.7%	74.2%	70.06	69.13	74.1%	75.4%	69.79	**68.84**	**74.4%**	**75.5%**
Poly2B	30.37	30.11	75.1%	77.4%	30.44	**29.63**	**76.3%**	**77.5%**	30.54	30.40	74.4%	76.6%	30.52	30.28	74.7%	76.1%
Poly3B	41.00	40.66	75.0%	**76.5%**	40.97	40.63	75.1%	76.0%	41.20	41.06	74.3%	75.8%	41.02	**40.50**	**75.3%**	76.4%
Poly4B	52.66	52.33	73.9%	74.7%	52.32	51.84	74.6%	75.3%	52.10	51.72	74.8%	75.7%	51.67	**51.18**	**75.6%**	**76.9%**
Poly5B	61.68	61.14	75.1%	76.1%	61.62	61.31	74.9%	**76.4%**	61.61	**60.86**	**75.4%**	75.9%	61.81	61.71	74.4%	75.8%
SHAPES	57.40	**56.00**	**71.2%**	**73.7%**	57.80	57.00	70.0%	71.1%	57.90	57.50	69.4%	70.7%	58.20	**56.00**	**71.2%**	72.5%
SHAPES0	62.00	**60.00**	**66.5%**	**70.2%**	62.40	62.00	64.4%	66.5%	62.30	61.00	65.4%	67.3%	64.22	62.50	63.8%	65.5%
SHAPES1	56.20	**55.00**	**72.5%**	**74.3%**	57.00	57.00	70.0%	71.6%	58.46	58.00	68.8%	71.1%	58.20	58.00	68.8%	70.4%
SHIRTS	63.71	**63.40**	**85.2%**	**86.9%**	64.10	63.98	84.4%	86.3%	64.04	63.93	84.5%	86.5%	64.09	63.85	84.6%	86.2%
SWIM	6464.73	6311.28	70.1%	71.3%	6531.24	6305.94	70.1%	**71.5%**	6416.59	**6270.88**	**70.5%**	71.4%	6566.58	6499.91	68.1%	70.0%
TROUSERS	248.56	245.95	88.7%	89.5%	247.75	246.80	88.4%	89.4%	246.60	**245.28**	**88.9%**	**89.8%**	247.98	247.17	88.2%	89.1%
Profiles1	1389.14	1380.10	81.6%	83.0%	1391.40	1386.00	81.3%	82.5%	1412.88	1404.80	80.2%	81.4%	1394.54	**1359.90**	**82.8%**	**84.5%**
Profiles2	3262.10	3216.06	50.0%	50.8%	3230.98	**3194.19**	**50.3%**	**51.5%**	3264.40	3252.70	49.4%	50.3%	3267.78	3223.30	49.8%	50.8%
Profiles3	8073.88	**7881.13**	**52.9%**	**54.0%**	8230.33	8189.66	50.9%	51.7%	8074.34	7999.74	52.1%	53.0%	8177.60	8045.84	51.8%	53.0%
Profiles4	2476.46	2452.42	75.2%	75.7%	2475.58	2464.35	74.8%	75.4%	2466.55	**2425.26**	**76.0%**	**76.5%**	2486.49	2482.72	74.2%	74.8%
Profiles5	3394.32	3367.88	69.4%	70.6%	3401.17	**3351.94**	**69.8%**	**70.8%**	3385.32	3364.35	69.5%	70.5%	3399.48	3384.90	69.1%	70.4%
Profiles6	3134.01	**3121.36**	**75.0%**	**78.6%**	3172.97	3156.02	74.2%	75.6%	3161.51	3146.56	74.4%	76.8%	3176.52	3161.22	74.1%	76.3%
Profiles7	1305.78	**1292.30**	**77.4%**	**79.9%**	1307.44	**1292.30**	**77.4%**	**79.9%**	1316.32	1296.30	77.1%	77.7%	1313.44	1309.10	76.4%	79.6%
Profiles8	1303.17	1268.98	78.8%	79.8%	1283.19	1268.98	78.8%	79.8%	1308.61	1293.88	77.3%	77.5%	1285.44	**1263.11**	**79.1%**	**82.1%**
Profiles9	1314.73	**1278.21**	**52.7%**	**54.2%**	1298.42	1290.00	52.2%	53.2%	1308.93	1298.62	51.9%	52.9%	1303.43	1292.63	52.1%	53.3%
Profiles10	11373.40	**11219.60**	**65.8%**	**66.6%**	11515.03	11403.99	64.8%	65.4%	11392.36	11302.50	65.3%	66.0%	11653.40	11585.81	63.7%	64.7%

Table 19. Experiments on density evaluated literature benchmark problems

	Hill Climbing								Tabu Search							
	Length Ordered				Area Ordered				Length Ordered				Area Ordered			
Problem	Average	Best Result			Average	Best Result			Average	Best Result			Average	Best Result		
	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2	Length	Length	Density1	Density2
Albano	10203.84	**9980.86**	**87.2%**	**88.3%**	10196.87	10169.47	85.6%	86.4%	10130.41	10010.73	87.0%	87.4%	10174.15	10037.38	86.7%	87.5%
Blasz2	24.94	24.80	75.9%	79.9%	24.92	24.90	75.6%	79.9%	24.96	**24.80**	**75.9%**	**80.4%**	25.06	24.90	75.6%	79.0%
Dighe1	1257.96	1239.60	80.7%	81.1%	1260.38	1250.00	80.0%	81.9%	1289.62	1270.00	78.7%	80.1%	1257.72	**1210.00**	**82.6%**	**83.8%**
Dighe2	1224.66	1215.70	82.3%	83.0%	1205.66	**1180.00**	**84.7%**	**86.5%**	1221.12	1190.00	81.1%	84.7%	1229.32	1226.60	81.5%	83.6%
Mao	1857.70	1847.20	79.8%	83.1%	1841.52	1821.70	80.9%	**83.9%**	1853.90	1821.20	80.9%	83.1%	1838.78	**1811.50**	**81.4%**	83.6%

Table 20. Min, max, average length and standard deviation statistics for the proposed method on the literature benchmark problems

	Hill Climbing								Tabu Search							
	Length Ordered				Area Ordered				Length Ordered				Area Ordered			
Problem	Min Length	Max Length	Average Length	Std Dev	Min Length	Max Length	Average Length	Std Dev	Min Length	Max Length	Average Length	Std Dev	Min Length	Max Length	Average Length	Std Dev
Albano	9980.86	10380.57	10203.84	143.52	10169.47	10276.34	10196.87	44.98	10010.73	10197.14	10130.41	74.87	10037.38	10258.82	10174.15	92.46
Blasz1	26.80	27.90	27.54	0.44	27.20	27.90	27.73	0.30	27.10	27.50	27.33	0.18	27.00	27.80	27.51	0.30
Blasz2	24.80	25.00	24.94	0.09	24.90	25.00	24.92	0.04	24.80	25.10	24.96	0.11	24.90	25.20	25.06	0.13
Dagli	59.94	61.16	60.81	0.49	60.36	61.27	60.87	0.36	60.24	61.49	61.03	0.55	60.38	61.73	61.15	0.54
Dighe1	1239.60	1280.00	1257.96	16.06	1250.00	1270.00	1260.38	9.11	1270.00	1300.00	1289.62	11.88	1210.00	1280.80	1257.72	29.26
Dighe2	1215.70	1226.90	1224.66	5.01	1180.00	1226.90	1205.66	19.73	1190.00	1230.00	1221.12	17.45	1226.60	1230.00	1229.32	1.52
Fu	32.89	33.00	32.98	0.05	31.57	33.00	32.33	0.70	32.70	33.00	32.94	0.13	31.57	33.80	32.92	0.81
Jakobs1	11.50	12.00	11.90	0.22	12.00	12.00	12.00	0.00	11.86	12.00	11.97	0.06	11.50	12.00	11.79	0.23
Jakobs2	26.00	26.00	26.00	0.00	26.00	26.00	26.00	0.00	26.00	26.00	26.00	0.00	24.70	26.00	25.41	0.63
Mao	1847.20	1865.80	1857.70	8.14	1821.70	1869.00	1841.52	19.23	1821.20	1875.00	1853.90	20.51	1811.50	1860.80	1838.78	18.99
Marques	78.00	79.00	78.80	0.45	79.00	80.00	79.59	0.54	78.93	80.00	79.71	0.47	79.00	80.00	79.60	0.55
Poly1A	13.30	13.83	13.67	0.21	13.70	13.86	13.81	0.07	13.69	13.83	13.78	0.05	13.69	14.10	13.97	0.17
Poly2A	27.09	27.77	27.53	0.26	27.58	27.98	27.79	0.15	27.09	27.32	27.19	0.08	27.13	27.82	27.55	0.29
Poly3A	41.07	41.68	41.35	0.22	41.53	41.96	41.75	0.19	41.25	41.81	41.55	0.26	41.67	41.94	41.77	0.13
Poly4A	55.14	56.24	55.78	0.50	55.53	55.90	55.73	0.18	54.60	56.86	55.75	0.80	55.10	56.22	55.73	0.42
Poly5A	69.84	70.11	69.98	0.11	69.56	70.53	70.20	0.40	69.13	70.57	70.06	0.63	68.84	70.36	69.79	0.69
Poly2B	30.11	30.78	30.37	0.28	29.63	30.87	30.44	0.46	30.40	30.69	30.54	0.11	30.28	30.76	30.52	0.21
Poly3B	40.66	41.26	41.00	0.24	40.63	41.26	40.97	0.25	41.06	41.30	41.20	0.09	40.50	41.31	41.02	0.32
Poly4B	52.33	52.88	52.66	0.24	51.84	52.56	52.32	0.29	51.72	52.42	52.10	0.28	51.18	51.98	51.67	0.33
Poly5B	61.14	62.14	61.68	0.42	61.31	62.01	61.62	0.29	60.86	61.98	61.61	0.45	61.71	61.98	61.81	0.12
SHAPES	56.00	59.00	57.40	1.14	57.00	58.00	57.80	0.45	57.50	58.00	57.90	0.22	56.00	59.00	58.20	1.30
SHAPES0	60.00	63.00	62.00	1.22	62.00	63.00	62.40	0.55	61.00	64.00	62.30	1.14	62.50	65.00	64.22	1.04
SHAPES1	55.00	57.00	56.20	0.76	57.00	57.00	57.00	0.00	58.00	59.00	58.46	0.51	58.00	59.00	58.20	0.45
SHIRTS	63.40	64.09	63.71	0.30	63.98	64.20	64.10	0.09	63.93	64.11	64.04	0.08	63.85	64.41	64.09	0.26
SWIM	6311.28	6590.38	6464.73	116.16	6305.94	6608.86	6531.24	127.43	6270.88	6535.02	6416.59	114.10	6499.91	6614.67	6566.58	45.30
TROUSERS	245.95	249.48	248.56	1.47	246.80	248.30	247.75	0.59	245.28	247.56	246.60	0.93	247.17	248.82	247.98	0.68
Profiles1	1380.10	1392.10	1389.14	5.12	1386.00	1397.50	1391.40	4.14	1404.80	1417.40	1412.88	5.61	1359.90	1414.20	1394.54	22.72
Profiles2	3216.06	3302.21	3262.10	41.59	3194.19	3258.04	3230.98	29.64	3252.70	3280.60	3264.40	14.48	3223.30	3301.74	3267.78	29.87
Profiles3	7881.13	8155.11	8073.88	111.70	8189.66	8258.72	8230.33	25.61	7999.74	8133.98	8074.34	57.91	8045.84	8227.48	8177.60	78.39
Profiles4	2452.42	2486.76	2476.46	14.04	2464.35	2482.38	2475.58	7.48	2425.26	2488.26	2466.55	23.54	2482.72	2489.30	2486.49	2.43
Profiles5	3367.88	3415.66	3394.32	19.81	3351.94	3459.24	3401.17	38.43	3364.35	3398.02	3385.32	14.67	3384.90	3413.20	3399.48	11.45
Profiles6	3121.36	3149.66	3134.01	12.79	3156.02	3183.51	3172.97	12.41	3146.56	3171.17	3161.51	12.04	3161.22	3184.02	3176.52	9.29
Profiles7	1292.30	1317.60	1305.78	12.79	1292.30	1317.60	1307.44	9.23	1296.30	1325.40	1316.32	13.23	1309.10	1322.30	1313.44	6.17
Profiles8	1268.98	1321.78	1303.17	20.09	1268.98	1293.60	1283.19	11.24	1293.88	1322.10	1308.61	10.03	1263.11	1297.11	1285.44	14.09
Profiles9	1278.21	1341.88	1314.73	23.55	1290.00	1305.60	1298.42	7.18	1298.62	1324.45	1308.93	11.56	1292.63	1316.22	1303.43	8.97
Profiles10	11219.60	11484.88	11373.40	96.37	11403.99	11601.51	11515.03	94.89	11302.50	11477.91	11392.36	65.73	11585.81	11699.05	11653.40	54.82

Table 21 provides a summary of the results and a comparison to both the best-known solutions and the solutions obtained in chapter three. The best result is highlighted in bold type and results are underlined where the proposed approach achieved less than 1% worse solutions but did so in a faster time.

Table 21. A comparison of the best results of the proposed no-fit polygon packing approach to that of Burke et al. (2006) and the best results from the available literature

Problem	Best Literature	Best Literature Time (s)	Burke, Hellier, Kendall and Whitwell (2006)					Proposed No-Fit Polygon Packing Best					% Improvement over	
			Length	Density1	Density2	Time / Nest (s)	Time To Best (s)	Length	Density1	Density2	Time / Nest (s)	Time To Best (s)	Best Literature	Burke et al. (2006)
Blasz1	**26.57**	5603	27.80	77.7%	81.0%	0.32	21	26.80	80.6%	82.8%	0.09	**281**	-0.87%	**3.60%**
Dagli	**57.64**	17331	60.57	83.7%	89.0%	2.04	189	59.94	84.6%	85.8%	0.42	252	-3.99%	**1.05%**
Fu	31.57	1192	32.80	86.9%	90.8%	0.24	21	**31.57**	90.2%	92.1%	0.17	**139**	**0.00%**	**3.75%**
Jakobs1	**11.40**	2193	11.86	82.6%	92.6%	0.74	43	11.50	85.2%	90.3%	0.19	**29**	-0.88%	**3.02%**
Jakobs2	**24.97**	4540	25.80	74.8%	83.3%	2.13	81	**24.70**	78.1%	82.5%	0.64	**51**	**1.08%**	**4.26%**
Marques	**77.97**	10692	80.00	86.5%	89.3%	0.25	5	78.00	88.7%	89.6%	0.07	**21**	-0.04%	**2.50%**
Poly1A	14.00	12	14.00	73.2%	78.2%	0.36	12	**13.30**	77.1%	82.8%	0.11	**254**	**5.03%**	**5.03%**
Poly2A	28.17	121	28.17	72.8%	77.5%	1.24	121	**27.09**	75.7%	78.8%	0.50	**239**	**3.84%**	**3.84%**
Poly3A	**40.33**	1515	41.65	73.8%	75.5%	2.01	210	41.07	74.9%	76.5%	1.00	159	-1.83%	**1.40%**
Poly4A	54.93	**203**	54.93	74.6%	75.9%	2.43	203	**54.60**	75.1%	76.7%	1.71	**224**	**0.60%**	**0.59%**
Poly5A	69.37	476	69.37	73.9%	75.7%	5.04	476	**68.84**	74.4%	75.5%	1.76	**300**	**0.76%**	**0.76%**
Poly2B	30.00	180	30.00	75.4%	77.5%	2.50	180	**29.63**	76.3%	77.5%	0.90	**189**	**1.23%**	**1.24%**
Poly3B	40.74	418	40.74	74.9%	77.1%	4.26	418	**40.50**	75.3%	76.4%	2.22	**114**	**0.58%**	**0.58%**
Poly4B	51.73	96	51.73	74.8%	77.4%	8.24	96	**51.18**	75.6%	76.9%	6.00	**176**	**1.07%**	**1.07%**
Poly5B	**57.53**	52514	60.54	75.8%	77.2%	14.70	677	60.86	75.4%	75.9%	12.62	299	-5.78%	-0.52%
SHAPES	59.00	31	59.00	67.6%	69.1%	0.60	31	**56.00**	71.2%	73.7%	0.27	**226**	**5.08%**	**5.09%**
SHAPES0	**60.00**	3914	66.00	60.5%	62.6%	0.93	21	**60.00**	66.5%	70.2%	0.14	**274**	**0.00%**	**9.09%**
SHAPES1	**55.00**	398	60.00	66.5%	68.9%	0.82	2	**55.00**	72.5%	74.3%	0.22	**239**	**0.00%**	**8.33%**
SHIRTS	**61.33**	7033	63.80	84.6%	87.3%	4.99	58	63.40	85.2%	86.9%	0.77	194	-3.38%	**0.62%**
SWIM	**5895.17**	15721	6462.40	68.4%	71.6%	12.39	607	6270.88	70.5%	71.4%	1.24	141	-6.37%	**2.96%**
TROUSERS	**241.00**	5988	246.57	88.5%	90.1%	7.89	756	245.28	88.9%	89.8%	1.02	253	-1.77%	**0.52%**
Albano	**87.9%**	5460	10292.90	84.6%	86.5%	1.18	93	9980.86	87.23%	88.25%	0.25	**299**	-0.75%	**3.03%**
Blasz2	**83.6%**	2257	25.28	74.5%	79.9%	0.16	11	24.80	75.94%	80.41%	0.07	14	-10.10%	**1.88%**
Dighe1	**100.0%**	1.2	1292.30	77.4%	78.9%	0.22	9	1210.00	82.65%	83.84%	0.15	3	-21.00%	**6.37%**
Dighe2	**100.0%**	0.3	1260.00	79.4%	84.3%	0.10	7	1180.00	84.75%	86.50%	0.70	148	-18.00%	**6.35%**
Mao	**81.0%**	667	1854.30	79.5%	82.9%	0.38	30	1821.70	80.91%	83.86%	0.13	**152**	-0.12%	**1.76%**
Profiles1	1377.74	189	1377.74	82.0%	85.2%	0.83	189	**1359.90**	82.8%	85.4%	0.10	**15**	**1.29%**	**1.29%**
Profiles2	3216.1	1264	3216.10	50.0%	51.3%	31.15	264	**3194.19**	50.3%	51.5%	2.88	**295**	**0.68%**	**0.68%**
Profiles3	8193.89	1759	8193.89	50.9%	52.6%	7.68	1759	**7881.13**	52.9%	54.0%	0.80	**283**	**3.82%**	**3.82%**
Profiles4	2453.12	1131	2453.12	75.1%	75.7%	1.04	1131	**2425.26**	76.0%	76.5%	0.16	**256**	**1.14%**	**1.14%**
Profiles5	**3332.7**	1317	3332.70	70.2%	73.6%	65.92	1317	3351.94	69.8%	70.8%	7.39	**300**	-0.58%	-0.58%
Profiles6	**3097.86**	813	3097.86	75.6%	77.8%	2.44	813	3121.36	75.0%	78.6%	0.67	**171**	-0.76%	-0.76%
Profiles7	1296.3	680	1296.30	77.1%	80.2%	0.06	680	**1292.30**	77.4%	79.9%	0.02	**211**	**0.31%**	**0.31%**
Profiles8	1318.7	354	1318.70	75.8%	77.2%	0.55	354	**1263.11**	79.1%	82.1%	0.08	**279**	**4.22%**	**4.22%**
Profiles9	1290.67	1215	1290.67	53.1%	54.9%	8.8	1215	**1278.21**	52.7%	54.2%	0.39	**98**	**0.97%**	**0.97%**
Profiles10	**11160**	**111**	11160.10	66.2%	66.8%	1.89	111	11219.60	65.8%	66.6%	0.42	247	-0.53%	-0.53%

Comparing the "Time / Nest" columns in Table 21, it can be seen that the use of the no-fit polygon has considerably reduced the amount of time required to produce layouts compared to the previous trigonometric approach presented in chapter three. This difference is perhaps most evident in the problems "SWIM" and "Profiles9" which both involve shapes consisting of numerous small edges. For "SWIM", the no-fit polygon implementation is approximately 10 times quicker and, for "Profiles9", layouts can be produced about 22 times faster. With some other problems, such as "Fu" and the "Poly" problems, there are only small improvements in layout generation time. On closer examination, these datasets contain shapes with relatively few edges (4, 5 or 6) and, therefore, the benefit of resolving intersections in one step via the NFP does not provide as great an advantage over the edge by edge intersection resolution using trigonometry.

In comparison to the best solutions from the literature on the 36 benchmark problems, the proposed approach obtains 16 new outright best solutions, 3 equal best solutions in a significantly faster time, and 7 solutions that are within 1% of the best literature solution but found in a significantly faster time. These three groups are reproduced in Table 22.

Table 22. Summary of problems for which the proposed approach yields better solution quality and/or time than the best known solutions from the literature.

Group	**Problem**	**Best Literature**		**Proposed Approach**	
		Evaluation	**Time Taken (s)**	**Evaluation**	**Time Taken (s)**
(1) **Better Evaluation**	Jakobs2	24.97	4540	**24.70**	**51**
	Poly1A	14.00	**12**	**13.30**	254
	Poly2A	28.17	**121**	**27.09**	239
	Poly4A	54.93	**203**	**54.60**	224
	Poly5A	69.37	476	**68.84**	**300**
	Poly2B	30.00	**180**	**29.63**	189
	Poly3B	40.74	418	**40.50**	**114**
	Poly4B	51.73	**96**	**51.18**	176
	SHAPES	59.00	**31**	**56.00**	226
	Profiles1	1377.74	189	**1359.90**	**15**
	Profiles2	3216.1	1264	**3194.19**	**295**
	Profiles3	8193.89	1759	**7881.13**	**283**
	Profiles4	2453.12	1131	**2425.26**	**256**
	Profiles7	1296.3	680	**1292.30**	**211**
	Profiles8	1318.7	354	**1263.11**	**279**
	Profiles9	1290.67	1215	**1278.21**	**98**
(2) **Equal Evaluation & Better Time**	Fu	**31.57**	1192	**31.57**	**139**
	SHAPES0	**60.00**	3914	**60.00**	**274**
	SHAPES1	**55.00**	398	**55.00**	**239**
(3) **Slightly Worse Evaluation (< 1%) & Better Time**	Blasz1	**26.57**	5603	26.80	**281**
	Jakobs1	**11.40**	2193	11.50	**29**
	Marques	**77.97**	10692	78.00	**21**
	Profiles5	**3332.7**	1317	3351.94	**300**
	Profiles6	**3097.86**	813	3121.36	**171**
	Albano	**87.9%**	5460	87.23%	**299**
	Mao	**81.0%**	667	80.91%	**152**

In group 2, the proposed packing approach achieves equal solutions to the best reported for "Fu", "Shapes0" and "Shapes1". However, with the presented approach, the solutions are found 8, 14 and 1.5 times quicker respectively. The final group reports solutions in which the proposed approach found a slightly worse solution (within 1%) than the best solution reported in the scientific literature but in a significantly faster time. The most extreme example of this is demonstrated by the "Marques" dataset whereby the best literature solution was found after around 3 hours of computation whilst an almost equivalent solution was found by the proposed approach well within the 5 minutes allowed (over 500 times faster). These results have been included to demonstrate that very good solutions of comparative quality can be achieved within timeframes that would be acceptable for use in industry.

6.7. Summary

Until the generation of this technique no-fit polygon algorithms only operated on straight edge representations. If arcs were present the polygons had to be represented by straight line segments in order for the no-fit polygon to be used. This led to computational inefficiencies, especially if a low resolution was used, so that the arcs could be more accurately represented.

The main contribution of this chapter is the presentation of a no-fit polygon algorithm that is able to cope with arcs. Its effectiveness has been shown by using an identical layout technique to that outlined in chapter four utilising the generated no-fit polygons.

Due to the efficiency of the overlap resolution, compared to the entity by entity overlap resolution technique utilised in the work presented in chapter four as demonstrated in section 6.5, the new bottom-left fill heuristic algorithm has been able to more extensively explore the solution space resulting in higher quality solutions being discovered. The combination of line arc no-fit polygons and the layout algorithm has been able to produce better results on 16 (of 36) benchmark problems at the time of publication. On another three instances the combined technique finds results which are equal to the best known solution. Perhaps of more interest is the fact that the algorithm is much faster than other approaches in the scientific literature.

Chapter Seven

7. Irregular Packing with a non-greedy bottom-left fill approach

7.1. Introduction

In chapter four a new bottom left fill nesting heuristic was introduced. This heuristic was able to produce some of the best results seen in the scientific literature at the point of its publication. However the placement heuristic, whilst entirely accurate in the *y* axis, required a discrete *x* value to be used whilst searching for a feasible position to place a part. It possible therefore that more efficient placement positions could have been overlooked as they could have fallen between the discrete *x* values tested.

Chapter five presents an extension of Mahadevan's sliding no-fit polygon generation technique that allows for the reliable generation of no-fit polygons. Furthermore this work extends no-fit polygon generation to allow for those cases exhibiting all previously known degenerate cases and to include generation for shapes incorporating circular arc segments within their boundaries.

In chapter six the placement heuristic devised in chapter four utilised these no-fit polygons to significantly improve the performance of the placement heuristic, once again this allowed the generation of some new best results on the literature benchmark problems. However this implementation still suffered from the inaccuracy inherent in the discrete *x* values being tested during shape placement.

In 2002 Gomes and Oliveira introduced the TOPOS placement heuristic which utilises the no-fit polygons of the shapes to generate accurate placements for shapes during the nesting process (Gomes & Oliveira, 2002). By correctly positioning and intersecting the no-fit polygons of the shape being placed with those of the already placed shapes, along with the inner fit polygon of the shape being placed and the sheet, it is possible to determine all feasible placement positions in which a shape may be placed without overlap with placed shapes or the containing sheet. The TOPOS placement mechanism has also been very effectively by Bennell and Song in (Bennell & Song, 2010) who used an improved implementation of the TOPOS technique which allowed for the filling of holes in a partial layout.

Unlike the new bottom-left fill heuristic used in the earlier chapters the TOPOS placement technique does not suffer from any inaccuracy in the *x* or *y* axis. Importantly for the work presented in this chapter the TOPOS approach also offers a set of feasible placement points for a shape at each placement. In order to produce a bottom-left fill nesting approach it would only require that, of all the valid placement points, the lowest left most position of the set of feasible positions is consistently chosen as the placement location.

The set of feasible placement positions produced by the TOPOS approach for each shape placement is utilised in the work presented in this chapter as part of an exploration into the usefulness of a less greedy approach to the bottom-left fill placement heuristic.

7.2. Motivation

The exploration of a non-greedy placement method has been motivated by the significant amount of feedback Aptia Solutions Ltd. receives regarding layouts produced by its greedy bottom left fill nesting algorithms (see chapter eight for a company case study).

Through Aptia's "MyNesting" application, which has over 5,700 registered users, the user base is able to submit nests for review by the development team. Following a review of a sample of the nests that users regarded as unsatisfactory it was clear that excessively greedy placement was one cause of user dissatisfaction.

Figure 67 shows a simple example of greedy placement leading to inefficient nesting. In the example four very similar shapes are to be placed onto a sheet, these parts have the rotational constraints of 0 and 180 degrees. The bottom left fill technique used in chapters four and six of this thesis has been used to place the shapes. This technique has worked well for the first two parts placed. However when the third part is placed in the lowest left most position found by the algorithm the placement results in a lowest left most placement for the fourth part that is not particularly efficient.

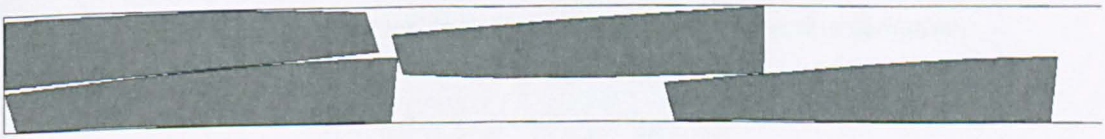


Figure 67. Greedy Placement (12643 units in length)

Figure 68 shows an example of non-greedy placement of the same parts which allows for a significant reduction in the length of the layout. By allowing the third shape to be placed slightly further to the right, in its other orientation, the fourth shape can now fit beneath it.

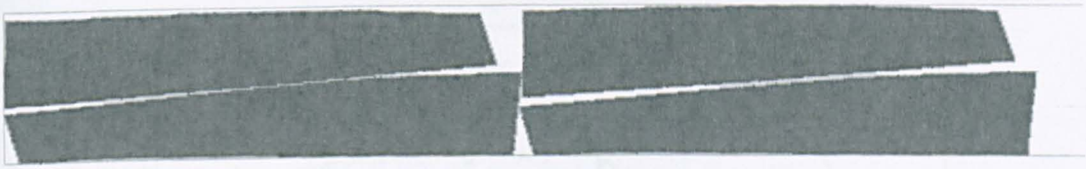


Figure 68. Non-greedy placement (9615 units in length)

The following chapter explores, utilising the TOPOS placement technique, the effectiveness of a simple non-greedy nesting placement technique across a range of literature problems.

The aim of the introduction of this technique and its testing on the literature benchmark problems is to determine if a non-greedy placement strategy is likely to be useful across a broad range of the potential problems an industrial automatic nesting application may encounter.

The results of experiments to explore the non-greedy control variables are presented and conclusions on the effectiveness of the technique are drawn. Several of the results generated using this non-greedy TOPOS technique improve upon results generated in the earlier chapters of this thesis.

7.3. TOPOS Placement

The implementation of TOPOS used in this chapter is similar to that utilised in the work presented in (Bennell & Song, 2008) in that it allows for the filling of holes between the placed shapes. However the implementation of TOPOS used in this chapter does not join up placed shapes and generate new no-fit polygons during the layout process, as in the original 2002 implementation of Gomes & Oliveira, but uses pre-generated no-fit polygons placed correctly, i.e. with respect to the non-sliding shape that was used in its generation, at each step of the placement sequence. This implementation therefore relies upon fast and accurate intersection and point inside detection in order to determine candidate placements.

In order to explain the technique adopted to investigate the effectiveness of using a less greedy placement method it is useful to present a brief example of how the implementation the TOPOS placement technique is able to generate a set of the feasible placement points. The follow figures present a step by step visualisation of the TOPOS process of placing a single part, with 90 degree rotational constraints, onto a layout already containing two placed parts.

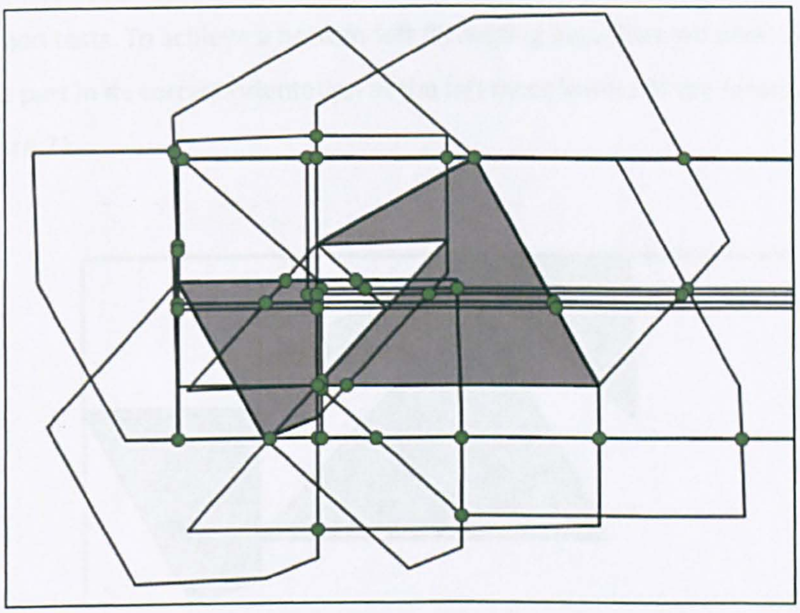


Figure 69. Placed parts, no-fit polygons and inner-fit polygons

Figure 69 shows the placed shapes' no-fit polygons, inner fit polygons and possible placements. The possible placements are determined by using the intersection points between the in position no-fit polygons and some of the free vertices of the no-fit polygons. Potential placement points are then eliminated by testing the points using the point inside or winding number method, discussed in chapter three. If a point is contained inside a no-fit polygon it is removed from the list of potential placements likewise if the point is removed if it is outside of the sheet inner-fit polygon for the candidate shape's orientation.

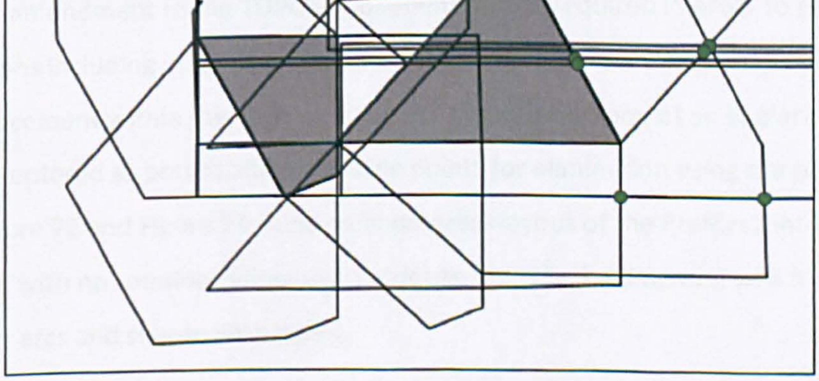


Figure 70. Feasible placement points for TOPOS placement

Figure 70 shows the set of feasible placement points following elimination by no-fit and inner-fit polygon tests. To achieve a bottom-left fill nesting algorithm we need only to place the candidate part in its correct orientation at the left most lowest of the feasible points, as shown in Figure 71.

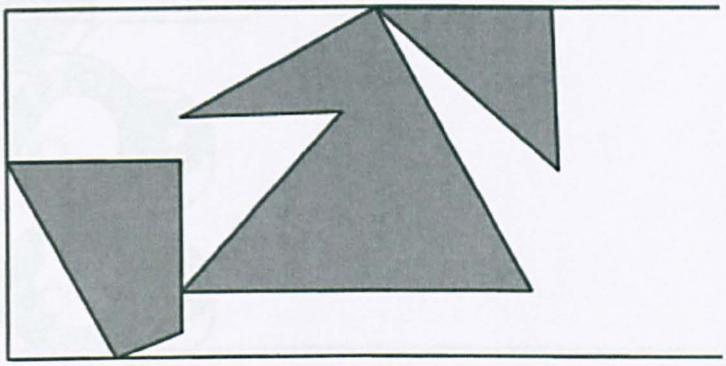


Figure 71. Placed candidate shape in left most position using TOPOS method

Chapter 4 introduced new benchmark problems inspired by the metal profiling industry which the industrial partner of the CASE / TCS projects, Esprit Automation Ltd., is aligned to. These problems include features unique to the literature, namely holes within shapes and circular arcs in the boundary definitions of the shapes. The TOPOS based approach used in this chapter can cope with these features by using intersection routines capable of accurately detecting intersection between circular arc segments and line segments and combinations thereof in order to produce candidate placements. Additionally the point inside or winding number routines used in this chapter, as with the previous chapters, can determine if a point is inside a shape including holes and circular arcs in its boundaries.

One further amendment to the TOPOS implementation is required in order to cope with no-fit polygons including inner paths, where a shape fits within another shape's holes. To allow for placement within this regions all points of any inner loop of an in place no-fit polygon are entered as potentially admissible points for elimination using the point inside method. Figure 72 and Figure 73 show an in progress layout of the Profiles2 introduced in chapter four with no rotations allowed (in order to simplify the diagram) which includes both circular arcs and shapes with holes.

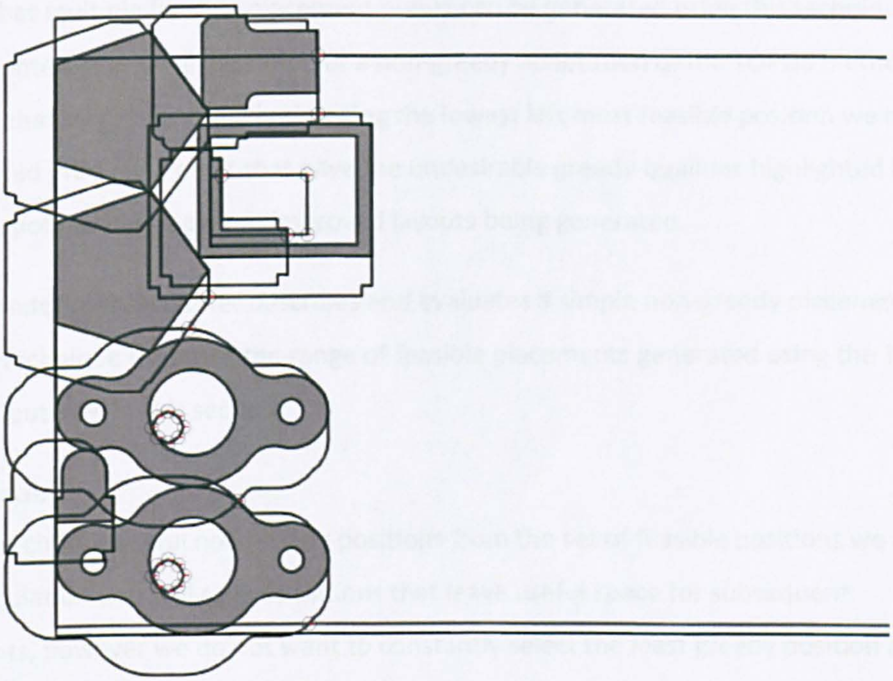


Figure 72. TOPOS placement generation with circular arcs and holes

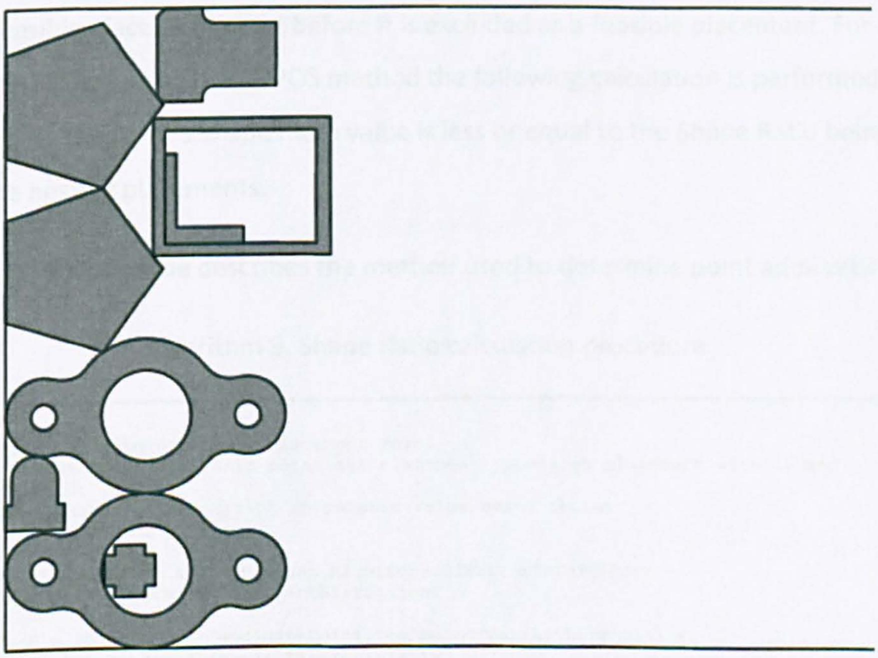


Figure 73. Placement of part into circular hole using TOPOS technique

Using the previously described implementation of the TOPOS technique it is possible to generate multiple feasible placement positions for layouts containing holes both between and within placed shapes, including shapes that have circular arcs in their boundary definitions.

The fact that multiple feasible placement points can be generated using this technique is of particular interest in the exploration of a non-greedy adaptation of the TOPOS method. It is proposed that by not continually choosing the lowest left most feasible position we may be able to avoid producing nests that have the undesirable greedy qualities highlighted in Figure 67, potentially leading to improved layouts being generated.

The remainder of this chapter describes and evaluates a simple non-greedy placement selection technique that uses the range of feasible placements generated using the TOPOS approach outlined in this section.

7.4. Shape Ratio

In order to choose useful non-greedy positions from the set of feasible positions we require some calculation that will select positions that leave useful space for subsequent placements, however we do not want to constantly select the least greedy position and therefore generate very poor layouts, to this end the Shape Ratio calculation was devised.

The Shape Ratio controls the how far a candidate non-greedy placement may be from the left most feasible placement point before it is excluded as a feasible placement. For each feasible point found using the TOPOS method the following calculation is performed to determine that the point's Shape Ratio value is less or equal to the Shape Ratio being used to guide the nesting placements.

The following pseudocode describes the method used to determine point admissibility:

Algorithm 9. Shape Ratio calculation procedure

```
Input
CandidatePoint = valid potential placement point
BestPoint = most efficient valid potential placement (produces placement with least
additional layout length)
AdmissibleShapeRatio = the current ShapeRatio value being tested

Begin
        double xDiff = CandidatePoint.RightBoundOfPartAtThisPoition -
        BestPoint.RightBoundOfPartAtThisPoition;

        double yDiff = abs(CandidatePoint.TopBoundOfPartAtThisPosition -
        BestPoint.TopBoundOfPartAtThisPosition);

        double shapeRatio = xDiff/yDiff;

        bool bValid = false;
        if(shapeRatio <= AdmissibleShapeRatio)
                bValid = true;

        return bValid;
End
```

This formulation means that points that would produce placements that are very non-greedy will be assigned a higher Shape Ratio value. The maximum admissible Shape Ratio value explored in the work presented is 1.0 as this describes points that will produce placements are at most 45 degrees away from the left most feasible placement point, with respect to the right side of the sheet.

Figure 74 and Figure 75 show an in progress non-greedy placement sequence, being produced with an admissible Shape Ratio of 0.7, the Shape Ratio values for the various valid placement points that the TOPOS placement technique has identified are shown as is the implied Shape Ratio = 1.0 boundary.

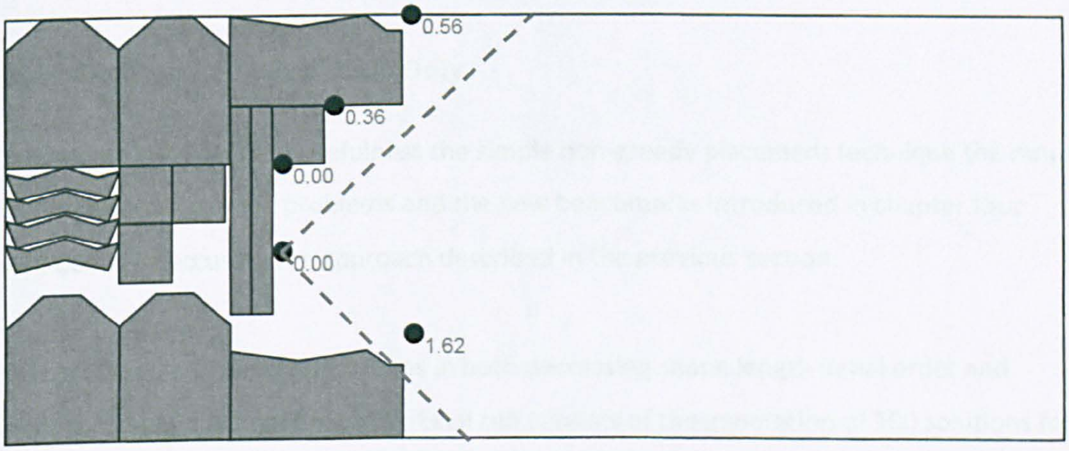


Figure 74. Shape Ratio values for potential placement points and the 1.0 boundary

The shape about to be placed in Figure 74 is shown placed in the top right position in Figure 75. The placement of another copy of the same shape is being considered in Figure 75. In this example no shape rotations are being considered.

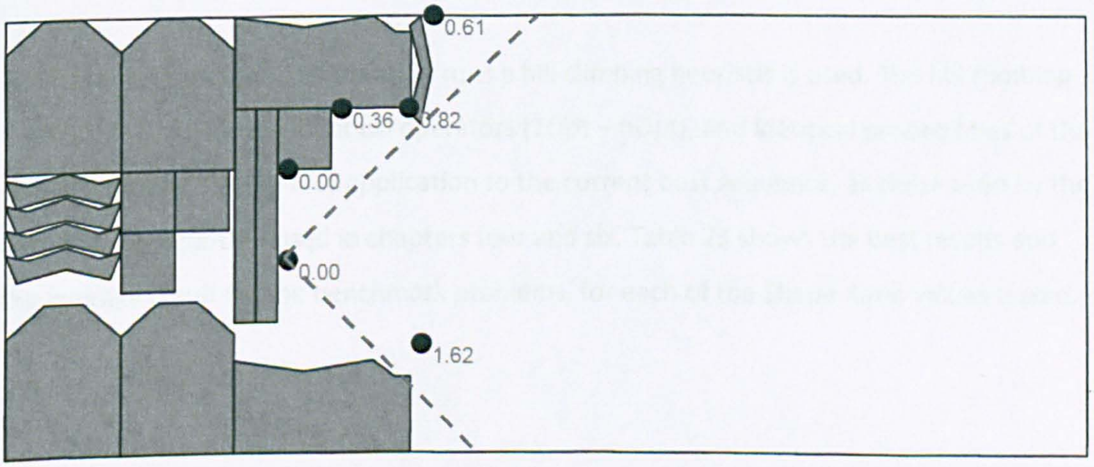


Figure 75. Shape Ratio values and 1.0 boundary for the next shape placement

Of the placements points deemed admissible, i.e. those with a Shape Ratio less than or equal to the required value, the admissible point that has the greatest difference in its *y* value from the left most feasible point is selected as the placement point. This is designed to encourage distribution to both sides of the sheet over the course of the nesting leaving useful spaces centrally in the nest for later placements to utilise.

The results of experiments using Shape Ratio alone to select placement positions, on the benchmark problems, are presented in the next section.

7.5. Experiments with Shape Ratio Only

To evaluate the potential usefulness the simple non-greedy placement technique the range of literature benchmark problems and the new benchmarks introduced in chapter four have been tested using the approach described in the previous section.

Each problem has been run 10 times in both decreasing shape length initial order and decreasing shape area initial order. Each run consists of the generation of 100 solutions for the values of Shape Ratio from value 0 (greedy) to value 1 in 0.1 increments i.e. (0, 0.1, 0.2 ... 0.9, 1).

All experiments in this chapter were performed on an Intel i5 CPU at 2.67 Ghz with 4GB of RAM. The testing application and TOPOS implementation was developed in C# on the .NET 4.0 framework and utilised only a single core of the multicore CPU.

For the duration of the 100 iteration runs a hill climbing heuristic is used. The hill climbing heuristic is run with the identical operators (1Opt – nOpt), and identical probabilities of the operators being selected for application to the current best sequence, as those used by the hill climbing algorithm used in chapters four and six. Table 23 shows the best results and the average result for the benchmark problems, for each of the Shape Ratio values tested.

Table 23. Experimental Results for Shape Ratio values 0 to 1

			Averages										
Problem	**Best Result**	**Best Result SR**	**SR=0.0**	**SR=0.1**	**SR=0.2**	**SR=0.3**	**SR=0.4**	**SR=0.5**	**SR=0.6**	**SR=0.7**	**SR=0.8**	**SR=0.9**	**SR=1**
albano (A)	10355.86	0.1	10733.12	**10693.93**	10745.42	10729.70	10843.16	10886.93	10858.12	10989.06	11024.17	11104.05	11285.98
albano (L)	10360.59	0.1	10754.26	**10662.30**	10683.67	10752.58	10809.30	10935.18	10880.78	10942.00	10899.39	10998.38	11147.15
blaz (A)	28.08	0.3	**29.07**	29.33	29.31	29.19	29.43	29.65	29.78	30.21	30.35	31.13	30.82
blaz (L)	28.08	0.1	**28.63**	28.75	28.91	29.11	29.08	29.57	30.12	30.48	30.50	31.28	31.16
dagli (A)	62.62	0.1	64.75	**64.31**	64.94	65.24	68.08	70.24	71.06	71.78	72.57	75.14	75.67
dagli (L)	61.97	0.0	**63.81**	65.50	64.51	65.80	67.56	69.25	70.53	70.90	73.33	74.07	75.23
dighe1 (A)	129.03	0.1	**136.93**	137.55	137.27	142.88	143.77	146.72	146.72	149.25	146.27	154.18	154.93
dighe1 (L)	131.42	0.1	143.78	**139.70**	140.43	142.83	144.80	149.16	148.94	152.94	150.30	155.22	155.94
dighe2 (A)	126.15	0.0	**135.90**	137.50	136.68	138.03	139.86	138.85	142.86	143.59	142.70	146.48	146.70
dighe2 (L)	127.60	0.4	137.07	**135.30**	135.35	137.44	139.69	141.02	141.80	144.02	145.40	144.07	150.19
fu (A)	33.09	0.1	36.03	**35.94**	36.67	37.07	37.26	37.63	37.81	39.08	39.48	38.68	39.98
fu (L)	34.00	0.1	36.74	**36.54**	36.61	36.78	37.19	37.90	37.89	38.14	37.95	38.26	38.86
jakobs1 (A)	12.00	0.0	**12.81**	13.61	14.30	14.91	16.38	17.15	18.33	20.64	22.79	25.20	26.03
jakobs1 (L)	12.57	0.0	**13.28**	13.89	15.00	14.96	15.87	17.01	17.67	20.38	22.03	25.22	26.75
jakobs2 (A)	2600.00	0.0	**2752.07**	2823.51	2977.19	3147.92	3277.71	3452.27	3604.55	3681.44	3744.24	3821.21	3801.76
jakobs2 (L)	2755.00	0.0	2842.68	**2830.79**	2940.73	3174.44	3266.67	3440.90	3574.12	3672.36	3696.53	3768.40	3839.07
mao (A)	1901.41	0.1	2014.49	**1963.29**	1996.73	2065.23	2137.81	2159.03	2223.58	2242.60	2256.98	2316.86	2362.44
mao (L)	1927.05	0.0	2032.26	**2006.12**	2031.69	2068.75	2147.62	2194.83	2172.55	2237.67	2259.95	2296.20	2334.81
marques (A)	80.85	0.0	**84.12**	85.11	86.29	87.32	89.39	89.45	91.87	93.66	95.91	97.78	99.31
marques (L)	82.00	0.0	**84.51**	84.51	85.90	87.18	88.89	89.34	90.22	93.06	93.86	96.37	96.27
poly1a (A)	15.35	0.0	**16.32**	16.64	17.30	17.71	18.40	19.68	20.63	21.44	23.42	25.34	26.94
poly1a (L)	15.71	0.0	**16.70**	16.81	17.23	18.01	18.28	19.43	20.25	21.24	23.03	24.46	27.15
poly2a (A)	2978.60	0.0	**3085.93**	3145.51	3219.63	3272.06	3331.85	3392.18	3483.45	3585.17	3623.39	3821.64	3854.26
poly2a (L)	2997.01	0.0	**3092.54**	3136.93	3212.17	3323.18	3333.31	3397.97	3559.16	3610.18	3695.33	3787.75	3935.84
poly2b (A)	32.24	0.1	**33.45**	33.87	34.67	35.55	36.10	36.79	37.37	38.55	39.43	40.33	41.23
poly2b (L)	32.57	0.0	**33.79**	33.82	34.85	35.58	35.76	37.02	38.08	39.00	39.78	41.01	41.22
poly3a (A)	4444.83	0.0	**4581.66**	4656.15	4697.16	4767.39	4830.78	4895.74	4940.43	5025.27	5185.28	5201.02	5326.31
poly3a (L)	4493.54	0.0	**4600.23**	4644.92	4709.59	4776.15	4853.02	4933.59	4980.63	5097.75	5178.69	5263.51	5341.77
poly3b (A)	42.16	0.1	**43.96**	44.12	44.94	45.65	46.53	47.63	48.14	49.19	49.60	51.08	51.74
poly3b (L)	43.21	0.1	44.91	**44.31**	44.97	45.55	46.66	47.55	48.03	49.22	50.57	50.38	51.52
poly4a (A)	5902.27	0.0	**6056.42**	6129.39	6181.84	6308.71	6347.89	6408.34	6476.02	6573.24	6559.93	6682.29	6730.84
poly4a (L)	5943.66	0.0	**6103.83**	6111.37	6198.07	6291.99	6371.13	6426.94	6507.04	6593.50	6650.61	6741.94	6784.20
poly4b (A)	52.71	0.0	**54.95**	55.72	55.94	57.51	57.98	58.94	60.15	60.61	61.30	61.61	63.60
poly4b (L)	54.25	0.0	55.27	**55.09**	56.02	57.19	58.32	58.68	59.67	60.39	61.64	62.35	63.28
poly5a (A)	6916.11	1.0	**7534.36**	7616.53	7643.15	7736.69	7828.43	7908.50	8000.73	8053.74	8131.32	8153.69	8124.54
poly5a (L)	7427.43	0.0	**7605.56**	7651.62	7722.67	7802.81	7891.41	7929.57	8030.67	8069.36	8166.82	8137.36	8236.33
poly5b (A)	62.24	0.0	**64.94**	65.49	66.08	67.48	68.70	68.91	69.62	70.84	71.38	72.18	74.29
poly5b (L)	63.36	0.1	**65.23**	65.24	66.25	66.82	67.99	69.24	69.60	70.79	71.99	72.88	73.37

continued

Problem	Best Result	Best Result SR	SR=0.0	SR=0.1	SR=0.2	SR=0.3	SR=0.4	SR=0.5	SR=0.6	SR=0.7	SR=0.8	SR=0.9	SR=1
profiles1 (A)	1475.65	0.2	1532.01	**1530.55**	1535.28	1562.43	1640.02	1737.57	1816.05	1861.09	1876.23	1956.68	1964.57
profiles1 (L)	1447.71	0.0	**1492.32**	1521.15	1505.85	1585.26	1670.53	1697.07	1867.93	1906.75	1939.27	1931.32	1960.45
profiles2 (A)	3426.74	0.1	**3553.13**	3577.36	3669.88	3601.22	3743.35	3889.91	3975.37	3995.10	4053.80	4200.64	4224.83
profiles2 (L)	3309.58	0.1	3437.52	**3370.91**	3423.95	3474.12	3532.05	3583.74	3616.20	3806.38	3890.68	3988.28	3988.93
profiles3 (A)	8201.07	0.4	8752.50	**8650.55**	8712.12	8721.77	8638.97	8796.82	8850.33	9085.69	9245.12	9225.88	9190.83
profiles3 (L)	8325.45	0.2	8618.30	**8487.11**	8495.58	8508.36	8558.55	8640.69	8749.29	8909.36	8961.16	9147.70	9096.54
profiles4 (A)	2506.48	0.2	2559.22	2543.46	**2537.89**	2624.46	2634.79	2617.73	2655.59	2657.91	2669.26	2674.05	2680.29
profiles4 (L)	2468.34	0.1	2557.10	**2517.77**	2531.80	2591.50	2611.79	2621.08	2646.38	2648.26	2672.45	2671.35	2692.12
profiles5 (A)	3492.64	0.0	3565.29	**3563.44**	3581.65	3746.73	3832.76	3891.73	4043.34	4095.76	4142.05	4227.21	4354.65
profiles5 (L)	3510.19	0.1	3641.58	**3581.85**	3671.53	3723.41	3846.12	3940.06	4031.19	4096.98	4185.00	4344.12	4454.52
profiles6 (A)	2710.33	0.5	**3282.39**	3299.91	3437.96	3492.36	3744.44	3511.64	3698.18	3810.52	3877.45	4111.19	4061.57
profiles6 (L)	3073.46	0.6	**3210.50**	3291.55	3383.18	3472.50	3642.74	3768.34	3696.95	3808.01	3913.79	4040.71	4054.89
profiles7 (A)	1221.32	0.6	**1409.48**	1428.97	1411.38	1433.42	1435.21	1410.70	1359.87	1329.61	1330.53	1339.32	1385.12
profiles7 (L)	1221.32	0.0	**1389.64**	1422.67	1409.73	1459.72	1441.78	1457.29	1408.84	1442.45	1443.06	1408.48	1516.90
profiles8 (A)	1298.12	0.1	1390.51	1376.31	**1362.92**	1419.82	1437.35	1528.17	1588.67	1617.35	1540.13	1574.15	1618.03
profiles8 (L)	1339.89	0.1	1419.81	**1416.70**	1433.13	1449.54	1466.49	1467.79	1539.37	1541.75	1573.07	1656.38	1638.66
profiles9 (A)	1331.42	0.0	**1354.17**	1405.02	1436.19	1447.33	1500.18	1516.02	1551.95	1576.02	1635.46	1683.08	1707.84
profiles9 (L)	1335.52	0.1	1383.40	**1358.32**	1427.11	1483.82	1504.42	1548.06	1558.62	1599.79	1615.64	1648.55	1703.03
profiles10 (A)	11696.80	0.6	**11822.46**	11839.80	11876.11	11861.62	11920.20	11949.93	12050.52	12080.73	12185.29	12198.24	12329.09
profiles10 (L)	11425.73	0.2	11686.60	11697.25	11674.17	**11611.75**	11736.70	11842.80	11861.71	11910.98	11955.57	12058.36	12157.09
shapes0 (A)	64.00	0.1	67.25	**65.85**	67.25	68.22	68.95	69.40	70.44	71.56	71.11	71.80	74.13
shapes0 (L)	64.00	0.1	66.55	**65.17**	66.35	67.35	67.50	68.50	68.67	70.06	69.67	71.70	73.71
shapes1 (A)	64.50	0.1	66.05	**65.33**	67.20	68.10	68.42	69.29	70.45	70.28	70.60	73.00	72.11
shapes1 (L)	63.00	0.1	65.60	**64.94**	65.78	67.40	67.65	68.40	68.90	70.03	71.38	71.00	72.44
shapes (A)	62.50	0.1	**65.00**	65.50	65.95	67.33	69.20	70.50	71.88	71.78	72.20	72.00	73.10
shapes (L)	63.00	0.0	**65.00**	65.18	64.70	67.00	67.85	69.50	70.65	71.93	72.57	72.50	73.30
shirts (A)	65.48	0.0	**66.73**	66.82	66.92	67.60	68.14	68.50	68.80	69.99	69.72	71.64	71.22
shirts (L)	64.80	0.0	**65.45**	66.00	66.45	67.57	67.61	67.90	69.20	69.87	70.85	71.40	72.12
swim (A)	6545.42	0.0	**6672.64**	6720.17	6942.00	7020.21	7124.14	7176.68	7387.82	7541.09	7571.71	7817.29	7870.92
swim (L)	6594.84	0.0	**6876.75**	6932.17	7023.01	6952.82	7233.57	7317.71	7460.63	7478.81	7811.33	7762.17	7915.95
trousers (A)	262.98	0.9	279.10	**271.58**	274.99	272.05	276.42	279.53	277.15	277.58	278.12	275.55	277.76
trousers (L)	261.38	0.0	274.11	273.33	273.63	**273.07**	277.57	277.27	281.28	281.19	281.71	277.44	277.62

It is clear that whilst the non-greedy placement approach has been able to outperform the greedy placement approach in some cases the majority of the results generated are poor when compared to the greedy approach, especially when the Shape Ratio is high. Both the best result and the best average results tend to occur in when the approach is using the greedy placement strategy or the Shape Ratio = 0.1. This is because of the detrimental effect a late in the placement sequence non-greedy placements can have on the efficiency of the layout generated. For this reason the Sequence Factor, described in the next section, was introduced to control non-greedy placement towards the end of a sequence of placements.

The average layout generation time for the benchmark problems is presented in Table 24. No-fit polygon generation is performed in advance of the testing and it therefore not included in the table. As the implementation of this technique was developed in a different programming language it is difficult to make useful comparisons with the performance of techniques used in chapters 4 and 6. However, the time per layout appears to be acceptable for industrial application.

Table 24. Average Layout Generation Times

Problem	Average Generation Time (s)
Albano	0.33
Blaz	0.25
Dagli	0.5
dighe1	0.07
dighe2	0.04
Fu	0.05
jakobs1	0.25
jakobs2	0.2
Mao	0.33
marques	0.33
poly1a	0.1
poly2a	0.25
poly2b	0.6
poly3a	0.8
poly3b	1.2
poly4a	1.1
poly4b	1.3
poly5a	1.3
poly5b	2.4
profiles1	0.3
profiles2	0.95
profiles3	2.3
profiles4	0.25
profiles5	7
profiles6	1.8
profiles7	0.04
profiles8	0.2
profiles9	0.14
profiles10	0.8
shapes0	0.13
shapes1	0.13
shapes	0.13
shirts	0.7
swim	3.1
trousers	0.6

7.6. Sequence Factor

Whilst non greedy placement selection using the Shape Ratio is a useful method for selecting non-greedy placements it is obvious that selecting non-greedy positions toward the end of a layout will produce a longer layout and therefore less efficient layouts. Furthermore selecting late non-greedy placements has no benefit as there are fewer and fewer shapes to be placed later on to utilise the space deliberately left for later placements.

To react to this observation the Sequence Factor calculation was introduced following the experiments shown in section 7.5. The Sequence Factor acts as a progressive reduction on the selection of non-greedy placements allowing for non-greedy selection to be reduced or eliminated towards the end of a layout sequence.

The Sequence Factor is used to limit the *x* distance, a factor of the shape's *x* span, which a candidate placement may be from the most efficient placement point. If the Sequence Factor for all candidate placements is greater than the supplied limit then the most efficient placement point will be selected, switching the placement strategy back to a greedy strategy.

Shape Factor is calculated for each point in the following manner:

Algorithm 10. Sequence Factor calculation process

```
Input
CandidatePoint = valid potential placement point
BestPoint = most efficient valid potential placement (produces placement with least
additional layout length)
SequenceIndexOfCandidateShape = zero based index in sequence of part being placed
SequenceLength = the length of the placement sequence
SequenceFactor = the current SequenceFactor value being tested

Begin
        double xDiff = CandidatePoint.RightBoundOfPartAtThisPoition -
        BestPoint.RightBoundOfPartAtThisPoition;

        double factor = ((SequenceLength - SequenceIndexOfCandidateShape) / SequenceLength)
        * SequenceFactor;

        double SequenceValue = xDiff -((candidatePart.PartWidthAtPlacementAngle) * factor);

        bool bValid = false;
        if(sequenceValue <= 0.0)
                bValid = true;

        return bValid;
End
```

The effect of the Sequence Factor is demonstrated in the following figures. The second value displayed for each point is the calculated `SequenceValue` from the above algorithm. In order for a point to be admissible, given the Sequence Factor and the index of the shape in the sequence of shapes to be placed, the Sequence Factor calculation must result in a negative or zero value. If the value is greater than zero the point is not considered as a valid potential placement.

In the following example (Figures 76, 77 and 78) a Shape Ratio of 0.7 is used, in common with the example shown in previous section. The Sequence Factor being used in the following example is 0.5. It should be noted that when a Sequence Factor value is being used the final placement of any layout sequence will always be in the greedy bottom-left position.

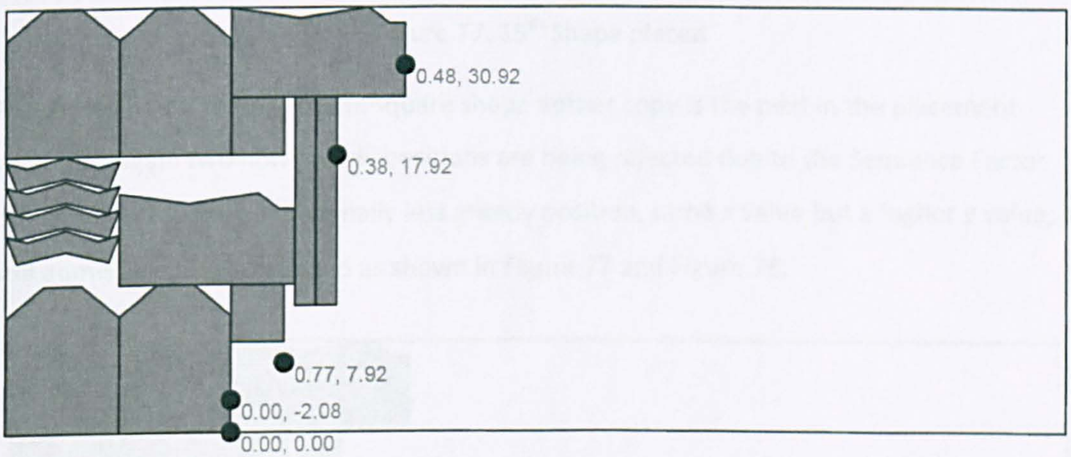


Figure 76. Potential placements including their Sequence Factor value

Figure 76 shows the potential placements for another copy of the square shape, the 15th shape of a sequence of 24, which the TOPOS algorithm has generated. However although three potential placements are admissible, according to the Shape Ratio criteria (0.7), the calculated Sequence Factor values mean that the points with 0.38 and 0.48 Shape Ratio values are no longer regarded as valid potential placements.

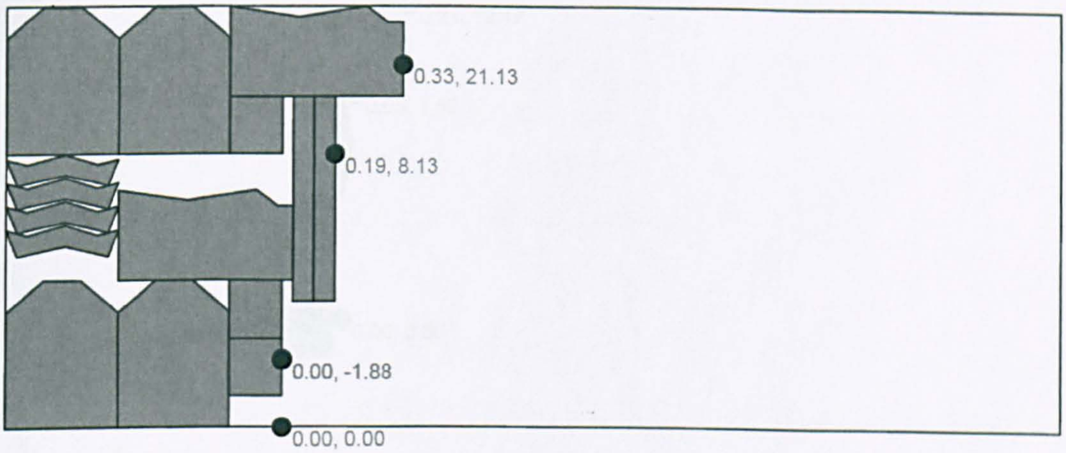


Figure 77. 15th Shape placed

Following the placement of the square shape anther copy is the next in the placement sequence again two non-greedy positions are being rejected due to the Sequence Factor calculation. However a marginally less greedy position, same *x* value but a higher *y* value, is still admissible and is selected as shown in Figure 77 and Figure 78.

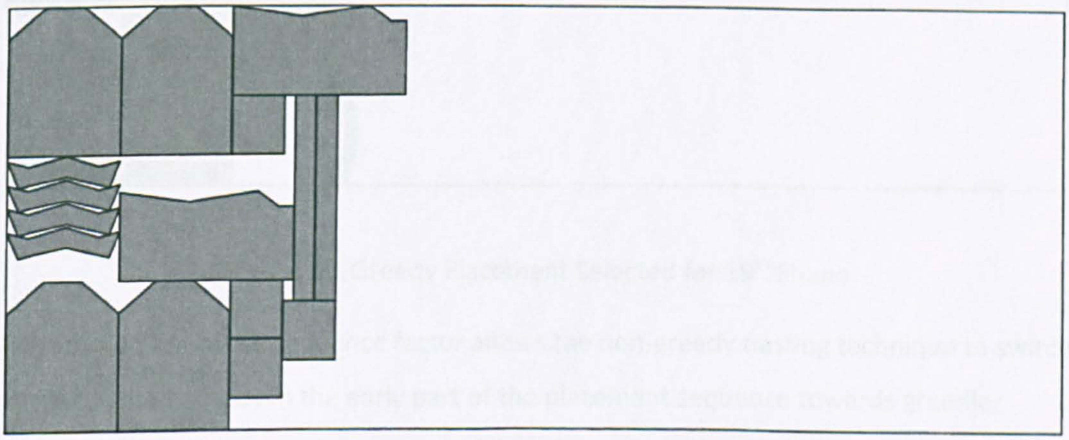


Figure 78. 16th Shape placed

Later in the placement sequence we can see the Sequence Factor having a greater effect. Figure 79 shows the placement of the 19th shape in the sequence, here all non-greedy positions are no longer admissible and so the greedy placement is selected as shown in Figure 80.

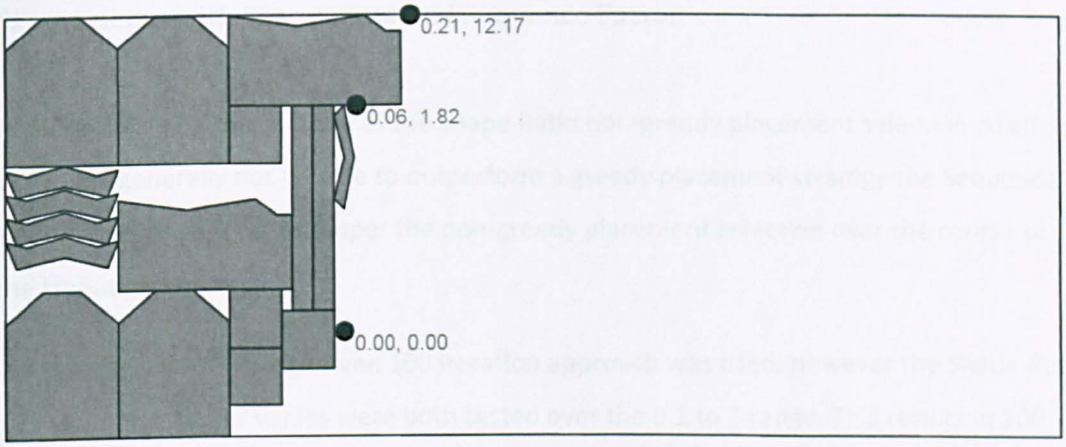


Figure 79. 19th Shape Potential Placements

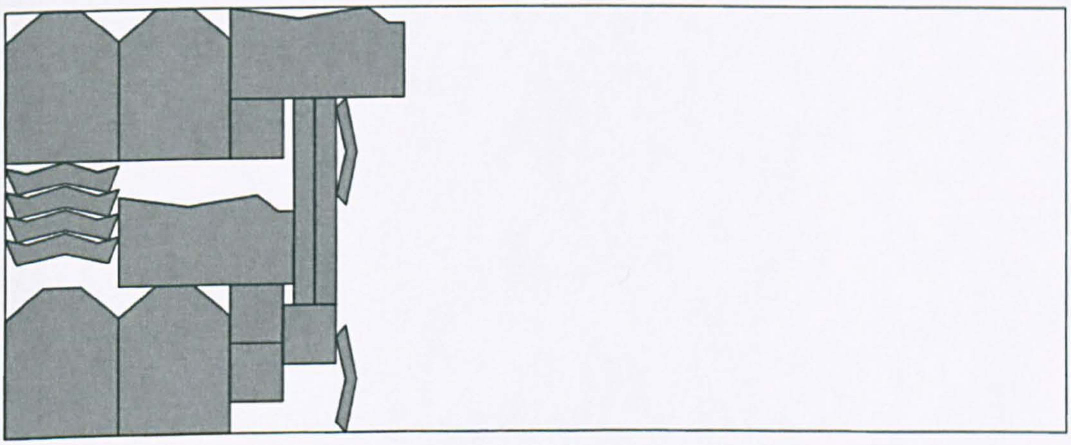


Figure 80. Greedy Placement Selected for 19th Shape

The introduction of the sequence factor allows the non-greedy nesting technique to switch from being non-greedy in the early part of the placement sequence towards greedier placement choices later in the sequence. This allows the central gaps deliberately left by the Shape Ratio calculation to be filled and also avoids placing shapes in non-greedy placements towards the end of the sequence and therefore adversely affecting the overall layout length. Using these two simple inputs, Shape Ratio and Sequence Factor, we can control the greediness of the selection of placements points that the TOPOS technique offers.

Section 7.7 presents the experimental results where both Shape Ratio and Sequence Factor are utilised.

7.7. Experiments with Shape Ratio and Sequence Factor

After establishing that the use of the Shape Ratio non-greedy placement selection strategy alone will generally not be able to outperform a greedy placement strategy the Sequence Factor was introduced to temper the non-greedy placement selection over the course of the layout generation.

Once again a hill climbing driven 100 iteration approach was used, however the Shape Ratio and Sequence Factor values were both tested over the 0.1 to 1 range. This results in 100 runs of 100 iterations for each problem, with both length and area initial orderings.

Table 25 presents the best results from these experiments:

Table 25. Best Results with Shape Ratio and Sequence Factor

Problem	Area/Length	SR	SF	Best Result	SR Only Best	Literature Best	Note
Albano	Area	0.1	0.2	**10199.58 (85.4%)**	10355.86	(Sato et al., 2012)	< Chapter 4
Albano	Length	0.1	0.2	10256.02 (84.9%)	10360.59	89.21%	
Blaz	Area	0.5	0.9	27.54	28.08	(Bennell & Song, 2008)	
Blaz	Length	0.9	0.4	27.67	28.08	26.57	
Dagli	Area	0.2	0.7	**60.19 (84%)**	62.62	(Sato et al., 2012)	< Chapter 4
Dagli	Length	0.1	0.6	61.11 (82.7%)	61.97	57.4	
dighe1	Area	0.9	0.1	124.20 (80.5%)	129.03	(Gomes & Oliveira, 2006)	< Chapter 4
dighe1	Length	0.6	0.7	**122.20 (81.8%)**	131.42	100%	
dighe2	Area	0.8	0.1	**111.60 (89.6%)**	126.15	(Gomes & Oliveira, 2006)	< Chapter 6
dighe2	Length	0.9	0.3	113.16 (88.4%)	127.60	100%	
Fu	Area	0.8	0.1	33.00	33.09	(Egeblad et al., 2007)	
Fu	Length	0.8	0.2	33.00	34.00	30.8	
jakobs1	Area	1	0.3	12.00	12.00	(Sato et al., 2012)	
jakobs1	Length	1	0.1	12.00	12.57	11	
jakobs2	Area	0.7	0.1	2600.00	2600.00	(Sato et al., 2012)	
jakobs2	Length	0.1	0.4	2600.00	2755.00	2275	
Mao	Area	0.7	0.7	1858.05 (79.3%)	1901.41	(Egeblad et al., 2007)	
Mao	Length	0.3	0.8	1863.83 (79%)	1927.05	85.15%	
Marques	Area	0.2	0.7	**80.00**	80.85	(Egeblad et al., 2007)	= Chapter 4
Marques	Length	1	0.6	**80.00**	82.00	76.67	
poly1a	Area	0.6	0.1	14.50	15.35	(Burke et al., 2010)	
poly1a	Length	0.1	0.2	14.69	15.71	13.3	
poly2a	Area	0.4	0.9	2820.76	2978.60	(Burke et al., 2010)	
poly2a	Length	0.7	0.7	2885.29	2997.01	2790	
poly2b	Area	0.1	0.2	31.28	32.24	(Burke et al., 2010)	
poly2b	Length	0.1	0.7	31.55	32.57	29.63	
poly3a	Area	0.1	0.2	4309.94	4444.83	(Burke et al., 2006)	
poly3a	Length	0.4	1	4381.80	4493.54	4033	
poly3b	Area	0.5	0.1	41.41	42.16	(Burke et al., 2010)	
poly3b	Length	1	0.2	41.44	43.21	40.5	
poly4a	Area	0.4	0.2	5741.21	5902.27	(Burke et al., 2010)	
poly4a	Length	0.8	0.7	5811.37	5943.66	5460	
poly4b	Area	0.3	0.1	52.29	52.71	(Burke et al., 2010)	
poly4b	Length	0.8	0.6	52.19	54.25	51.18	
poly5a	Area	0	0	7249.06	6916.11	(Burke et al., 2010)	
poly5a	Length	0.9	1	7299.00	7427.43	6884	
poly5b	Area	0.3	0.4	62.11	62.24	(Bennell & Song, 2008)	
poly5b	Length	0.9	0.2	62.26	63.36	57.53	
profiles1	Area	0.5	0.9	1423.31	1475.65	(Burke et al., 2010)	
profiles1	Length	0.3	0.9	1402.00	1447.71	1359.9	
profiles2	Area	0.1	0.7	3244.90	3426.74	(Burke et al., 2010)	
profiles2	Length	0.2	0.4	3204.68	3309.58	3194.19	
profiles3	Area	0.9	0.6	8142.56	8201.07	(Burke et al., 2010)	
profiles3	Length	0.4	0.5	8141.98	8325.45	7881.13	
profiles4	Area	0.6	0.9	2450.154	2506.48	(Burke et al., 2010)	
profiles4	Length	0.2	0.3	2441.481	2468.34	2425.26	
profiles5	Area	0.2	1	3367.67	3492.64	(Burke et. al 2006)	
profiles5	Length	0.1	0.9	3432.91	3510.19	3332.7	
profiles6	Area	0.5	0.9	**3094.72**	3282.39	(Burke et. al 2006)	New best result
profiles6	Length	0.8	0.3	**3061.58**	3073.46	3097.86	
profiles7	Area	0.8	0.6	**1000**	1221.32	(Burke et al., 2010)	New best result
profiles7	Length	0.9	0.8	**1000**	1221.32	1292.30	
profiles8	Area	0.7	0.4	**1261.44**	1298.12	(Burke et al., 2010)	New best result
profiles8	Length	0.6	0.9	**1229.18**	1339.89	1263.11	
profiles9	Area	0.5	0.1	1325.52	1331.42	(Burke et al., 2010)	
profiles9	Length	0.1	0.8	1311.50	1335.52	1278.21	
profiles10	Area	0.2	1	11446.26	11696.80	(Burke et. al 2006)	
profiles10	Length	0.3	0.7	11248.85	11425.73	11160	
shapes0	Area	0.2	0.5	**62.00**	64.00	(Imamichi et al., 2009)	< Chapter 4
shapes0	Length	0.9	0.1	**62.00**	64.00	58.3	
shapes1	Area	0.1	0.7	63.00	64.50	(Leung et al., 2012)	
shapes1	Length	0.1	0.3	61.50	63.00	53	
Shapes	Area	1	0.8	62.00	62.50	(Burke et al., 2010)	
Shapes	Length	0.6	0.6	62.00	63.00	56	
Shirts	Area	0.3	0.7	64.36	65.48	(Imamichi et al., 2009)	
Shirts	Length	0.9	0.9	64.48	64.80	60.83	
Swim	Area	0.6	0.1	**6396.32**	6545.42	(Imamichi et al., 2009)	< Chapter 6
Swim	Length	0.4	0.4	6549.98	6594.84	5875.17	
Trousers	Area	0.4	0.1	257.67	262.98	(Bennell & Song, 2008)	
Trousers	Length	1	0.3	255.95	261.38	241	

As shown in Table 25 the introduction of the Sequence Factor has significantly improved the performance of the non-greedy placement technique. Indeed it has been able to improve upon several best results from the earlier chapters of this thesis and find an optimal result for Profiles7, which is not possible using a greedy placement strategy. Whilst it has not been able to achieve any new best results for the literature benchmarks it is, in all but a few cases, reasonably competitive especially given that the tests consisted of only 100 iteration hill climb driven runs.

With so many values of Shape Ratio and Sequence Factor being explored in the second set of experiments displaying average values for each problem is best achieved in the form of a graph. Graphs of the minimum and average values, across each Shape Ratio and Sequence Factor value pair, for the 10 runs of all problems in both initial orderings are provided in Appendix D. These graphs present the result in its layout length form in the *y* axis and value pairs of Shape Ratio / Sequence Factor in the *x* axis. Along the *x* Shape Ratio rises steadily and Sequence Factor moves from 0.1 to 1.0 for each value increment of Shape Ratio. The first point on the graph is always the greedy implementation's min and average, as show in Figure 81.

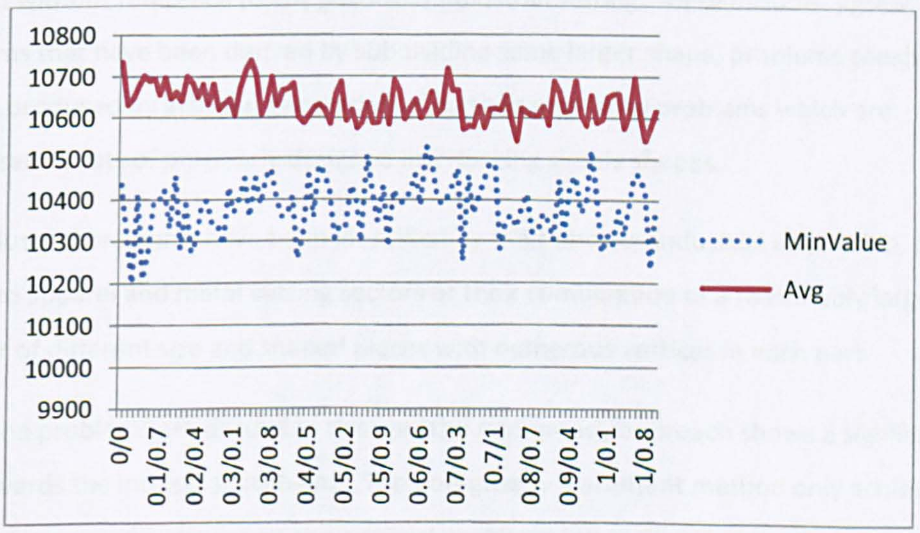


Figure 81. Min and Average data for Albano (Area Sorted)

In order to discern if the non-greedy placement approach is a potentially useful addition to a general automatic nesting application, i.e. one designed to work well on a wide range of problem types, it is important that we analyse not only the best results but also the average results from the non-greedy and greedy approaches. Indeed best results could simply be attributed to a fortuitous set of operators acting upon the sequence for only one of the test runs used to explore this new technique and not point towards the general usefulness of the new technique.

To assess the average effectiveness of the technique across the range of problems the average result for the greedy approach is compared to the average result for the non-greedy value pairings of Shape Ratio and Sequence Factor. Table 26 presents the results of this comparison highlighting the percentage of occasions the Shape Ratio and Sequence Factor controlled runs resulted in a better run average than the greedy placement runs. If the non-greedy technique is having little or no effect on average we would expect to see approximately only 50% of the non-greedy average results better than the greedy approach average. However when the problems are taken as a whole we see a figure of 60.8% of non-greedy runs presenting an improved average result over the greedy run averages.

Furthermore given that this approach is aimed at improving on the results generated for the industrial problems, that a general automatic nesting engine may expect to encounter, the benchmarks have then been separated into two categories, artificial and industrial.

Problems classified as artificial are those from the benchmark instances that have been created without reference to any particular industrial setting. These include "jigsaw" problems that have been derived by subdividing some larger shape, problems consisting of shapes produced by artificial generation of vertices and those problems which are composed of sets of purposely designed interlocking simple shapes.

The industrial problems have been identified by their obvious industrial inspiration, often from the apparel and metal cutting sectors or their combination of a reasonably large number of different size and shaped pieces with numerous vertices in each part.

When the problems are divided in this way the non-greedy approach shows a significant bias towards the industrial problems. The non-greedy placement method only achieves 55.4% of improved averages on the artificial problems but in 68.8% of the tests the on the problems in the industrial category the non-greedy approach is able to produce improved averages, when compared to the greedy placement strategy.

Table 26. Artificial and industrial problems average result comparison

Problem Type	Name	Area / Length	% of SR+SF averages better than greedy average
Artificial	Blaz	Area	21
	Blaz	Length	4
	dighe1	Area	12
	dighe1	Length	97
	dighe2	Area	43
	dighe2	Length	83
	Fu	Area	33
	Fu	Length	91
	jakobs1	Area	11
	jakobs1	Length	54
	jakobs2	Area	43
	jakobs2	Length	100
	poly1a	Area	30
	poly1a	Length	95
	poly2a	Area	16
	poly2a	Length	38
	poly2b	Area	28
	poly2b	Length	90
	poly3a	Area	16
	poly3a	Length	83
	poly3b	Area	46
	poly3b	Length	100
	poly4a	Area	9
	poly4a	Length	83
	poly4b	Area	17
	poly4b	Length	94
	poly5a	Area	20
	poly5a	Length	90
	poly5b	Area	36
	poly5b	Length	65
	profiles7	Area	64
	profiles7	Length	27
	profiles8	Area	93
	profiles8	Length	75
	profiles9	Area	0
	profiles9	Length	52
	Shapes	Area	29
	Shapes	Length	88
	shapes0	Area	100
	shapes0	Length	100
	shapes1	Area	59
	shapes1	Length	92
Industrial	Albano	Area	99
	Albano	Length	100
	Dagli	Area	91
	Dagli	Length	7
	Mao	Area	99
	Mao	Length	100
	Marques	Area	67
	Marques	Length	75
	profiles1	Area	52
	profiles1	Length	28
	profiles2	Area	26
	profiles2	Length	91
	profiles3	Area	98
	profiles3	Length	100
	profiles4	Area	53
	profiles4	Length	55
	profiles5	Area	93
	profiles5	Length	92
	profiles6	Area	93
	profiles6	Length	29
	profiles10	Area	52
	profiles10	Length	93
	Shirts	Area	93
	Shirts	Length	14
	Swim	Area	24
	Swim	Length	36
	Trousers	Area	98
	Trousers	Length	69
		Overall	**60.8**
		Industrial	**68.8**
		Artificial	**55.4**

It is also of interest to identify the values of shape ratio and sequence factor that are able to produce good average results. To determine these values the data from all of the benchmark problems was analysed for the lowest 10 average values, for each of these values the shape ratio and sequence factor responsible for the low average was noted and occurrences of these value pairs summed. This data is presented in the chart in Figure 82.

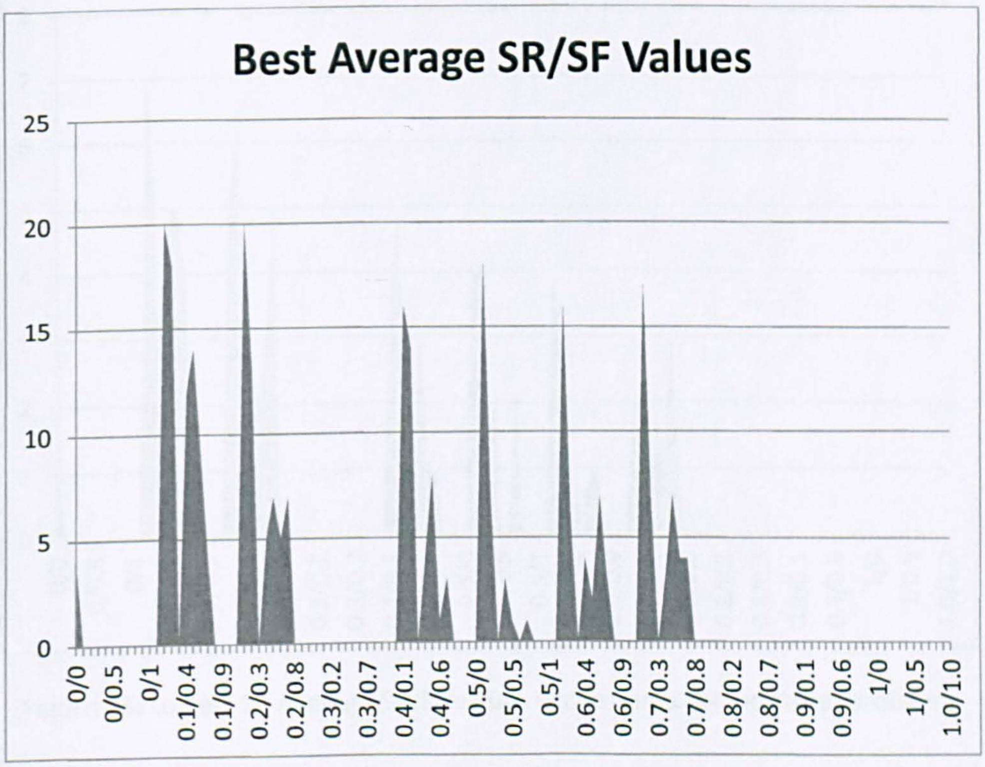


Figure 82. Lowest 10 Average SR/SF value occurrences

The analysis points toward the set of values show in Table 27 being the most useful combination of settings for shape ratio and sequence factor across a broad range of problems. This information will allow for the more focused application of the technique and likely allows the exclusion of values of shape ratio above 0.7 from any further application.

Table 27. The most productive Shape Ratio and Sequence Factor values

Shape Ratio	Sequence Factor
0.1	0 – 0.8
0.2	0 – 0.8
0.4	0 – 0.8
0.5	0 – 0.8
0.6	0 – 0.8
0.7	0 – 0.8

The same analysis was then performed on the two categories of benchmark problems, artificial and industrial, as shown in Figure 83 and Figure 84. Each of these categories also appears to respond positively to a similar set of shape ratio and sequence factor values.

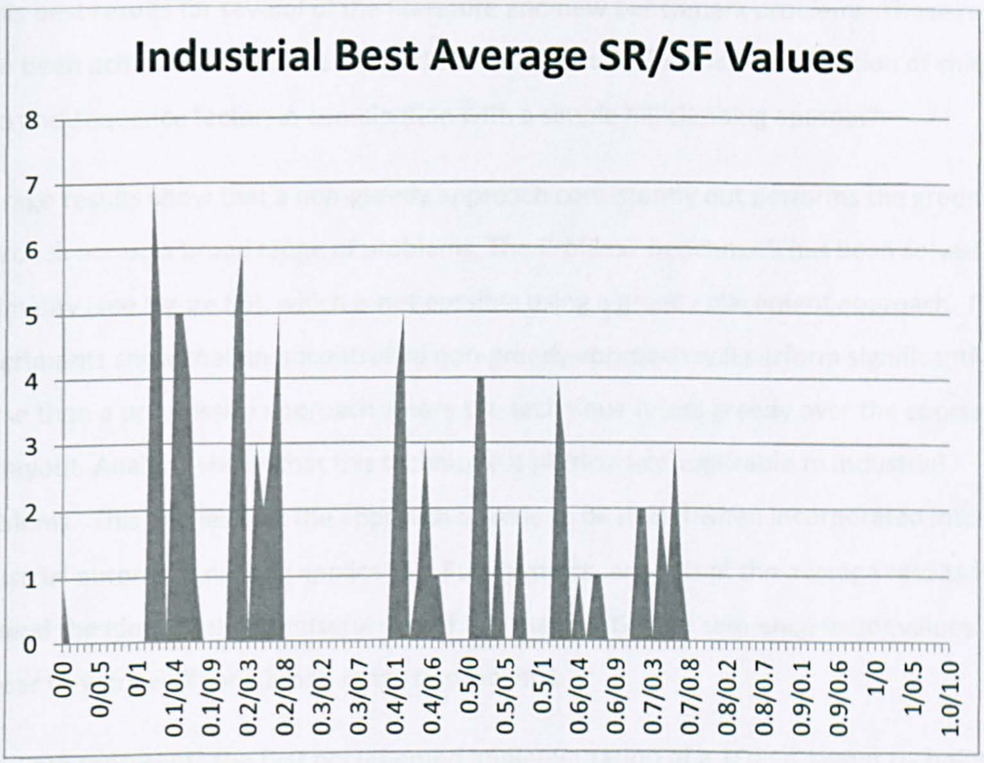


Figure 83. Lowest 10 Average SR/SF values occurrences for industrial problems

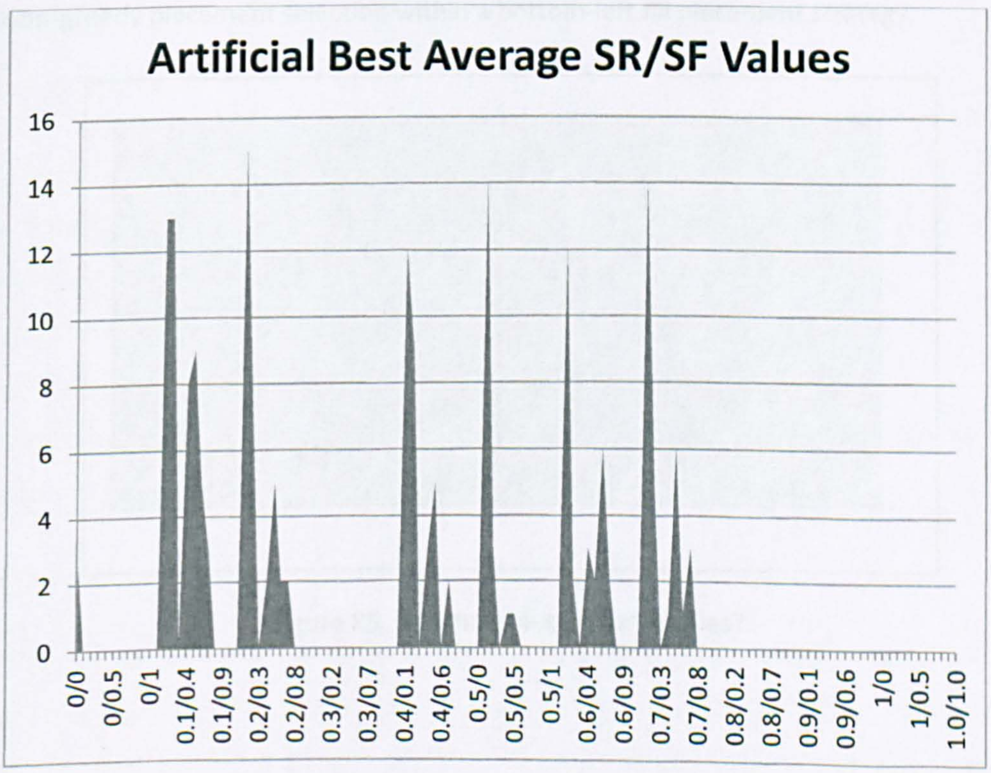


Figure 84. Lowest 10 Average SR/SF values occurrences for artificial problems

7.8. Summary

The simple non-greedy placement technique introduced in this chapter has generated new thesis best results for several of the literature and new benchmark problems. These results have been achieved using a set of 10 x 100 iteration runs, for each combination of shape ratio and sequence factor, in combination with a simple hill climbing approach.

Average results show that a non-greedy approach consistently out performs the greedy approach across a broad range of problems. The Profiles7 benchmark has been solved to optimality (see Figure 85), which is not possible using a greedy placement approach. The experiments show that an uncontrolled non-greedy approach will perform significantly worse than a progressive approach where the technique is less greedy over the course of the layout. Analysis shows that this technique is particularly applicable to industrial problems. This implies that the approach is likely to be useful when incorporated into an industrial automatic nesting application. Furthermore, analysis of the average results has allowed the identification of useful sets of the shape ratio and sequence factor values that appear to work well for a broad range of problems.

This work represents the first documented implementation of a TOPOS layout technique for problems including arc segments and shapes with holes and is the only literature reference for non-greedy placement selection within a bottom-left fill placement strategy.

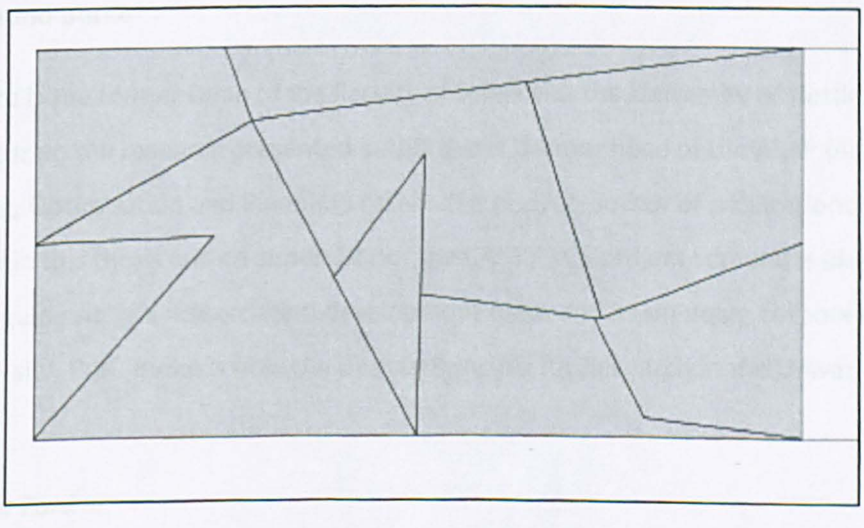


Figure 85. Optimal solution for Profiles7

CHAPTER EIGHT

8. Case Study

This chapter discusses the companies, products and projects utilising the research generated during the CASE / TCS projects and also the research presented in this thesis. Bridging the gap between research and practise has been an important theme in the research community and the industrial foundation of the research and its applicability to numerous industries ensures that the day to day considerations of practitioners and end-users have been of upmost importance throughout this thesis. The on-going industrial application of the research presented in this thesis has provided additional challenging constraints and drivers for further research and developments.

8.1. Introduction

Aptia Solutions Limited was founded in July 2004 as a spin-out company from the University of Nottingham. The intention of the company is to exploit and further the research presented in this thesis by developing products applicable to the many industrial sectors where nesting is of interest.

The following team of people sit on the company board:

Prof. Edmund Burke

Prof. Burke is the former Dean of the Faculty of Science at the University of Nottingham. As a supervisor on the research presented in this thesis, former head of the ASAP (Automated Scheduling, Optimisation and Planning) research group, co-author of publications presented in this thesis and co-supervisor of the CASE / TCS project Edmund is ideally placed to guide Aptia's research and development focus and its on-going collaboration with the university. Prof. Burke is now the Deputy Principle for Research at the University of Stirling.

Prof. Peter Ford

Peter is an Emeritus Professor of the Faculty of Science at the University of Nottingham. Peter was the Head of the School of Computer Science during the formulation of Aptia as a spin out company. With wide experience as a company director of several UK companies, an accredited fellow of the British Computer Society and long experience in the field of computer science Peter is a valued member of the board on both business and technical matters.

Prof. Graham Kendall

Graham is the Dunford Professor of Computer Science, head of ASAP research group and a supervisor of the research presented in this thesis. Graham has over 20 years of industrial experience in addition to his subsequent academic career. With a comprehensive background in both academia and industry Graham has been instrumental in the formation and start-up period of Aptia and continues to guide its growth. In addition to his board duties Graham also acts as Aptia's press officer.

Dr. Glenn Whitwell (Chairman)

Glenn received his PhD in Computer Science in 2005 from the University of Nottingham, as part of the ASAP research group. As co-author of the research of this thesis and in his position as a research assistant in the ASAP group Glenn is key to the research, software development and business activities of Aptia on a day to day basis. His efforts during the first years of Aptia's formation have laid good foundations for future growth and expansion. Glenn is the Managing Director of Aptia, taking responsibility for all aspects of its running and activities.

University Observer (currently Bruce Venning)

Bruce Venning is the University of Nottingham's observer on the Aptia board. Bruce works as a commercial manager in university's Technology Transfer office. His background in software sales and software businesses have enabled him to give valuable input to the board meetings and he has kept the board abreast of help and support options available to Aptia from the Technology Transfer office.

8.2. Thesis Research Utilised

Aptia uses the following elements of the research presented in this thesis within the company's product range:

- Heuristic Search Techniques – used in chapters 4 and 6
- Line Arc no-fit polygon – used in chapter 6
- Non-greedy placements – used in chapter 7

The use of these techniques allows Aptia's products to offer the on-going search mechanism, which is a unique selling point of Aptia's products. Additionally the fast and highly accurate nesting that the line arc no-fit polygon can provide allows the product range to handle common industrial problem features such as shapes with arcs, holes and nesting onto irregular sheets. The line arc no-fit polygon approach is also used for robust spacing and cut offsetting calculations.

The entire range of Aptia nesting products use the same highly optimised automatic nesting library, including the above techniques and further commercially sensitive developments.

8.3. Aptia Products and Services

To date Aptia has developed a range of products, utilising nesting techniques, which have been adopted in various industries. The most prominent adoption of these products has been in the aerospace and carbon fibre manufacturing sectors.

In the near future it is expected that Aptia will target the boat / ship building and metal cutting sectors in addition to expanding its reputation and reach within the expanding carbon fibre market worldwide.

The following subsections cover the range of products that Aptia have developed over the period 2006 to 2012. All of the products utilise the research presented in this thesis and have been designed to be as user friendly as possible. Indeed the challenge of making products that deliver the nesting technologies to end users in simple, easy to use and easy to understand packages is a very challenging aspect of bridging the gap between research and practise.

Without being delivered as part of a consistent, straight forward and high quality application the quality of the techniques presented in this thesis could not have made such headway in the market place.

Aptia's products have often been chosen over competitive products, even when the competitor product has been shown to outperform Aptia's product on certain problem instances, as the application range is deliberately simple to use and understand. The simplicity and focus on the common end user tasks of the products reduces staff training requirements and the likelihood of software bugs in operation. These two factors make selling the software to end users much easier. Furthermore these aspects make the adoption of the product range by resellers more likely as this allows the reseller to avoid providing long, costly training courses and on-going support requirement.

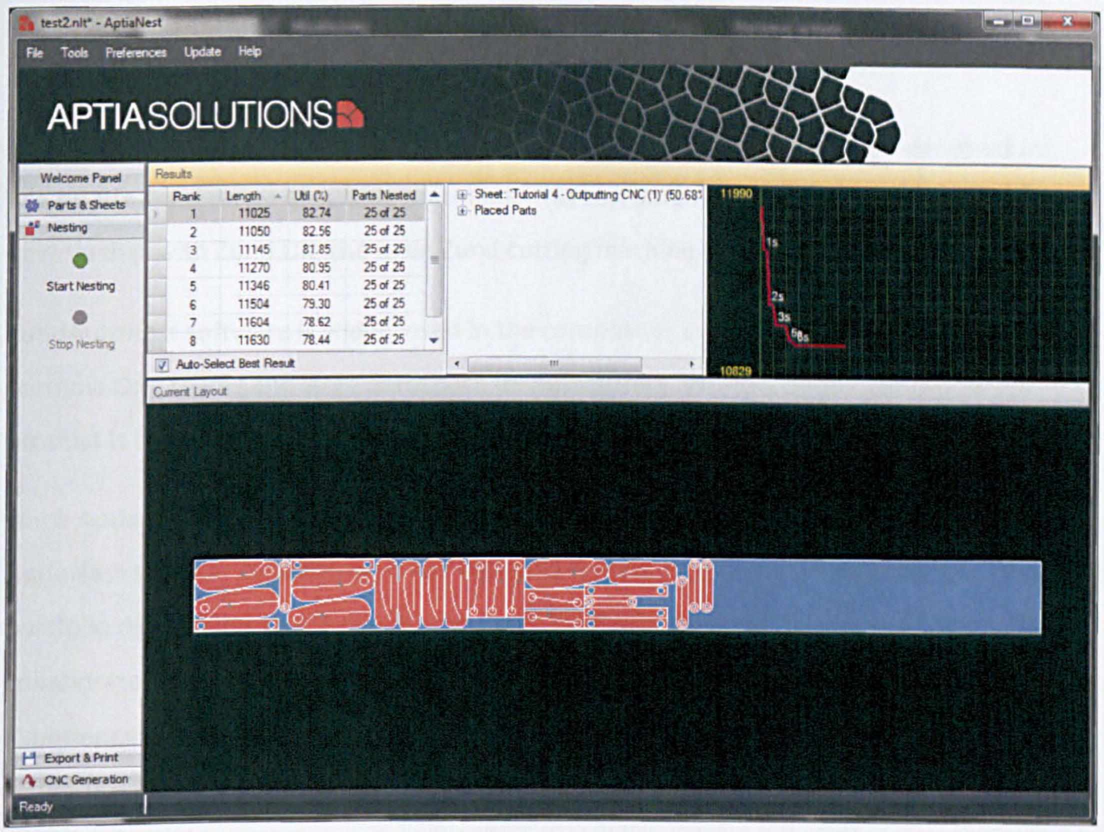
8.3.1. AptiaNest

AptiaNest is Aptia's main product that uses the nesting algorithms developed in the course of this thesis. It can robustly import part and sheet geometry defined in DXF and DWG formats, automatically layout the parts according to user instructions and restrictions and convert the layouts to any required CNC format in order to drive a cutting machine.

It has been adopted by machine resellers worldwide as a software option to accompany various cutting machine sales. As depicted in Figure 86 AptiaNest utilises the on-going search methodologies investigated in this thesis, allowing customers to leave the system finding improved layouts until required for cutting.

One of AptiaNest's main strengths is that it is purposefully easy to use the user interface; see Figure 86 and Figure 87. Many competitive products have been established for considerably longer; in general these were initially focused on the metal cutting or fabrics sectors. This long standing and resultant feature creep has often left the competitor products with multiple options and tools which are only occasionally used by the end users.

The AptiaNest product has been designed with extendibility and the ability to be rebranded from its first inception. It is therefore highly flexible and uses plugin architecture and bespoke licensing to allow it to be configured to particular reseller and customer needs. This has allowed it to drive (generate CNC) for various cutting machines and present multiple additional customisations without becoming a complex piece of software to maintain, sell or use.

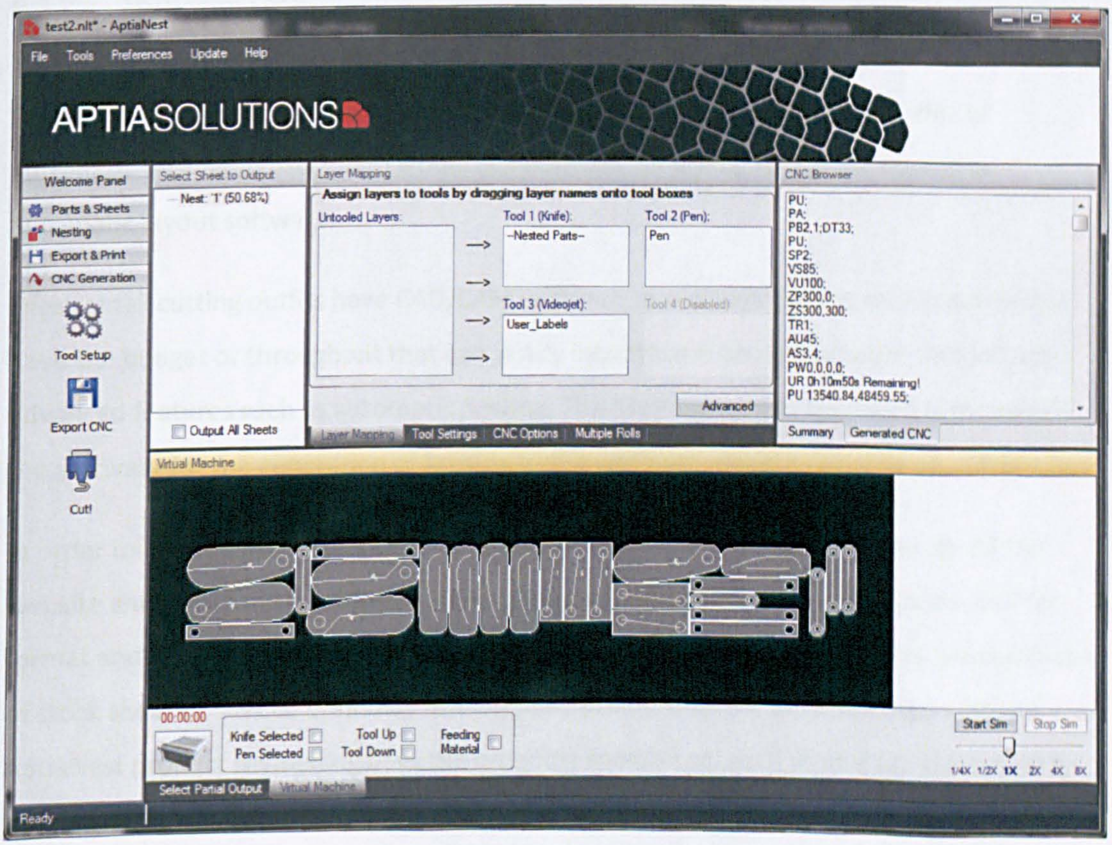


Rank	Length	Util (%)	Parts Nested
1	11025	82.74	25 of 25
2	11050	82.56	25 of 25
3	11145	81.85	25 of 25
4	11270	80.95	25 of 25
5	11346	80.41	25 of 25
6	11504	79.30	25 of 25
7	11604	78.62	25 of 25
8	11630	78.44	25 of 25

☑ Auto-Select Best Result

⊞ Sheet: 'Tutorial 4 - Outputting CNC (1)' (50.68%
⊞ Placed Parts

Figure 86. Continuous search nesting in AptiaNest



Nest: '1' (50.68%)

□ Output All Sheets

Assign layers to tools by dragging layer names onto tool boxes

Untooled Layers:

Tool 1 (Knife): --Nested Parts--

Tool 2 (Pen): Pen

Tool 3 (Videojet): User_Labels

Tool 4 (Disabled)

```
PU;
PA;
PB2,1;DT33;
PU;
SP2;
VS85;
VU100;
ZP300,0;
ZS300,300;
TR1;
AU45;
AS3,4;
PW0,0,0,0;
UR 0h10m50s Remaining!
PU 13540.84,48459.55;
```

00:00:00

Knife Selected □ Tool Up □ Feeding Material □
Pen Selected □ Tool Down □

Figure 87. CNC generation and support for various cutting, drawing and printing options

8.3.2. ZundAutonest

ZundAutonest is a branded and specialised version of AptiaNest specifically designed for use on the Zund range of cutting machines. This software has been developed in partnership with Zund UK, the main Zund cutting machine reseller in the UK.

ZundAutonest software is widely used in the composites cutting field in the UK, notably by Formula One teams and large aerospace manufacturers. Worldwide the ZundAutonest product is also distributed by Zund territorial resellers.

Since Aptia first approached Zund UK with the idea of producing a specialised version of the AptiaNest solution ZundAutonest has replaced several other competitive solutions in the portfolio of products that Zund UK once sold. Furthermore Zund UK and Aptia now often collaborate on various special projects and products in order to improve the end-user experience and the range of functionality the cutting machines can engage in. In the near future it is expected that these projects will begin to include products beyond those with automatic nesting as their key component.

8.3.3. MyNesting.com

This product is a mould breaking attempt to give low cost access to the benefits of automatic nesting to users who would not normally be willing to pay for expensive automatic layout software.

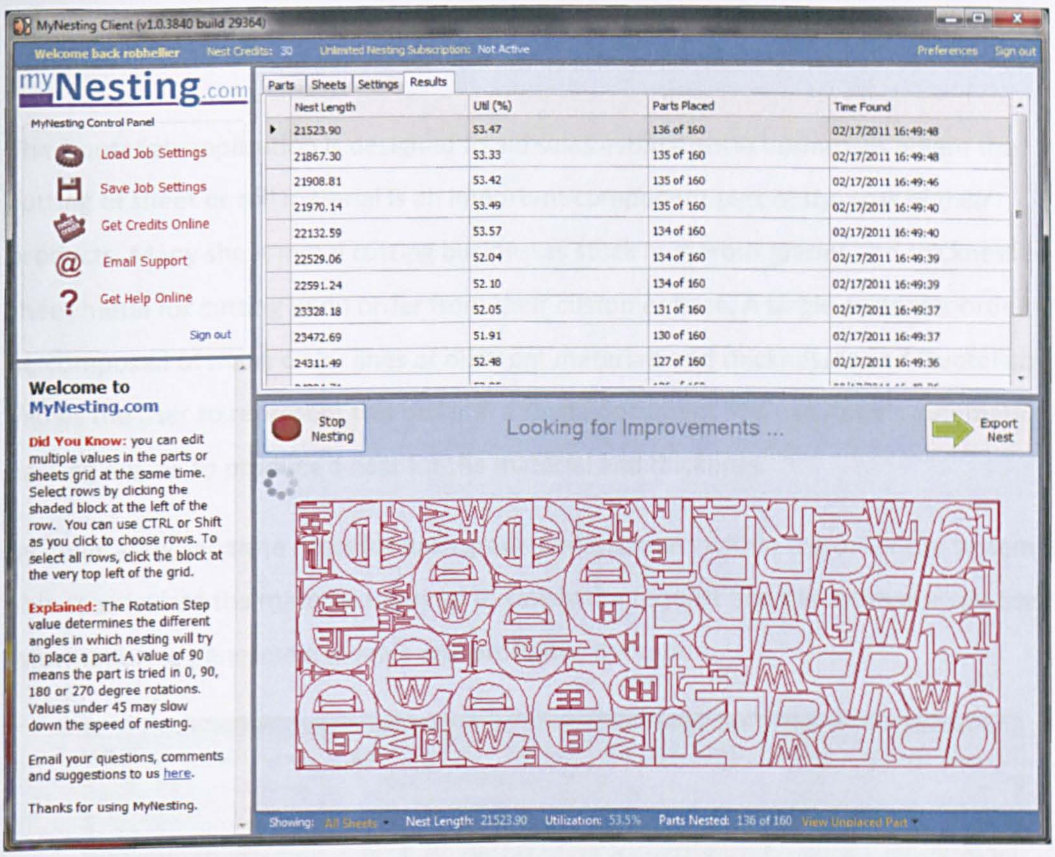
Often small cutting outfits have CAD/CAM software to run their cutting tables but do not have the budget or throughput that can justify expensive suites of software that include advanced features such as automatic nesting. The MyNesting.com approach is to provide the software for the generation of layouts to the end user without charging an upfront fee.

In order to use the MyNesting.com software the user is only required to sign up via the website and download the free software. The user can import their parts via the DXF file format and generate layouts using various options and rotational constraints, on any shape of stock sheet or sheets, until they are happy with the resultant layout. Similarly to the AptiaNest product MyNesting uses the on-going search approach to find very high quality layouts, as shown in Figure 88.

When the user is satisfied with the generated nesting and requires the layout to be exported from the software, in order to pass it a CAM system for cutting, they use a Nest Credit which can be bought through the MyNesting.com website (http://www.mynesting.com). The layout can be exported to the DXF format, illustrated in Figure 89, which allows maximum compatibility with most CAM products on the market.

Uniquely, of the few online nesting solutions available, MyNesting uses a fat client application and utilises the users CPU and memory resources to generate the solutions. This has proven to be a popular aspect of the software as end users can be wary of letting their confidential part designs leave their own computers/premises; all other solutions require that the parts are uploaded to a web server farm for nesting. The only time that MyNesting requires an internet connection, or contact with the MyNesting web server, is to log the user into the application and to use a nest credit. The MyNesting website is used to purchase credits independently of the client application.

To date there have been over 5700 unique user who have signed up to this service and numerous related enquiries and opportunities have arisen due to its popularity and reach in the sector. Indeed several software development companies have already integrated MyNesting into their software products.



	Nest Length	Util (%)	Parts Placed	Time Found
▸	21523.90	53.47	136 of 160	02/17/2011 16:49:48
	21867.30	53.33	135 of 160	02/17/2011 16:49:48
	21908.81	53.42	135 of 160	02/17/2011 16:49:46
	21970.14	53.49	135 of 160	02/17/2011 16:49:40
	22132.59	53.57	134 of 160	02/17/2011 16:49:40
	22529.06	52.04	134 of 160	02/17/2011 16:49:39
	22591.24	52.10	134 of 160	02/17/2011 16:49:39
	23328.18	52.05	131 of 160	02/17/2011 16:49:37
	23472.69	51.91	130 of 160	02/17/2011 16:49:37
	24311.49	52.48	127 of 160	02/17/2011 16:49:36

Welcome to MyNesting.com

Did You Know: you can edit multiple values in the parts or sheets grid at the same time. Select rows by clicking the shaded block at the left of the row. You can use CTRL or Shift as you click to choose rows. To select all rows, click the block at the very top left of the grid.

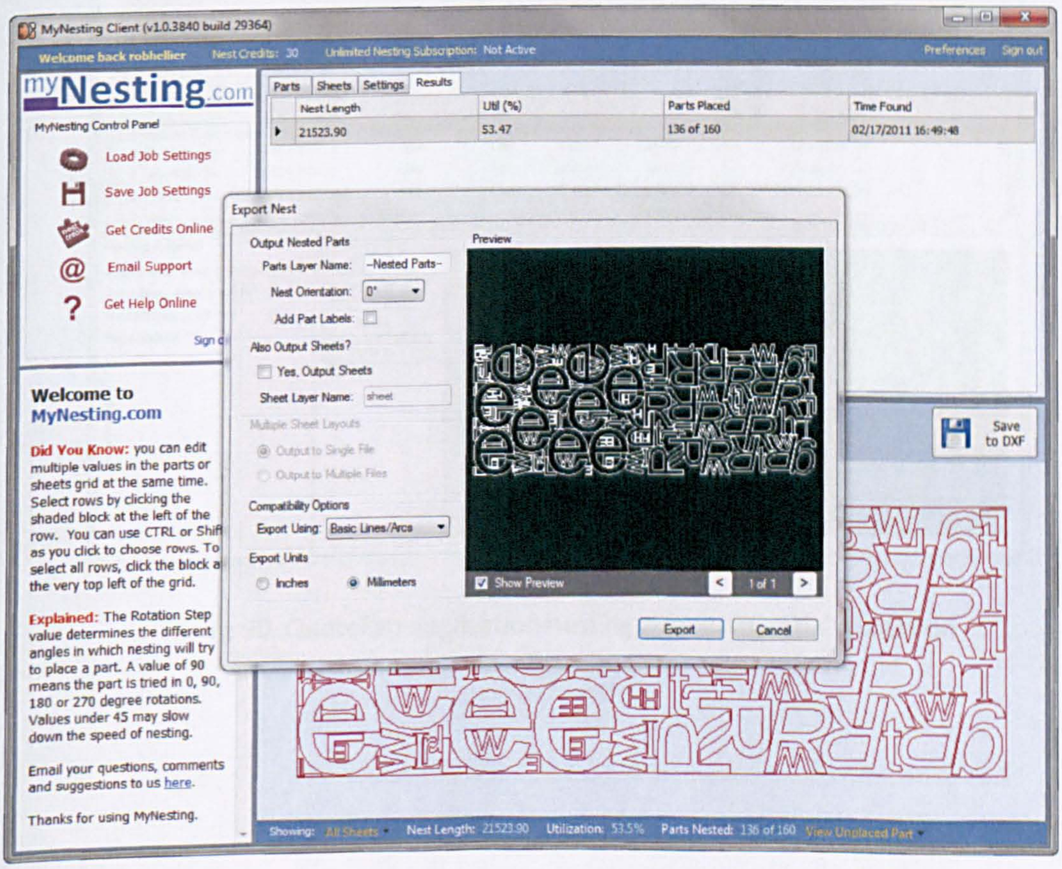
Explained: The Rotation Step value determines the different angles in which nesting will try to place a part. A value of 90 means the part is tried in 0, 90, 180 or 270 degree rotations. Values under 45 may slow down the speed of nesting.

Email your questions, comments and suggestions to us here.

Thanks for using MyNesting.

Showing: All Sheets · Nest Length: 21523.90 Utilization: 53.5% Parts Nested: 136 of 160 View Unplaced Part

Figure 88. A nesting of Profiles9 parts in the MyNesting client application



	Nest Length	Util (%)	Parts Placed	Time Found
▸	21523.90	53.47	136 of 160	02/17/2011 16:49:48

Export Nest

Output Nested Parts

Parts Layer Name: –Nested Parts–

Nest Orientation: 0°

Add Part Labels: □

Also Output Sheets?

□ Yes, Output Sheets

Sheet Layer Name: sheet

Multiple Sheet Layouts

● Output to Single File

○ Output to Multiple Files

Compatibility Options

Export Using: Basic Lines/Arcs

Export Units

○ Inches ● Milimeters

Welcome to MyNesting.com

Did You Know: you can edit multiple values in the parts or sheets grid at the same time. Select rows by clicking the shaded block at the left of the row. You can use CTRL or Shift as you click to choose rows. To select all rows, click the block at the very top left of the grid.

Explained: The Rotation Step value determines the different angles in which nesting will try to place a part. A value of 90 means the part is tried in 0, 90, 180 or 270 degree rotations. Values under 45 may slow down the speed of nesting.

Email your questions, comments and suggestions to us here.

Thanks for using MyNesting.

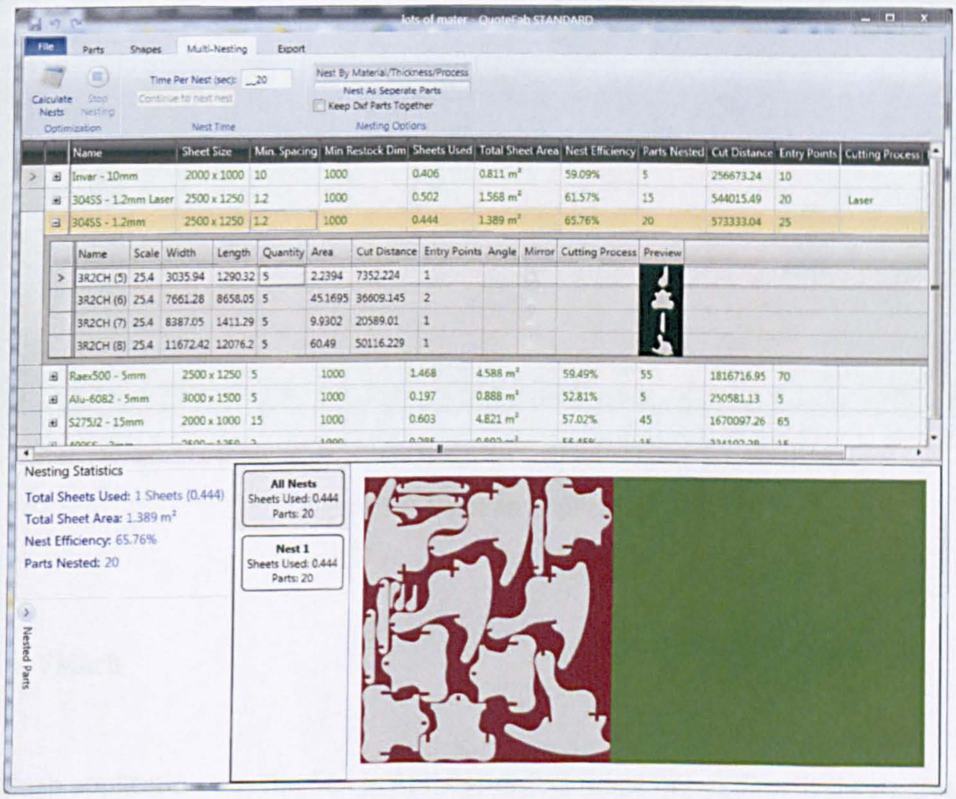
Showing: All Sheets · Nest Length: 21523.90 Utilization: 53.5% Parts Nested: 136 of 160 View Unplaced Part

Figure 89. Exporting a purchased layout in the MyNesting client application

8.3.4. QuoteFab

The QuoteFab application is designed to aid sales estimators in businesses where the cutting of sheet or roll material is an important component part of the cost of their products. Many sheet metal cutting businesses stock numerous grades and thicknesses of sheet metal for cutting upon order from their customer base. A single customer order can be composed of many order lines of different materials and thicknesses and QuoteFab allows the user to represent this order in a single document and use Aptia's automatic nesting engine to produce a nest for the material and thickness.

By generating separate nests for each grade and thickness within the order the system is able to calculate the material required to satisfy the request and the estimator can use this information to generate a reliable and profitable quotation.



Time Per Nest (sec): __20

□ Keep Dxf Parts Together

Name	Sheet Size	Min. Spacing	Min Restock Dim	Sheets Used	Total Sheet Area	Nest Efficiency	Parts Nested	Cut Distance	Entry Points	Cutting Process
Invar - 10mm	2000 x 1000	10	1000	0.406	0.811 m²	59.09%	5	256673.24	10	
304SS - 1.2mm Laser	2500 x 1250	1.2	1000	0.502	1.568 m²	61.57%	15	544015.49	20	Laser
304SS - 1.2mm	2500 x 1250	1.2	1000	0.444	1.389 m²	65.76%	20	573333.04	25	

Name	Scale	Width	Length	Quantity	Area	Cut Distance	Entry Points	Angle	Mirror	Cutting Process	Preview
3R2CH (5)	25.4	3035.94	1290.32	5	2.2394	7352.224	1				
3R2CH (6)	25.4	7661.28	8658.05	5	45.1695	36609.145	2				
3R2CH (7)	25.4	8387.05	1411.29	5	9.9302	20589.01	1				
3R2CH (8)	25.4	11672.42	12076.2	5	60.49	50116.229	1				

Name	Sheet Size	Min. Spacing	Min Restock Dim	Sheets Used	Total Sheet Area	Nest Efficiency	Parts Nested	Cut Distance	Entry Points	Cutting Process
Raex500 - 5mm	2500 x 1250	5	1000	1.468	4.588 m²	59.49%	55	1816716.95	70	
Alu-6082 - 5mm	3000 x 1500	5	1000	0.197	0.888 m²	52.81%	5	250581.13	5	
S275J2 - 15mm	2000 x 1000	15	1000	0.603	4.821 m²	57.02%	45	1670097.26	65	

Nesting Statistics

Total Sheets Used: 1 Sheets (0.444)

Total Sheet Area: 1.389 m²

Nest Efficiency: 65.76%

Parts Nested: 20

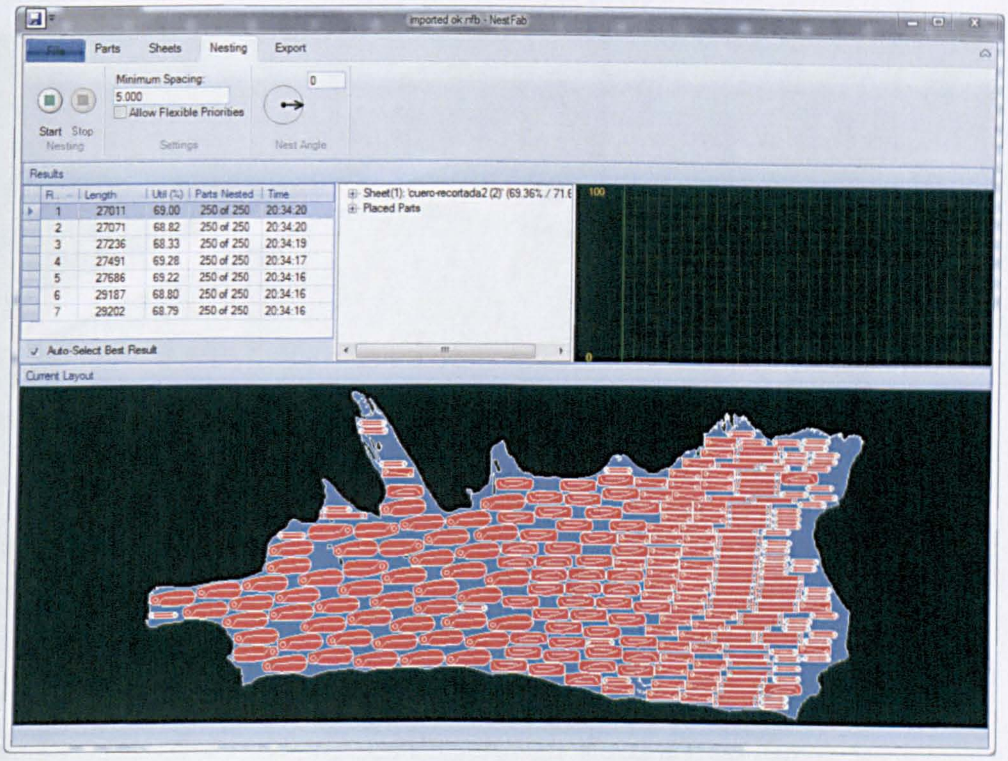
All Nests
Sheets Used: 0.444
Parts: 20

Nest 1
Sheets Used: 0.444
Parts: 20

Figure 90. QuoteFab application nesting a multi material quotation

8.3.5. NestFab

NestFab is a simple to use and inexpensive DXF in DXF out buy once application that performs similar functionality to the successful MyNesting product. The inclusion of additional third party integration options, reporting and other additional valuable features has made this product an attractive option for machine vendors and software vendors to include as an optional add on to their own offerings.



Minimum Spacing: 5.000

□ Allow Flexible Priorities

Nest Angle: 0

R...	Length	Util (%)	Parts Nested	Time
1	27011	69.00	250 of 250	20:34:20
2	27071	68.82	250 of 250	20:34:20
3	27236	68.33	250 of 250	20:34:19
4	27491	69.28	250 of 250	20:34:17
5	27686	69.22	250 of 250	20:34:16
6	29187	68.80	250 of 250	20:34:16
7	29202	68.79	250 of 250	20:34:16

✓ Auto-Select Best Result

Sheet(1): 'cuero-recortada2 (2)' (69.36% / 71.6

Placed Parts

Figure 91. NestFab application

8.3.6. VMach

The VMach application was the first in Aptia's product range not to directly incorporate nesting technology. The system is a flexible 3D simulation of cutting machines for machine reseller marketing and demonstration purposes. The system interprets the CNC code that an Aptia product (or any other product) has generated and shows how the cutting machine will perform the cuts, other functions and material feeds (shown in Figure 92). This product is useful in ensuring the cutting behaviour of a CNC file will be as desired and has been used in numerous instances at trade shows in place of a physical cutting machine. Various machine resellers who have adopted Aptia's product range have requested bespoke models of this application as it has proven to be a useful sales tool.

This application is also of interest to end users and is likely to be offered in the near future as a machine monitoring application. The enhanced VMach will maintain synchronisation with a physical cutting machine, warning of problems and reporting on progress. This product could allow for real time synchronised simulation and supervision of a cutting machine from anywhere in the world via internet based web services.

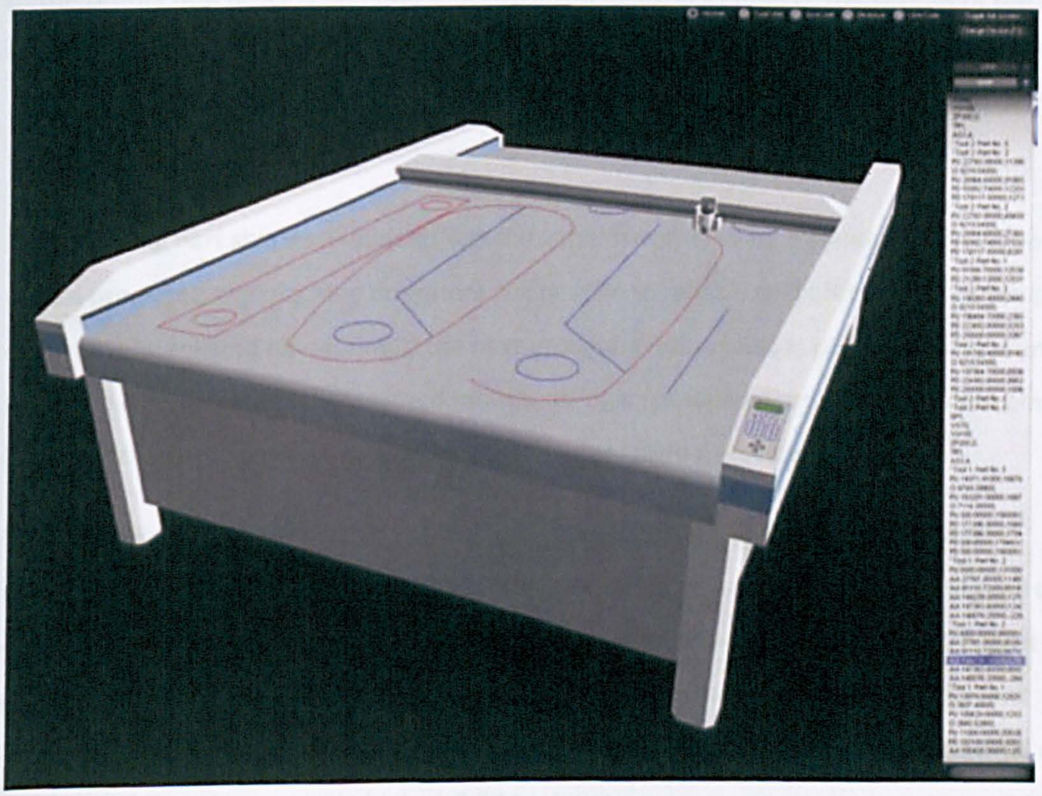


Figure 92. VMach application simulating a Zund Machine cutting HPGL instructions

8.3.7. LightSpeed

Aptia and Zund UK are currently working in partnership with the UK arm of a large aerospace manufacturer which specialise in the production of nacelles (engine covers for jet engines) particularly for the Airbus group.

As part of the on-going relationship with Zund UK, Aptia have been commissioned to work on the development of a part projection system for use on the Zund cutting machines which are a vital part of the aerospace company's manufacturing process.

The installation of the ZundAutonest branded Aptia software in 2009 has already saved the aerospace manufacturer in the region of 5% of their £11,000,000 carbon fibre material budget per annum.

The projection system, called LightSpeed, will allow various jobs in the factory to be mixed together in the nesting process which experiments have shown will provide further savings. Currently only one kit of parts can be nested to a sheet at a time as part identification is too difficult due to the similar nature of parts from various nacelle kits. This causes a bottleneck in the production process and does not allow the fullest potential of the savings of the nesting system to be realised. It has been estimated, by the manufacturer, that further material savings achieved using the mixed nacelle kit approach could be in the region of a further 3% to 4% per annum.

By projecting part shapes and identifiers onto the material, using a standard high power digital projector, the machine operators will be able to reliably and efficiently pick the parts out from the roll and take them to the next process, which involves placing and adhering the cut parts onto specially designed moulds, using a specialised resin glue, before being baked into a single piece in large pressurised ovens called Autoclaves.

The identification of kits is achieved by highlighting parts using fill colour and letters projected onto the picking area of the cutting machines, as each frame feed of the roll of cut carbon fibre moves onto a run off table of the cutting machine. As various different grades of carbon fibre used by in the aerospace parts are manufactured with different coloured backing material the kit projection colours will be chosen from a per material user defined palette of compatible high visibility colours.

9. Conclusions

The following sections outline the contributions to the state-of-the-art in this thesis.

9.1. A New Arc Compatible Placement Heuristic for Bottom-Left Fill Placement Strategy

The introduction of the new bottom-left fill algorithm, that utilises a vertical overlap resolution technique which allowed the generation of results that were a major improvement on those presented previously in the literature on 25 of the 26 literature benchmark instances. Furthermore the methodology allows for fast overlap resolution when both arcs and holes are present in the problem instance. This is the first time that any algorithm published in the literature has been able to deal with these specific features. The benchmark improvements are on average 5% better than the previous benchmarks at the point of publication.

9.2. Robust No-Fit Polygon Generation

This thesis (see chapter 5) presented the first complete and robust no-fit polygon generation algorithm (based on the sliding technique) which is able to cope with all known degenerate cases. The use of no-fit polygons in the arena of irregular two-dimensional layout generation has become a major theme of the research community since 2000. The introduction of a simple, efficient and robust technique for generating no-fit polygons, which does not suffer from the known degenerate cases, is a valuable contribution to the state of the art. No-fit polygon generation times for this approach were shown to be competitive with other approaches reported in the literature.

9.3. Line-Arc No Fit Polygon Generation

In addition to overcoming the known degenerate cases in the sliding no-fit polygon generation technique, this thesis has extended the technique to allow the generation of no-fit polygons including lines and arcs. This allows practitioners to tackle many more real world problems where arcs are common and their accurate representation is a major advantage. In combination with the new layout technique introduced in chapter 4 the no-fit polygon can be used to dramatically speed up overlap detection and resolution. This speed up allows for the generation of many more solutions per second ensuring that the search heuristics are able to explore more of the solution space, increasing the possibility of discovering high quality solutions quickly. The results presented in chapter 6 show that the approach is able to generate high quality results for existing and new benchmarks in durations which are significantly less than other competitive literature techniques. The speed of solution generation was a significant priority for the industrial partner in the CASE and TCS projects that motivated the research of this thesis.

9.4. Generation of new benchmark problems

In order to utilise the unique capabilities of the techniques presented in chapters 4 and 6 it was necessary to develop new benchmark problems containing features such as holes and arcs, some drawn directly from the libraries of the industrial partner in the CASE and TCS projects. These benchmarks will allow future researchers and practitioners to test the effectiveness of their techniques. These benchmarks have been published on several occasions and are described in full detail in the appendices of the thesis.

9.5. Generation of new best results for numerous literature benchmark problems

Over the course of this research the techniques generated have produced the best known results, at the time of generation, for the full range of literature benchmark problems. Upon publication the techniques described in chapters 4, 5 and 6 introduced new benchmark results and new problem instances to the literature. The benchmark problems have been drawn from over 25 years of research and new instances introduced from industrial partners. Furthermore the majority of the best known solutions generated by this thesis have been generated in less than five minutes of computational time, which is a significant improvement on the generation times of other approaches reported in the scientific literature. The speed of quality solution generation makes the research presented here highly relevant to industrial practitioners, where nesting is of relevance.

9.6. Experimentation with non-greedy placement in a bottom-left fill algorithm

Chapter 7 presents a simple non-greedy placement technique that improves on results generated in earlier chapters of the thesis; indeed it solves one of the new benchmark problems to optimality. This implementation is the only reference in the scientific literature to non-greedy placements being selected during a bottom-left fill placement algorithm.

Additionally the work in chapter 7 presents the first results generated for problems including holes and circular arc segments using a TOPOS placement strategy. Furthermore analysis of average results confirms that this method is useful on a broad range of problem types and in particular industrial style problems. Additional analysis identifies a range of values that control the technique appear to be productive settings for the approach across the wide range of problems tested.

CHAPTER NINE

10. Future Work

As discussed in chapter 8 the research developed in the course of the CASE and TCS schemes, which forms the work detailed in this thesis, has been successfully used in various commercial settings and products. Indeed the author is now directly involved with a spin-out company which was conceived as a vehicle for commercialisation of the techniques presented herein. With this in mind the continuation of this research is inspired by industrial considerations and trends.

10.1. Production Scheduling and Nesting

The nesting and cutting of shapes from flat materials in industrial contexts never occurs without a production schedule or time sensitive production requirement, for example (Chryssolouris et al., 2000) and (Morabito & Arenales, 2000) have both approached the nesting problem with the production requirements and schedules as an aspect of their solution methodologies.

In unpredictable environments, for example in steel profiling centres (which offer to cut steel profiles of varying thicknesses and grades to order), pressing deadlines and mismatches in ordered part thicknesses can require a complex balancing act been nesting efficiency and timely delivery.

A future direction of research will be to develop a constantly running nesting system which will respond to a production schedule, changing priorities as new orders arrive.

10.2. Multiple Complementary Nesting Approaches in Parallel

The majority of nesting approaches in the scientific literature use a single approach to generate layouts, as is the case with the techniques introduced in this thesis. A minority of approaches use a bottom-left fill technique to produce an initial layout which is then compacted using some method such as facilitated by phi-functions or linear compaction, for example (Egeblad et al., 2007) and (Chernov et al., 2010). With the increasing trend towards multiple core processors, in even the most basic computers, it is now possible for multiple techniques to run simultaneously on a problem instance.

If sets of algorithms, such as differently seeded tabu search or hill-climbing controlled bottom-left fill approaches (as presented in this thesis) or mixed approaches like those in Egeblad and Chernov were run under the control of a Hyper-heuristic techniques ((Burke et al., 2003), (Burke et al., 2005)) there could be potential for sharing information and improved solution space coverage. If information such as best result and good shape clusters were recorded and shared in a commonly available repository any applied nesting approach, and the hyper-heuristic, would be able to take advantage of the information produced by other techniques. Furthermore opportunities for aborting less successful lines of exploration early would become possible allowing for additional processing efficiency.

Additionally as the no-fit polygon is widely regarded, and widely utilised, as one of the most efficient methods for overlap detection and resolution all algorithm instances would be able to utilise the shared pool of generated no-fit polygons.

10.3. Cutting Process Considerations Influencing Nesting

The scientific literature contains few references to considerations of the cutting of packed shapes. Various materials have marked differences in the potential cutting technologies that may be applied and each technology can require specific approaches in order to achieve the required cut quality or speed. Indeed it is sometimes necessary when using cutting methods such as plasma, gas, laser and knife cutting to adjust layouts and cutting geometry after nesting in order to achieve the required cut compatibility. This could have an effect on nesting quality. Future research will focus on introducing the additional considerations of material, cutting technology and best practise into the nesting process in order to produce high quality layouts that have desirable cutting considerations built into them.

10.4. Nesting in the Cloud

The emergence of cloud computing platforms (Armbrust et al., 2010) and widely distributed clusters of internet connected computers, for example the Folding@Home and Genome@Home projects (Larson et al., 2002), is a potentially useful area of exploration for complex problems such as cutting and packing. By vastly increasing the processing power available to tackle this family of problems it may be possible to co-ordinate and communicate with a large number of problem solving nodes in a widely distributed network. Once again there is potential for hyper-heuristics to coordinate the search node's activities both, as discussed previously, at the in-node level and at the macro level across multiple nodes (Ouelhadj & Petrovic, 2010).

13.5. Cutting Technologies and Material Types

The cutting technology and the physical properties of the material a layout is to be cut from can have an impact on the manner in which a layout is generated, guillotine cutting for instance impacts on the admissibility of a layout.

Throughout the work of this thesis the applicability to any particular cutting technology or material that the generated layouts may be cut from has been largely ignored, mainly as the plasma cutting technology (and the Procut application) of the industrial partner, Esprit Automation, could cope with the cutting aspects of almost any generated layout.

The following section aims to give an overview of cutting technology and material specific properties that may be factored into layout techniques as part of future research. These aspects of cutting and materials have been encountered during the authors' eleven years of industrial experience in the field of cutting and packing across numerous industries.

13.5.1. Knife Cutting

Unlike most other cutting techniques once an incision has been made a cutting knife has limited range of movement. This limitation requires that in order to cut a corner of a shape beyond a certain angle the knife must lift out of the material, rotate and drop back into the material. In combination with the shape of the cutting blade and the thickness of the material this can produce an overcut as shown in Figure 93a.

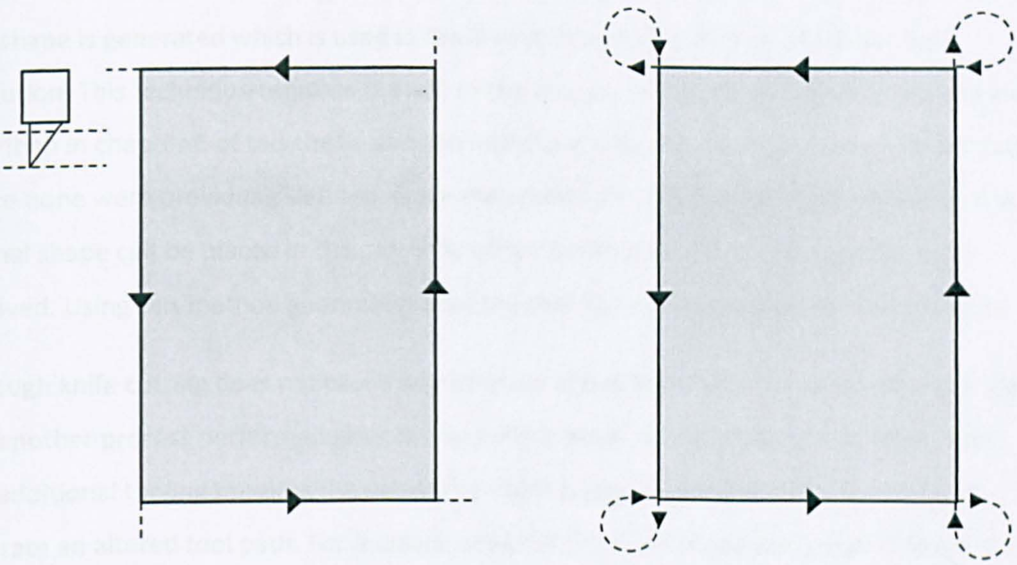


Figure 93. a) Overcuts in knife cutting. b) Mouse Ears / Butterfly cuts

The additional overcut geometry can be dealt with in two ways; either the layout should be produced with the overcuts as part of the nesting geometry or a calculation of any potential overcut can be used to determine a minimum spacing between all parts and the sheet edge, each approach presents its own problems.

The inclusion of overcut geometry can complicate the geometry library required to calculate no-fit polygons and/or overlap resolution techniques as many geometric techniques require that the polygons that construct a shape are continuous non crossing loops. Overcutting geometry is likely to break a geometric routine's pattern but if it can be supported then additional efficiency maybe achieved over a higher spacing model. It should be noted that in almost every cutting process, even in knife cutting where wasting of material around cuts is not a factor, it is desirable to have some spacing between nested parts to avoid the possibility of one part being cut, even minutely, into another.

By ensuring that the minimum possible spacing between shapes is enough to cope with any overcuts that may be required it is possible to ignore the overcut geometry entirely. However this approach will obviously lead to potentially unnecessary spaces and therefore inefficiency in the generated layouts.

Where spacing is required in industrial settings, see chapter 8 for industrial applications of the research of this thesis, the approach used in this work is to derive a no-fit polygon of each shape in the layout with a circle of a radius of the required spacing.

By tracking the circle's centre point during the no-fit polygon generation (see Figure 94a) a new shape is generated which is used in the layout process for overlap detection and resolution. This technique requires the use of the line arc no-fit polygon generation routine described in chapter 6 of this thesis and can introduce arcs into the shapes used for nesting where none were previously defined. Once the spaced shape has been used for layout the original shape can be placed in the correctly offset position and its spaced counterpart removed. Using this method guarantees that the minimum distance rule will be respected.

Although knife cutting does not cause any kerf loss of the material it is occasionally the case that another process performed prior to the cutting stage may produce a kerf like effect. This additional tooling requires the use of the no-fit polygon based spacing approach to generate an altered tool path. For instance where some form of pre-sealing of a material is require before cutting (see Figure 94b) an inner no-fit polygon is generated for a sealing tool to follow as its sealing path. Once again the width of the seal is a factor in determining the radius of the circle used to generate the sealing path. Once the seal is made the outside path of the shape will cut the outer edge of the seal.

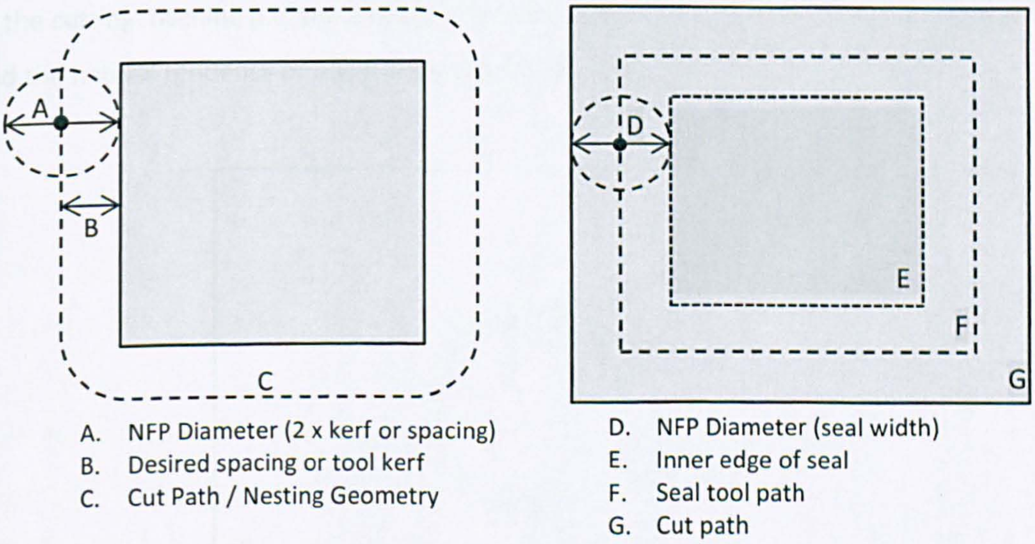

A
B
C
D
E
F
G
A. NFP Diameter (2 x kerf or spacing)
B. Desired spacing or tool kerf
C. Cut Path / Nesting Geometry
D. NFP Diameter (seal width)
E. Inner edge of seal
F. Seal tool path
G. Cut path

Figure 94. a) Kerf Compensation / Spacing b) Inner NFP for Sealing

13.5.2. Plasma Cutting

The process of plasma cutting is powered by an electric arc being passed through an inert gas which forms a jet of plasma which is sufficiently hot to melt and vaporise metal of various thicknesses. This cutting process produces a kerf of some width which is dependent upon various factors such as voltage, gases used, nozzle wear, cut speed, material type and material thickness.

When cutting using plasma spacing between parts is an absolute requirement in both the shape layout process and cutting path generation (see Figure 94a). Additionally due to the initiation sequence for piercing sheet metal it is important to start cuts away from the actual cut path of the shape, as piercing causes additional damage to the metal plate due to the power required to punch through the plate and the initially shallow nature of the cut. To ensure that the part being cut does not have the damaged piece of plate as part of its cut outline it is usual to use additional cut lines into and out of the cut path of the part (see Figure 95) called lead-ins and lead-outs. These additional cut lines are angled and shaped on a part by part basis to ensure that the cut quality of the part is optimised.

Additional cut geometry may also be used at corners of parts (i.e. 90 degree corners) when very high precision cuts are required. These cuts are called Mouse Ear or Butterfly cuts due to their shape (see Figure 93b). Without these additional cut features corners of plasma cut shapes will tend to be rounded, especially on thicker materials, due to mechanical features of the cutting machine (i.e. the ability of the cutting machine to make sharp turns at speed) and the natural tendency of the plasma stream to lag behind the cutting head.

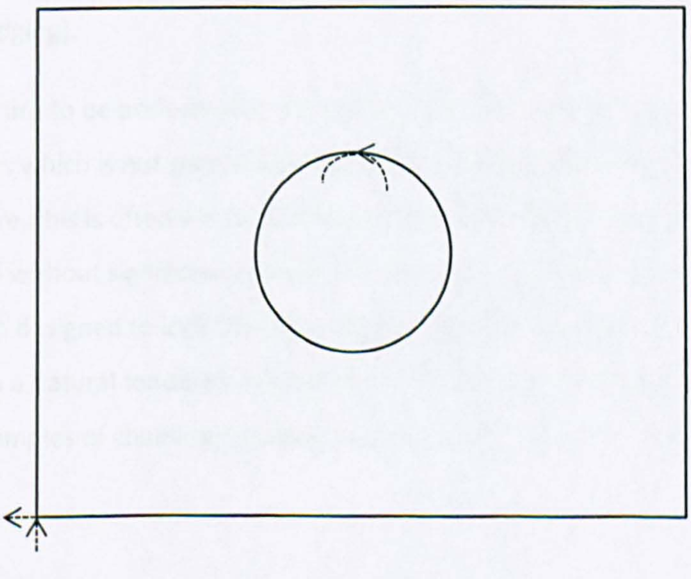


Figure 95. Examples of lead in/out placements and shapes

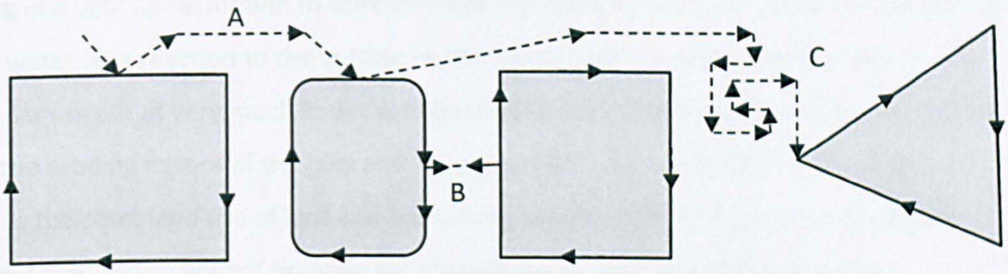
13.5.3. Flame Cutting

Like plasma cutting oxy-fuelled flame cutting is a destructive process which has similar kerf, lead-in/out and spacing requirements although generally the kerf and spacing requirements are for higher values as the process is less precise than that of plasma cutting. Gas cutting technology is generally used for thicker sheet metal cutting than plasma cutting although there is a range where their capabilities overlap.

Due to the slow cutting process and higher thicknesses of the materials being cut using this technology heating of the plate being cut and the associated problems of warping and expansion are of more significance. Once again lead-in/outs are used to avoid damaging the cut path of a part but the spacing requirements (and kerf values) are often higher as the piercing and cutting process is even more damaging that in the plasma case. Indeed for extremely thick materials it may be necessary to pre-pierce the plate before cutting using a manual process called thermal lancing.

Expansion during cutting and difficultly of piercing means that lead-in and lead-outs may be designed specifically to help to compensate for these factors. In order to avoid the long and damaging pierces it is not uncommon for one part's lead-out to join up with the next part's lead-in (chaining) or one part may be interrupted during its cut path to begin cutting another part (bridging).

If these activities are to be performed it will require that the parts are placed with respect to these activities which is not part of any automatic nesting algorithm in the current scientific literature, this is often why automatic nesting is not often used by those cutting thick sheet metal without significant manual intervention. Further to chaining and bridging leads, bridges can designed to lock the plate together to halt expansion around the point of cut (the plate has a natural tendency to peel apart like a zip behind the point of intense cutting heat). Examples of chaining, bridging and lead lock shaping are shown in Figure 96.



A. Chain link from Shape1 to Shape2
B. Bridging of Shape2 and Shape3
C. Lock shaped lead

Figure 96. Chaining, Bridging and Lead Lock Shaping

13.5.4. Laser Cutting

Laser cutting is the one of the fastest and most versatile cutting technologies available and is used across various industries and therefore many different materials. Like plasma and gas cutting the laser process burns away a portion of the material which means that kerf compensation, lead-in/outs and spacing approaches are also required.

With the additional speed of the laser cutting technology, especially for thinner sheet metal, careful travel path planning (the movement of the head between parts) is required as the head is often designed to traverse the plate whilst at its cutting height which means that if a piece of cut material has tilted out the plane of the sheet the head may hit it and be damaged. To avoid this, cut order, lead-in/out positions and travel path amendments may be made to avoid passing over any already cut parts.

13.5.5. Water Jet Cutting

Water jet cutting is another versatile cutting process used in some unique application areas as it does not require the heating of the material in order to make cuts, this is useful in materials such as stone which have very high vaporisation temperatures. A mixture of water and an abrasive material is passed through a nozzle at a very high pressure allowing the resultant jet to quickly erode the material being cut. Factors particular to this cutting technology include the requirement for specific kinds of piercing routines, some include moving in a tight spiral motion to bore through the material being cut, and the slow speed of the water jet's reaction to the cutting head's movement requiring that corners in particular are cut at very much lower speeds than straight edges or large radius arcs. Once again the eroding nature of the cuts and the propensity towards rounding of corners requires the combined use of kerf compensation, spacing between parts, lead-ins/outs, butterfly cuts and chaining / bridging techniques due to the slowness of piercing.

13.5.6. Further Industrial Considerations

In various technologies industrial practitioners often have to consider additional objectives beyond optimisation of layout generation. The following subsections describe some of the additional objectives encountered whilst working with the industrial partners of this research.

13.5.6.1. Cut Start Positions

Positioning the start/finish point of cuts can have a significant impact on the cut quality, traverse distance and therefore cycle time of cutting a layout. In metal cutting there is a relationship between the shape of lead in/outs and the position of their entry point on the cutting geometry, additionally several rules of thumb exist for the position of leads with respect to cut order and where the majority of the uncut metal is present (i.e. it is better to make the last part of a cut through the most stable part of the sheet). Optimising the position of lead in/outs and the order of cutting is a complex problem in its own right and maybe an interesting area for future study.

13.5.6.2. Multiple sheet sizes

When there exist multiple different possible sheet sizes and offcuts for a nesting job it is useful for a nesting system to be able to determine which size of sheet or sheets are most appropriate for the shapes to be nested onto. Choosing the correct size of sheet or sheets can significantly reduce the potential waste produced or allow the manufacturer to buy in stock in sizes amenable to the work.

13.5.6.3. Multiple Head Cutting

Particularly in gas and plasma cutting technologies it is not unusual for machines to have multiple cutting heads fitted. Additionally some machines may be able to adjust the spacing between the heads automatically or may require manual adjustment in order to alter head spacing. When nesting a sheet for cutting with multiple heads the nesting algorithm can be required to determine the optimum head spacing, minimise the head spacing changes or switch between different numbers of heads being used for cutting. This adds additional constraints and problems to the generation of viable and efficient layouts.

13.5.6.4. Offcut Management and Utilisation

The reuse of partially cut sheets is a common requirement in industrial settings when material sizes cannot be predetermined in order to ensure full utilisation for a nest. This is common in steel profiling shops where material values are high and the orders the cutter receives are from numerous customers. In these circumstances the accurate tracking of offcuts and the ability to nest onto irregular shaped sheets is vital.

References

Adamowicz, M. & Albano, A., 1976a. A solution to the rectangular cutting stock problem. *IEEE Transactions on Systems Man and Cybernetics*, 6(4), pp.302-10.

Adamowicz, M. & Albano, A., 1976b. Nesting two-dimensional shapes in rectangular modules. *Computer Aided Design*, 8, pp.27-33.

Agarwal, P.K., Flato, E. & Halperin, D., 2002. Polygon Decomposition for Efficient Construction of Minkowski Sums. *Computational Geometry: Theory and Applications*, 21, pp.39-61.

Albano, A., 1977. A Method to Improve Two-Dimensional Layout. *Computer Aided Design*, 9, pp.48-52.

Albano, A. & Sapuppo, G., 1980. Optimal Allocation of Two-Dimensional Shapes Using Heuristic Search Methods. *IEEE Transactions on Systems, Man and Cybernetics*, 10(5), pp.242-48.

Amenta, N., 1997. Computational geometry software. In *Handbook of Discrete and Computational Geometry*. Boca Raton: CRC Press LLC. Ch. 52. pp.951-60.

Anand, S., McCord, C. & Sharma, R., 1999. An Integrated Machine Vision Based System for Solving the Non-Convex Cutting Stock Problem Using Genetic Algorithms. *Journal of Manufacturing Systems*, 18(6), pp.396-415.

Armbrust, M. et al., 2010. A View of Cloud Computing. *Communications of the ACM*, 53(4), pp.50-58.

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

Avnaim, F. & Boissonnat, J.D., 1987. Simulataneous containment of several polygons. In *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry.*, 1987.

Baker, B.S., Coffman, E.G. & Rivest, R.L., 1980. Orthoginal Packings in Two Dimensions. *SIAM Journal on Computing*, 9, pp.846-55.

Barnett, S. & Kynch, G.J., 1967. Exact Solution of a Simple Cutting Stock Problem. *Operations Research*, 15, pp.1051-56.

Bennell, J.A. & Dowsland, K.A., 1999. A tabu threasholding implementation for the irregular stock cutting problem. *International Journal of Production Research*, 37(18), pp.4259-75.

Bennell, J.A. & Dowsland, K.A., 2001. Hybridising Tabu Search with Optimisation Techniques for Irregular Stock Cutting. *Management Science*, 47(8), pp.1160-72.

Bennell, J.A., Dowsland, K.A. & Dowsland, W.B., 2001. The irregular cutting-stock problem - a new proceedure for deriving the no-fit polygon. *Computers and Operations Research*, 28, pp.271-87.

Bennell, J.A. & Oliveira, J.F., 2008. The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184, pp.397-415.

Bennell, J., Scheithauer, G., Stoyan, Y. & Romanova, T., 2008a. Tools of mathematical modeling of arbitrary object packing problems. *Annals of Operations Research*, 179(1), pp.343-68.

Bennell, J.A. & Song, X., 2008. A comprehensive and robust procedure for obtaining the no-fit polygon using Minkowski sums. *Computers and Operations Research*, 35, pp.267-81.

Bennell, J.A. & Song, X., 2010. A beam search implementation for the the irregular shape packing problem. *Journal of Heuristics*, 16(2), pp.167-88.

Blazewicz, J., Hawryluk, P. & Walkowiak, R., 1993. Using a Tabu Search Approach for solving the Two-Dimensional Irregular Cutting Problem. *Annals of Operations Research*, 41, pp.313-25.

Blazewicz, J. & Walkowiak, R., 1994. DSS for Two-Dimensional Cutting - New Tools Improving Feasible Solutions. *Foundations Of Computing and Decision Sciences*, 19 No. 4, pp.285-98.

Bounsaythip, C. & Maouche, S., 1997. Irregular shape nesting and placing with evolutionary approach. In *IEEE International Conference on Systems, Man and Cybernetics*. Orlando, USA, 1997.

Bremermann, H.J., 1958. Contract No. 477(17) *The Evolution of Intelligence: The Nervous System as a Model of its Environment*. Technical Report. University of Washington Seattle.

Burke, E.K.B., Hellier, R.S.R., Kendall, G. & Whitwell, G., 2006. A New Bottom-Left-Fill Heuristic Algorithm for the Two-Dimensional Irregular Packing Problem. *Operations Research*, 54(3), pp.587-601.

Burke, E.K., Hellier, R.S.R., Kendall, G. & Whitwell, G., 2007. Complete and Robust No-Fit Polygon Generation for the Irregular Stock Cutting Problem. *European Journal of Operations Research*, 179(1), pp.27-49.

Burke, E.K., Hellier, R.S.R., Kendall, G. & Whitwell, G., 2010. Irregular Packing Using the Line and Arc No-Fit Polygon. *Operations Research*, 58(4-Part-1), pp.948-70.

Burke, E.K. & Kendall, G., 1999. Applying Ant Algorithms and the No Fit Polygon to the Nesting Problem. In *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*. Sydney, Austrailia, 1999.

Burke, E.K. & Kendall, G., 2002. A New Approach to Packing Non-Convex Polygons Using the No Fit Polygon and Meta-Heuristic and Evolutionary Algorithms. In *Proceedings of Adaptaive Computing in Design and Manufacture V (ACDM 2002)*. University of Exeter, UK, 2002. Springer Verlag.

Burke, E.K., Kendall, G. & Soubeiga, E., 2003. A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*, 9(6), pp.451-70.

Burke, E.K., Landa Silva, J.D. & Soubeiga, E., 2005. Multi-objective Hyper-heuristic Approaches for Space Allocation and Timetabling. In T. Ibaraki, K. Nonobe & M. Yagiura, eds. *Meta-heuristics: Progress as Real Problem Solvers*. Springer.

Cheng, S.K. & Rao, K.P., 1997. Quick and precise clustering of arbitrarily shaped flat patterns based on stringy effect. *Computers & Industrial Engineering*, 33(3-4), pp.485-88.

Cheng, S.K. & Rao, K.P., 2000. Large-scale nesting of irregular patterns using a compact neighbourhood algorithm. *Journal of Materials Processing Technology*, 103(1), pp.135-40.

Chernov, N., Stoyan, Y. & Romanova, T., 2010. Mathematical model and efficient algorithms for object packing problem. *Computational Geometry: Theory and Applications*, 43(5), pp.535-53.

Chezelle, B. & Dobkin, D.P., 1985. Optimal comvex decompositions. In G.T. Toussaint, ed. *Computational Geometry*. Amsterdam: North-Holland. pp.63-133.

Chryssolouris, G., Papakostas, N. & Mourtzis, D., 2000. A decision-making approach for nesting scheduling: a textile case. *International Journal of Production Research*, 38(17), pp.4555-64.

Clay, P. & Crispin, A.J., 2001. Automated lay-planning in leather manufacturing. In *17th National Conference on Manufacturing Research*. Cardiff, 2001.

Crispin, A.J. et al., 2003. Genetic algorithms applied to lay plan material utilization. *Journal of Engineering Manufacture*, 217, Part B, pp.1753-56.

Cunninghame-Green, R., 1989. Geometry, Shoemaking and the Milk Tray Problem. *New Scientist*, 1677, pp.50-53.

Dagli, C.H., 1990. Neural network in manufacturing: possible impacts on cutting stock problem. In *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing.*, 1990. IEEE Computer Society Press.

Dean, H.T., Tu, Y. & Raffensperger, J.F., 2006. An improved method for calculating the no-fit polygon. *Computers and Operations Research*, 33, pp.1521-39.

Degraeve, Z., Gochet, W. & Jans, R., 2002. Alternative formulations for a layout problem in the fashion industry. *European Journal of Operations Research*, 143, pp.80-93.

Dickinson, J.F. & Knopf, G.K., 2000. A moment based metric for 2D and 3D packing. *European Journal of Operations Research*, 122, pp.133-44.

Dighe, R. & Jakiela, M.J., 1996. Solving Pattern Nesting Problems with Genetic Algorithms Employing Task Decomposition and Contact Detection. *Evolutionary Computation*, 3, pp.239-66.

Dori, D. & Ben-Bassat, M., 1984. Efficient Nesting of Congruent Convex Figures. *Communications of the ACM*, 27, pp.228-35.

Dorigo, M., Vittorio, M. & Alberto, C., 1996. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 26(1), pp.29-41.

Dowsland, K.A. & Dowsland, W.B., 1992. Packing Problems. *European Journal of Operations Research*, 56, pp.2-14.

Dowsland, K.A. & Dowsland, W.B., 1993. *Heuristic approaches to irregular cutting problems.* Technical Report, Working Paper (EBMS/1993/13). UC Swansea, UK.

Dowsland, K.A., Dowsland, W.B. & Bennell, J.A., 1998. Jostling for position - local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, 49(6), pp.647-58.

Dowsland, K.A., Vaid, S. & Downsland, W.B., 2002. An algorithms for polygon placement using a bottom-left strategy. *European Journal of Operations Research*, 141, pp.371-81.

Dyckhoff, H., 1990. A Typology of Cutting and Packing Problems. *European Journal of Operational Research*, 44, pp.125-59.

Egeblad, E., Nielsen, B.K. & Odgaard, A., 2007. Fast neighbourhood search for two and three-dimensional nesting problems. *European Journal of Operations Research*, 183, pp.1249-66.

Fogel, D.B. & Anderson, R.W., 2000. Revisiting Bremermann's Genetic Algorithm: I. Simultaneous Mutation of All Parameters. In *Proceedings of Congress of Evolutionary Computation 2000 (CEC 2000)*. La Jolla, California, USA, 2000.

Fogel, D.B. & Fraser, A.S., 2000. Running Races with Fraser's Recombination. In *Proceedings of Congress on Evolutionary Computation 2000 (CEC 2000)*. La Jolla, California, USA, 2000.

Foley, J., van Dam, A., Feiner, S. & Hughes, J., 2003. *Computer Graphics: Principles and Practice in C (International Edition)*. Addison Wesley.

Fowler, R.J., Paterson, M. & Tanimoto, S.L., 1981. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3), pp.133-37.

Fraser, A.S., 1957. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10, pp.492-99.

Fujita, K., Akagi, S. & HiroKawa, N., 1993. Hybrid approach for optimal nesting using a genetic algorithm and a local minimization algorithm. *Advances in Design Automation*, 65(1), pp.477-84.

Garey, M. & Johnson, D., 1979. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman and Company.

Ghosh, P., 1991. An algebra of polygons through the notion of negative shapes. *CVGIP: Image Understanding*, 51(1), pp.119-44.

Ghosh, P., 1993. A unified computational framework for minkowski operations. *Computers and Graphics*, 17(4), pp.357-78.

Gilmore, P.C. & Gomory, R.E., 1961. A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, 9, pp.849-59.

Gilmore, P.C. & Gomory, R.E., 1963. A linear programming approach to the cutting stock problem: part II. *Operations Research*, 11, pp.863-88.

Gilmore, P.C. & Gomory, R.E., 1965. Multistage cutting-stock problmes of two and more dimensions. *Operations Research*, 13, pp.94-120.

Gilmore, P.C. & Gomory, R.E., 1966. The theory and computation of Knapsack Functions. *Operations Research*, 14, pp.1045-75.

Glover, F., 1977. Heuristics for Integer Programming using Surogate Constraints. *Decisions Sciences*, 8(1), pp.156-66.

Glover, F., 1989. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3).

Glover, F., 1990. Tabu Search - Part II. *ORSA Journal on Computing*, 2(1), pp.4-32.

Glover, F., 1992. *Simple tabu threasholding in optimisation*. Internal Report. Boulder, Colorado: University of Colorado.

Glover, F. & Laguna, M., 1998. *Tabu Search*. Kluwer Academic Publishers.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goldhar, J.D. & Jelinek, M., 1985. Computer Integrated Flexible Manufacturing: Organizational, Economic, and Strategic Implications. *Interfaces*, 15(3), pp.49-105.

Gomes, A.M. & Oliveira, J.F., 2001. A GRASP approach to the nesting problem. In *4th Metaheuristics International Conference*. Porto, Portugal, 2001.

Gomes, A.M. & Oliveira, J.F., 2002. A 2-exchange heuristic for nesting problems. *European Journal of Operations Research*, 141, pp.359-70.

Gomes, A.M. & Oliveira, J.F., 2006. Solving Irregular Strip Packing problems by hybridising simulated annealing an linear programming. *European Journal of Operations Research*, 171, pp.811-29.

Goodrich, M.T., Guibas, L.J., Hershberger, J. & Tanenbaum, P.J., 1997. Snap Rounding Line Segments Effciently in Two and Three Dimensions. In *Proceedings of the 13th Symposium on Computational Geometry.*, 1997.

Gradisar, M., Resinovic, G. & Kljajic, M., 2002. Evaluation of algorithms for one-dimensional cutting. *European Journal of Operational Research*, 29, pp.1207-20.

Grinde, R.B. & Cavalier, T.M., 1995. A new algorithm for the minimal-area convex enclosure problem. *EJOR*, 84, pp.522-38.

Grinde, R.B. & Daniels, K., 1999. Solving an apparel trim placement problem using a maximum cover problem approach. *IIE Transactions*, 31, pp.763-69.

Guibas, L.J. & Marimont, D.H., 1995. Rounding Arrangements Dynamically. In *Symposium on Computational Geometry.*, 1995.

Gurel, O., 1969. *Circular Graph of Marker Layout*. Techincal Report 320-2965. IBM Scientific Centre.

Haessler, R.W., 1968. No. 69-12 *An Application of Heuristic Programming to a Nonlinear Cutting-Stock Problem Occuring in the Paper Industry*. Doctoral Dissertation. Ann Arbor: The University of Michigan.

Haessler, R.W., 1971. A Heuristic Programming Solution to a Nonlinear Cutting Stock Problem. *Management Science*, 17, pp.793-803.

Haessler, R.W., 1975. Controling cutting pattern changes in one-dimensional trim problems. *Operations Research*, 23(3), pp.483-93.

Haims, M.J., 1966. *On the Optimum Two-Dimensional Allocation Problem*. Ph.D Thesis. New York University.

Haims, M.J. & Freeman, H., 1970. A mulitstage solution to the template layout problem. *IEEE Transactions on Systems, Science and Cybernetics*, 6, pp.145-51.

Halperin, D. & Packer, E., 2002. Iterated snap rounding. *Computational Geometry: Theory and Applications*, 23(2), pp.209-25.

Han, G.C. & Na, S.J., 1996. Two-stage approach for nesting two-dimensional cutting problems using neural network and simulated annealing. *Proceedings of the Institute of Mechanical Engineers*, 210, pp.509-19.

Hart, P.E., Nilsson, N.J. & Raphael, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEE Transactions on SSC*, 4, pp.100-07.

Hassoun, M.H., 1995. *Fundamentals of Artificial Neural Networks*. The MIT Press.

Hatvany, J., 1984. CAD - State of the art and a tentative forecast. *Robotics and Computer-Integrated Manufacturing*, 1(1), pp.61-64.

Hertel, S. & Mehlhorn, K., 1983. Fast triangulation of simple polygons. In *Proceedings of the 4th International Conference in Foundations of Computational Theory, Lecture Notes in Computer Science*. Berlin, 1983. 207-218.

Hifi, M. & M'Hallah, R., 2004. Approximate algorithms for constrained circular cutting problems. *Computers and Operations Research*, 31, pp.675-94.

Holland, J.H., 1975. *Adaptation in Natural and Artificial Systems*. Michigan, US: Ann Arbor: University of Michigan Press.

Hopper, E., 2000. *Two Dimensional Packing utilising Evolutionary Algorithms and other Meta-heuristic Methods*. Ph.D Thesis. Cardiff: University of Wales.

Hopper, E. & Turton, B.C.H., 2001. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operations Research*, 128, pp.34-57.

Huyao, L., Yuanjun, H. & Bennell, J.A., 2007. The irregular nesting problem: a new approach for nofit polygon calculation. *Journal of the Operational Research Society*, 58, pp.1235-45.

Imamichi, T., Yagiura, M. & Nagamochi, H., 2009. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6, pp.345-61.

Jackson, D., 1995. Boundary Representation modeling with local tolerances. In *Proc. Symp. on Solid Modeling Foundations and CAD/CAM applications.*, 1995.

Jain, P., Fenyes, P. & Richter, R., 1990. Optimal blank nesting using simulated annealing. *Transactions of the ASME*, 114, pp.160-65.

Jakobs, S., 1996. On genetic algorithms for the packing of polygons. *European Journal of Operations Research*, 88(1), pp.165-81.

Joshi, S. & Sudit, M., 1994. Procedures for solving single-pass strip layout problems. *IIE Transactions*, 26, pp.27-37.

Juster, N.P., 1992. Modelling and representaion of dimensions and tolerances: a survey. *Computer-Aided Design*, 24(1), pp.3-17.

Kendall, G., 2000. *Applying Meta-Heuristic Algorithms to the Nesting Problem Utilising the No Fit Polygon*. PhD Thesis. Nottingham: The University of Nottingham.

Kido, T., Tagagi, K. & Nakanishi, M., 1994. Analysis and Comparisons of Genetic Algorithm, Simulated Annealing, TABU Search and Evolutionary Combination Algorithm. *Informatica*, 18, pp.399-410.

Kirkpatrick, S., Gelatt, C.D. & Veechs, M.P., 1983. Optimisation by simulated annealing. *Science*, 220, pp.671-80.

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

Larson, S.M., Snow, C.D., Shirts, M. & Pande, V.S., 2002. Folding@Home and Genome@Home: using distributed computing to tackle previously intractible problems in computational biology. In R. Grant, ed. *Computational Genomics*. Horizon Press.

Leung, S.C., Lin, Y. & Zhang, D., 2012. Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, 39(3), pp.678-86.

Levine, J.M. & Ducatelle, F., 2004. Ant colony optimisation and local search for bin-packing and cutting stock problems. *Journal of the Operation Research Society*, 55(7), pp.705-16.

Li, Z. & Milenkovic, V., 1995. Compaction and separation algorithms for non-convex polygons and their application. *European Journal of Operations Research*, 84, pp.539-61.

Lowerre, B.T., 1976. *The HARPY speech recognition system*. PhD Thesis. Pittsburgh, PA: Carnegie Mellon University.

Lui, D.Q. & Teng, H.F., 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operations Research*, 112(2), pp.413-20.

Mahadevan, A., 1984. *Optimisation in Computer-Aided Pattern Packing*. PhD Thesis. North Carolina State University.

Marques, V., Bispo, C. & Sentieiro, J., 1991. A system for the compaction of two-dimensional irregular shapes based on simulated annealing. In *International Conference on Industrial Electronics, Control and Instrumentation*. Kobe, Japan, 1991.

Martins, T.C. & Tsuzuki, M.S.G., 2010. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. *Expert Systems with Applications*, 37(3), pp.1955-72.

McCulloch, W.S. & Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, pp.115-37.

Mehlhorn, K. & Näher, S., 2000. *LEDA: a platform for combinatorial and geometric computing*. New York: Cambridge University Press.

Metropolis, N., Rosenbluth, A.W., Teller, A.H. & Teller, E., 1953. Equation of State Calculation by Fast Computing Machines. *Journal of Chemical Physics*, 21, pp.1087-91.

Milenkovic, V.J., 1988. Verifiable implementation of geometric algorithms using finite precision arithemtic. *Artificial Intelligence - Special Issue on Geometric Reasoning*, 37(1-3), pp.377-401.

Milenkovic, V., 1997. Rotational Polygon Overlap Minimization. In *Proceedings of the 13th annual symposium on Computational Geometry.*, 1997.

Milenkovic, V., 1998. Rotational polygon overlap minimization and compaction. *Computational Geometry*, 10, pp.305-18.

Milenkovic, V.J., 1999. Rotational polygon containment and minimum enclosure using only robust 2D constructions. *Computational Geometry*, 13, pp.3-19.

Milenkovic, V.J., 2000. Shortest Path Geometric Rounding. *Algorithmica*, 27(1), pp.57-86.

Morabito, R. & Arenales, M., 2000. Optimizing the cutting of stock plates in a furniture company. *International Journal of Production Research*, 38(12), pp.2725-42.

Moreau, G.R. & DeSaint Hardavin, P.T., 1969. No. 320-2978 *Marker layout problem: An experimental attempt*. Techincal Report. IBM New York Scientific Center.

Niemi, E., 2003. Nesting Strategy for Standard Parts Under Variable Demand. *Journal of Engineering Manufacture*, 217, Part B, pp.1421-28.

Oliveira, J.F. & Ferreira, J.S., 1993. Algorithms for Nesting Problems. In X. Vidal & V.V. Rene, eds. *Lecture Notes in Economics and Mathematical Systems, Applied Simulated Annealing*. Springer-Verlag. pp.225-73.

Oliveira, J.F., Gomes, A.M. & Ferreira, J.S., 2000. TOPOS - A new constructive algorithm for nesting problems. *OR Spektrum*, 22(2), pp.263-84.

O'Rourke, J., 1998. *Computational Geometry in C - 2nd Edition*. Cambridge University Press.

Ouelhadj, D. & Petrovic, S., 2010. A cooperative hyper-heuristic search framework. *Journal of Heuristics*, 16(6), pp.835-57.

Overmars, M., 1996. Desgining the Computational Geometry Algorithms Library CGAL. In *Proceedings of the Workshop on Applied Computational Geometry*. Philadelphia, Pennsylvania, 1996.

Pennings, J.M., 1987. Technological Innovations in Manufacturing. In A. Buitendam, ed. *New Technology as organisational Innovation*. Cambridge, MA: Ballinger. pp.197-216.

Poshyanonda, P. & Bahrami, X., 1992. Artificial Neural Networks in Stock Cutting Problems. In Y.C. Shin, A.H. Abodelmonem & S. Kumara, eds. *Neural Networks in Manufacturing and Robotics*. New York: ASME Press. pp.143-53.

Poshyanonda, P. & Dagli, C.H., 1992. A hybrid approach to composite stock cutting: neural network and genetic algorithm. *Robotics and Manufacturing: Recent Trends in Research*, 4, pp.775-80.

Prasad, Y.K.D. & Somasundaram, S., 1991. CASNS - A Heuristic Algorithm for the Nesting of Irregular Shaped Sheet Metal Blanks. *Computer Aided Engineering Journal*, April, pp.69-73.

Prasad, Y.K.D., Somasundaram, S. & Rao, K.P., 1995. A sliding algorithm for optimal nesting of arbitrarily shaped sheet metal blanks. *International Journal of Production Research*, 33, pp.1505-20.

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

Qu, W. & Sanders, J.L., 1987. A nesting algorithm for irregular parts and factors affecting trim losses. *International Journal of Operations Research*, 25(3), pp.381-97.

Ramesh Babu, A. & Ramesh Babu, N., 1999. Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. *International Journal of Production Research*, 112(2), pp.1625-43.

Ramesh Babu, A. & Ramesh Babu, N., 2001. A genetic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. *Computer Aided Design*, 33, pp.879-91.

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.*, 1996.

Ratanapan, K. & Dagli, C.H., 1997. An object based evolutionary algorithm for solving irregular nesting problems. In *Proceedings for Artificial Nueral Networks in Engineering Conference*. New York, 1997.

Rich, E. & Knight, K., 1990. *Artificial Intelligence*. McGraw-Hill.

Russell, S. & Norvig, P., 2009. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall.

Sato, A.K., Martins, T.C. & Tsuzuki, M.S.G., 2012. An algorithm for the strip packing problem using collision free region and exact fitting placement. *Computer-Aided Design*, 44, pp.766-77.

Scheithauer, G. & Terno, J., 1993. Modeling of packing problems. *Optimization*, 28, pp.63-84.

Seidel, R., 1991. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry Theory and Applications*, 1, pp.51-64.

Smith, D., 1985. Bin Packing with adaptive search. In *Proceedings of the First International Conference on Genetic Algorithms and Applications*. Hillsdale, New Jersey, USA, 1985. Lawrence Erlbaum Associates.

Stoyan, Y.G., Novozhilova, M.V. & Kartashov, A.V., 1996. Mathematical model and method of searching for a local extremum for the non convex oriented polygons allocation problem. *European Journal of Operations Research*, 92, pp.193-210.

Stoyan, Y. & Pankratov, A.V., 1999. Regular packing of congruent polygons on the rectangular sheet. *European Journal of Operations Research*, 113, pp.653-75.

Stoyan, Y. & Ponomarenko, L.D., 1977. SER. A10 *Minkowski sum and hodograph of the dense placement vector function*. Technical Report. SSR Academy of Science.

Stoyan, Y.G., Scheithauer, G., Gil, N. & Romanova, T., 2002. MATH-NM-2-2002 *Phi-functions for complex 2D-objects*. Technical Report. Technische Universität Dresden.

Stoyan, Y. et al., 2001. Phi-functions for primary 2D-objects. *Studia Informatica Universalis*, 2(1), pp.1-32.

Sweeny, P.E. & Paternoster, E.R., 1992. Cutting and Packing Problems: A Catagorized Application Oriented Research Bibliography. *Journal of the Operational Research Society*, 43(7), pp.691-706.

Tanaka, M. & Wachi, T., 1973. Computerized Marker Making. *Textile Machinery Society of Japan*, 19, pp.74-81.

Tay, F.E.H., Chong, T.Y. & Lee, F.C., 2002. Pattern nesting on irregular-shaped stock using genetic algorithms. *Engineering Applications of Artificial Intelligence*, 15(6), pp.551-58.

Teresa Costa, M., Gomes, A.M. & Oliveira, J.F., 2009. Heuristic approaches to large-scale periodic packing of irregular shapes on a rectangular sheet. *European Journal of Operational Research*, 192(1), pp.29-40.

Wäscher, G., Haußner, H. & Schumann, H., 2004. An Improved Typology of Cutting and Packing Problems. In *1st ESICUP Meeting*. Lutherstadt Wittenberg, Germany, 2004.

Wäscher, G., Haußner, H. & Schumann, H., 2007. An Improved Typology of Cutting and Packing Problems. *European Journal of Operational Research*, 183(3), pp.1109-30.

WenQi, H., Yu, L. & RuChu, X., 2001. Local search based on a physical model for solving a circle packing problem. In *4th Metaheuristics International Conference*. Porto, Portugal, 2001.

Wong, W.K. et al., 2009. Solving the two-dimensional irregular object allocation problems by using a two-stage packing approach. *Expert Systems with Applications*, 36, pp.3489-96.

Wu, T.H., Chen, J.F., Low, C. & Tang, P.T., 2003. Nesting of two-dimensional parts in multiple plates using hybrid algorithm. *International Journal of Production Research*, 41(16), pp.3883-900.

Yang, H. & Lin, C., 2009. On genetic algorithms fro shoe macking nesting - A Taiwan case. *Expert Systems with Applications*, 36(2), pp.1134-41.

Yap, C., 1997. Robust Geometric Computation. In J.E. Goodman & J. O'Rourke, eds. *Handbook of Discrete and Computational Geometry*. Boca Raton, FL: CRC Press LLC. Ch. 35.

Yeung, L.H.W. & Tang, W.K.S., 2003. A Hybrid Genetic Approach for Garment Cutting in the Clothing Industry. *IEEE Transactions on Industrial Electronics*, 50(3), pp.449-55.

Appendix A - New Irregular Benchmark Problems

The data for the new benchmark problems introduced in chapter 4 are detailed below. For each dataset, the problem name is stated followed by the sheet size (width, height), number of different shapes and finally the rotational constraints. After the problem definition, each shape of that dataset is defined including the shape number, number of loops and quantity that must be packed onto the sheet. This is followed by a list of the closed loops that define the shape. Each loop of the shape is defined by its number, whether it is external or internal and the number of line/arc primitives that define the loop. Then a list of line/arc primitives is given for that closed loop. A line is defined by its start and end coordinates. An arc is described by its start and end coordinates, a centre point coordinate, radius, start angle and offset angle (-ve offset is convex, +ve offset is concave).

```
Profiles1: (4000,2000), Shapes: 8, Rotations: 90 incremental

 Shape 1 (Loops: 1, Quantity: 4)
   Loop 1 (external): 6 Primitives
      Line: (99.952271, 330.356628),(209.952271, 330.356628)
      Line: (209.952271, 330.356628),(209.952271, 210.356644)
      Line: (209.952271, 210.356644),(369.952271, 210.356644)
      Line: (369.952271, 210.356644),(369.952271, 90.356644)
      Line: (369.952271, 90.356644),(99.952271, 90.356644)
      Line: (99.952271, 90.356644),(99.952271, 330.356628)

 Shape 2 (Loops: 1, Quantity: 4)
   Loop 1 (external): 6 Primitives
      Line: (128.702362, 153.247681),(128.702362, 423.247681)
      Line: (128.702362, 423.247681),(202.820251, 423.247681)
       Arc: (202.820251, 423.247681),(454.584534, 423.247681),
            Cen: (328.702393, 423.247681), Rad: 125.882141,
            StAng: -3.141593, Offset 3.141593
      Line: (454.584534, 423.247681),(528.702393, 423.247681)
      Line: (528.702393, 423.247681),(528.702393, 153.247681)
      Line: (528.702393, 153.247681),(128.702362, 153.247681)

 Shape 3 (Loops: 1, Quantity: 4)
   Loop 1 (external): 8 Primitives
      Line: (158.517273, 238.100677),(158.517273, 338.100677)
      Line: (158.517273, 338.100677),(278.517273, 338.100677)
      Line: (278.517273, 338.100677),(278.517273, 308.100677)
      Line: (278.517273, 308.100677),(248.517273, 308.100677)
      Line: (248.517273, 308.100677),(248.517273, 268.100677)
      Line: (248.517273, 268.100677),(218.517273, 268.100677)
      Line: (218.517273, 268.100677),(218.517273, 238.100677)
      Line: (218.517273, 238.100677),(158.517273, 238.100677)

 Shape 4 (Loops: 1, Quantity: 4)
   Loop 1 (external): 1 Primitives
       Arc: (315.091614, 272.041870),(315.091614, 272.041870),
            Cen: (250.091621, 272.041870), Rad: 64.999992,
            StAng: 0.000000, Offset -6.283185

 Shape 5 (Loops: 1, Quantity: 4)
   Loop 1 (external): 1 Primitives
       Arc: (301.109985, 305.983063),(301.109985, 305.983063),
            Cen: (201.109993, 305.983063), Rad: 99.999992,
            StAng: 0.000000, Offset -6.283185

 Shape 6 (Loops: 1, Quantity: 4)
   Loop 1 (external): 4 Primitives
      Line: (381.943237, 381.199280),(581.943237, 381.199280)
      Line: (581.943237, 381.199280),(781.943237, 181.199265)
      Line: (781.943237, 181.199265),(181.943268, 181.199265)
      Line: (181.943268, 181.199265),(381.943237, 381.199280)

 Shape 7 (Loops: 1, Quantity: 4)
   Loop 1 (external): 10 Primitives
      Line: (117.980972, 190.649583),(246.121307, 226.775071)
       Arc: (246.121307, 226.775071),(237.980957, 285.649583),
            Cen: (237.980934, 255.649556), Rad: 30.000026,
            StAng: -1.296006, Offset: 2.866801
      Line: (237.980957, 285.649583),(237.980957, 370.649567)
      Line: (237.980957, 370.649567),(117.980972, 420.649567)
      Line: (117.980972, 420.649567),(117.980972, 470.649567)
      Line: (117.980972, 470.649567),(942.821777, 390.414704)
       Arc: (942.821777, 390.414704),(947.053282, 291.479534),
            Cen: (937.980947, 340.649583), Rad: 50.000009,
            StAng: 1.473828, Offset -2.862167
      Line: (947.053282, 291.479534),(237.980957, 160.649583)
      Line: (237.980957, 160.649583),(117.980972, 160.649583)
      Line: (117.980972, 160.649583),(117.980972, 190.649583)

 Shape 8 (Loops: 1, Quantity: 4)
   Loop 1 (external): 6 Primitives
      Line: (110.497086, 255.303055),(110.497086, 355.303040)
      Line: (110.497086, 355.303040),(261.497070, 437.303040)
      Line: (261.497070, 437.303040),(861.497070, 437.303040)
      Line: (861.497070, 437.303040),(1012.497070, 355.303040)
      Line: (1012.497070, 355.303040),(1012.497070, 255.303055)
      Line: (1012.497070, 255.303055),(110.497086, 255.303055)


Profiles2: (5000,2500), Shapes: 7, Rotations: 90 incremental

 Shape 1 (Loops: 4, Quantity: 5)
   Loop 1 (internal): 1 Primitives
       Arc: (202.238806, 283.582090),(202.238806, 283.582090),
            Cen: (152.238806, 283.582090), Rad: 50.000000,
            StAng: 0.000000, Offset -6.283185
   Loop 2 (internal): 1 Primitives
       Arc: (1002.238806, 283.582090),(1002.238806, 283.582090),
            Cen: (952.238806, 283.582090), Rad: 50.000000,
            StAng: 0.000000, Offset -6.283185
   Loop 3 (internal): 1 Primitives
       Arc: (727.238806, 283.582090),(727.238806, 283.582090),
            Cen: (552.238806, 283.582090), Rad: 175.000000,
            StAng: 0.000000, Offset -6.283185
   Loop 4 (external): 8 Primitives
       Arc: (787.506663, 469.724603),(913.002695, 428.359600),
            Cen: (881.613806, 544.181609), Rad: 120.000000,
            StAng: -2.472244, Offset 1.166101
       Arc: (913.002695, 428.359600),(913.002695, 138.804579),
            Cen: (952.238806, 283.582090), Rad: 150.000000,
            StAng: 1.835449, Offset -3.670898
       Arc: (913.002695, 138.804579),(787.506663, 97.439576),
            Cen: (881.613806, 22.982570), Rad: 120.000000,
            StAng: 1.306144, Offset 1.166101
       Arc: (787.506663, 97.439576),(316.970949, 97.439576),
            Cen: (552.238806, 283.582090), Rad: 300.000000,
            StAng: -0.669348, Offset -1.802896
       Arc: (316.970949, 97.439576),(191.474917, 138.804579),
            Cen: (222.863806, 22.982570), Rad: 120.000000,
            StAng: 0.669348, Offset 1.166101
       Arc: (191.474917, 138.804579),(191.474917, 428.359600),
            Cen: (152.238806, 283.582090), Rad: 150.000000,
            StAng: -1.306144, Offset -3.670898
       Arc: (191.474917, 428.359600),(316.970949, 469.724603),
            Cen: (222.863806, 544.181609), Rad: 120.000000,
            StAng: -1.835449, Offset 1.166101
       Arc: (316.970949, 469.724603),(787.506663, 469.724603),
            Cen: (552.238806, 283.582090), Rad: 300.000000,
            StAng: 2.472244, Offset -1.802896

 Shape 2 (Loops: 1, Quantity: 7)
   Loop 1 (external): 12 Primitives
      Line: (3.735266, 36.876257),(3.735266, 356.876257)
      Line: (3.735266, 356.876257),(33.735266, 386.876257)
      Line: (33.735266, 386.876257),(183.735266, 386.876257)
      Line: (183.735266, 386.876257),(183.735266, 371.876257)
       Arc: (183.735266, 371.876257),(228.735266, 326.876257),
            Cen: (228.735266, 371.876257), Rad: 45.000000,
            StAng: 3.141593, Offset 1.570796
      Line: (228.735266, 326.876257),(473.735266, 326.876257)
      Line: (473.735266, 326.876257),(473.735266, 76.876257)
      Line: (473.735266, 76.876257),(228.735266, 76.876257)
       Arc: (228.735266, 76.876257),(183.735266, 31.876257),
            Cen: (228.735266, 31.876257), Rad: 45.000000,
            StAng: 1.570796, Offset 1.570796
      Line: (183.735266, 31.876257),(183.735266, 16.876257)
      Line: (183.735266, 16.876257),(23.735266, 16.876257)
      Line: (23.735266, 16.876257),(3.735266, 36.876257)

 Shape 3 (Loops: 2, Quantity: 4)
   Loop 1 (internal): 4 Primitives
      Line: (244.630072, 188.544153),(244.630072, 633.651551)
      Line: (244.630072, 633.651551),(840.095465, 633.651551)
      Line: (840.095465, 633.651551),(840.095465, 188.544153)
      Line: (840.095465, 188.544153),(244.630072, 188.544153)
   Loop 2 (external): 4 Primitives
      Line: (196.897375, 143.198091),(196.897375, 681.384248)
      Line: (196.897375, 681.384248),(898.568019, 681.384248)
      Line: (898.568019, 681.384248),(898.568019, 143.198091)
      Line: (898.568019, 143.198091),(196.897375, 143.198091)

 Shape 4 (Loops: 1, Quantity: 6)
   Loop 1 (external): 8 Primitives
      Line: (193.648896, 229.989415),(193.648896, 129.989415)
      Line: (193.648896, 129.989415),(218.648896, 129.989415)
      Line: (218.648896, 129.989415),(218.648896, 29.989415)
      Line: (218.648896, 29.989415),(-11.351104, 29.989415)
      Line: (-11.351104, 29.989415),(-11.351104, 129.989415)
      Line: (-11.351104, 129.989415),(13.648896, 129.989415)
      Line: (13.648896, 129.989415),(13.648896, 229.989415)
       Arc: (13.648896, 229.989415),(193.648896, 229.989415),
            Cen: (103.648896, 229.989415), Rad: 90.000000,
            StAng: 3.141593, Offset -3.141593

 Shape 5 (Loops: 1, Quantity: 7)
   Loop 1 (external): 5 Primitives
      Line: (398.524900, 658.183912),(633.639000, 334.577114)
      Line: (633.639000, 334.577114),(398.524900, 10.970317)
      Line: (398.524900, 10.970317),(18.102293, 134.577114)
      Line: (18.102293, 134.577114),(18.102293, 534.577114)
      Line: (18.102293, 534.577114),(398.524900, 658.183912)

 Shape 6 (Loops: 1, Quantity: 9)
   Loop 1 (external): 6 Primitives
      Line: (236.459853, 24.359731),(-75.862367, 24.359526)
      Line: (-75.862367, 24.359526),(-75.862598, 377.685979)
      Line: (-75.862598, 377.685979),(-29.782271, 377.686009)
      Line: (-29.782271, 377.686009),(-29.782082, 88.652454)
      Line: (-29.782082, 88.652454),(236.459811, 88.652628)
      Line: (236.459811, 88.652628),(236.459853, 24.359731)

 Shape 7 (Loops: 1, Quantity: 12)
   Loop 1 (external): 12 Primitives
      Line: (4.588083, 72.575156),(4.588083, 192.575156)
      Line: (4.588083, 192.575156),(44.588083, 192.575156)
      Line: (44.588083, 192.575156),(44.588083, 232.575156)
      Line: (44.588083, 232.575156),(164.588083, 232.575156)
      Line: (164.588083, 232.575156),(164.588083, 192.575156)
      Line: (164.588083, 192.575156),(204.588083, 192.575156)
      Line: (204.588083, 192.575156),(204.588083, 72.575156)
      Line: (204.588083, 72.575156),(164.588083, 72.575156)
      Line: (164.588083, 72.575156),(164.588083, 32.575156)
      Line: (164.588083, 32.575156),(44.588083, 32.575156)
      Line: (44.588083, 32.575156),(44.588083, 72.575156)
      Line: (44.588083, 72.575156),(4.588083, 72.575156)
```

```
Profiles3: (10000,2500), Shapes: 6, Rotations: 45 incremental

 Shape 1 (Loops: 1, Quantity: 4)
   Loop 1 (external): 16 Primitives
      Line: (404.534606, 794.749403),(440.334129, 531.026253)
      Line: (440.334129, 531.026253),(686.157518, 616.945107)
      Line: (686.157518, 616.945107),(505.966587, 463.007160)
```

```
         Line: (505.966587, 463.007160),(706.443914, 340.095465)
         Line: (706.443914, 340.095465),(479.713604, 353.221957)
         Line: (479.713604, 353.221957),(508.353222, 107.398568)
         Line: (508.353222, 107.398568),(366.348449, 303.102625)
         Line: (366.348449, 303.102625),(219.570406, 95.465394)
         Line: (219.570406, 95.465394),(244.630072, 328.162291)
         Line: (244.630072, 328.162291),(-0.000000, 206.443914)
         Line: (-0.000000, 206.443914),(183.770883, 377.088305)
         Line: (183.770883, 377.088305),(-57.279236, 531.026253)
         Line: (-57.279236, 531.026253),(187.350835, 485.680191)
         Line: (187.350835, 485.680191),(134.844869, 751.789976)
         Line: (134.844869, 751.789976),(317.422434, 565.632458)
         Line: (317.422434, 565.632458),(404.534606, 794.749403)

   Shape 2 (Loops: 1, Quantity: 12)
      Loop 1 (external): 13 Primitives
         Line: (338.902148, 745.823389),(348.448687, 807.875895)
         Line: (348.448687, 807.875895),(474.940334, 809.069212)
         Line: (474.940334, 809.069212),(564.439141, 806.682578)
         Line: (564.439141, 806.682578),(637.231504, 725.536993)
         Line: (637.231504, 725.536993),(579.952267, 643.198091)
         Line: (579.952267, 643.198091),(348.448687, 667.064439)
         Line: (348.448687, 667.064439),(350.835322, 601.431981)
         Line: (350.835322, 601.431981),(558.472554, 610.978520)
         Line: (558.472554, 610.978520),(609.785203, 448.687351)
         Line: (609.785203, 448.687351),(371.121718, 335.322196)
         Line: (371.121718, 335.322196),(164.677804, 515.513126)
         Line: (164.677804, 515.513126),(132.458234, 652.744630)
         Line: (132.458234, 652.744630),(338.902148, 745.823389)

   Shape 3 (Loops: 1, Quantity: 7)
      Loop 1 (external): 5 Primitives
         Line: (58.303887, -30.035336),(58.303887, 888.692580)
         Line: (58.303887, 888.692580),(531.802120, 415.194346)
         Line: (531.802120, 415.194346),(1005.300353, 888.692580)
         Line: (1005.300353, 888.692580),(1005.300353, -30.035336)
         Line: (1005.300353, -30.035336),(58.303887, -30.035336)

   Shape 4 (Loops: 1, Quantity: 5)
      Loop 1 (external): 8 Primitives
         Line: (84.725537, 180.190931),(84.725537, 807.875895)
         Line: (84.725537, 807.875895),(855.608592, 807.875895)
         Line: (855.608592, 807.875895),(843.675418, 735.083532)
         Line: (843.675418, 735.083532),(136.038186, 735.083532)
         Line: (136.038186, 735.083532),(136.038186, 236.276850)
         Line: (136.038186, 236.276850),(843.675418, 236.276850)
         Line: (843.675418, 236.276850),(855.608592, 180.190931)
         Line: (855.608592, 180.190931),(84.725537, 180.190931)

  Shape 5 (Loops: 1, Quantity: 10)
      Loop 1 (external): 6 Primitives
         Line: (38.394054, 159.795716),(128.394054, 709.795716)
          Arc: (128.394054, 709.795716),(289.369894, 749.051604),
               Cen: (217.212767, 695.261745), Rad: 90.000000,
               StAng: 2.979394, Offset -2.338808
         Line: (289.369894, 749.051604),(699.369894, 199.051604)
          Arc: (699.369894, 199.051604),(627.212767, 55.261745),
               Cen: (627.212767, 145.261745), Rad: 90.000000,
               StAng: 0.640586, Offset -2.211382
         Line: (627.212767, 55.261745),(127.212767, 55.261745)
          Arc: (127.212767, 55.261745),(38.394054, 159.795716),
               Cen: (127.212767, 145.261745), Rad: 90.000000,
               StAng: -1.570796, Offset -1.732995

   Shape 6 (Loops: 1, Quantity: 8)
      Loop 1 (external): 6 Primitives
         Line: (-1036.992840, 529.832936),(-920.047733, 789.976134)
         Line: (-920.047733, 789.976134),(-237.470167, 565.632458)
         Line: (-237.470167, 565.632458),(-399.761337, 300.715990)
         Line: (-399.761337, 300.715990),(-843.675418, 613.365155)
         Line: (-843.675418, 613.365155),(-808.113597, 497.949032)
          Arc: (-808.113597, 497.949032),(-1036.992840,
529.832936),
               Cen: (-958.233890, 257.756563), Rad: 283.246403,
               StAng: 1.012197, Offset -5.442814
```

Profiles4: (5000,500), Shapes: 7, Rotations: 90 incremental

```
   Shape 1 (Loops: 1, Quantity: 4)
      Loop 1 (external): 4 Primitives
         Line: (38.121547, 49.490946),(38.121547, 249.490946)
          Arc: (38.121547, 249.490946),(238.121547, 249.490946),
               Cen: (138.121547, 244.490946), Rad: 100.124922,
               StAng: 3.091634, Offset -3.041676
         Line: (238.121547, 249.490946),(238.121547, 49.490946)
          Arc: (238.121547, 49.490946),(38.121547, 49.490946),
               Cen: (138.121547, 44.490946), Rad: 100.124922,
               StAng: 0.049958, Offset 3.041676

   Shape 2 (Loops: 1, Quantity: 7)
      Loop 1 (external): 11 Primitives
         Line: (239.856802, 730.310263),(255.369928, 735.083532)
         Line: (255.369928, 735.083532),(303.102625, 741.050119)
         Line: (303.102625, 741.050119),(365.155131, 732.696897)
         Line: (365.155131, 732.696897),(426.014320, 699.284010)
         Line: (426.014320, 699.284010),(434.367542, 619.331742)
         Line: (434.367542, 619.331742),(420.047733, 559.665871)
         Line: (420.047733, 559.665871),(369.928401, 513.126492)
         Line: (369.928401, 513.126492),(227.923628, 503.579952)
         Line: (227.923628, 503.579952),(145.584726, 559.665871)
         Line: (145.584726, 559.665871),(139.618138, 651.551313)
         Line: (139.618138, 651.551313),(239.856802, 730.310263)

   Shape 3 (Loops: 1, Quantity: 7)
      Loop 1 (external): 5 Primitives
         Line: (5.966587, 5.966587),(5.966587, 214.797136)
         Line: (5.966587, 214.797136),(87.708831, 296.539379)
         Line: (87.708831, 296.539379),(169.451074, 214.797136)
         Line: (169.451074, 214.797136),(169.451074, 5.966587)
         Line: (169.451074, 5.966587),(5.966587, 5.966587)

   Shape 4 (Loops: 1, Quantity: 15)
      Loop 1 (external): 3 Primitives
         Line: (137.231504, 836.515513),(223.150358, 850.835322)
         Line: (223.150358, 850.835322),(212.410501, 677.804296)
         Line: (212.410501, 677.804296),(137.231504, 836.515513)

   Shape 5 (Loops: 1, Quantity: 7)
      Loop 1 (external): 3 Primitives
         Line: (-0.000000, 0.000000),(53.776627, 84.309397)
         Line: (53.776627, 84.309397),(107.466293, -0.055405)
         Line: (107.466293, -0.055405),(-0.000000, 0.000000)

   Shape 6 (Loops: 1, Quantity: 7)
      Loop 1 (external): 1 Primitives
          Arc: (50.000000, 0.000000),(50.000000, 0.000000),
               Cen: (0.000000, 0.000000), Rad: 50.000000,
               StAng: 0.000000, Offset -6.283185

   Shape 7 (Loops: 1, Quantity: 7)
      Loop 1 (external): 4 Primitives
         Line: (0.000000, 0.000000),(0.000000, 50.000000)
          Arc: (0.000000, 50.000000),(50.000000, 50.000000),
               Cen: (25.000000, 46.743828), Rad: 25.211161,
               StAng: 3.012075, Offset -2.882557
         Line: (50.000000, 50.000000),(50.000000, 0.000000)
         Line: (50.000000, 0.000000),(0.000000, 0.000000)
```

Profiles5: (8000,4000), Shapes: 5, Rotations: 15 incremental

```
   Shape 1 (Loops: 1, Quantity: 10)
      Loop 1 (external): 10 Primitives
         Line: (117.980972, 190.649583),(246.121307, 226.775071)
          Arc: (246.121307, 226.775071),(237.980957, 285.649583),
               Cen: (237.980934, 255.649556), Rad: 30.000026,
               StAng: -1.296006, Offset 2.866801
         Line: (237.980957, 285.649583),(237.980957, 370.649567)
         Line: (237.980957, 370.649567),(117.980972, 420.649567)
         Line: (117.980972, 420.649567),(117.980972, 470.649567)
         Line: (117.980972, 470.649567),(942.821777, 390.414704)
          Arc: (942.821777, 390.414704),(947.053282, 291.479534),
               Cen: (937.980947, 340.649583), Rad: 50.000009,
               StAng: 1.473828, Offset -2.862167
         Line: (947.053282, 291.479534),(237.980957, 160.649583)
         Line: (237.980957, 160.649583),(117.980972, 160.649583)
         Line: (117.980972, 160.649583),(117.980972, 190.649583)

   Shape 2 (Loops: 1, Quantity: 10)
      Loop 1 (external): 4 Primitives
         Line: (381.943237, 381.199280),(581.943237, 381.199280)
         Line: (581.943237, 381.199280),(781.943237, 181.199265)
         Line: (781.943237, 181.199265),(181.943268, 181.199265)
         Line: (181.943268, 181.199265),(381.943237, 381.199280)

   Shape 3 (Loops: 1, Quantity: 10)
      Loop 1 (external): 5 Primitives
         Line: (398.524900, 658.183912),(633.639000, 334.577114)
         Line: (633.639000, 334.577114),(398.524900, 10.970317)
         Line: (398.524900, 10.970317),(18.102293, 134.577114)
         Line: (18.102293, 134.577114),(18.102293, 534.577114)
         Line: (18.102293, 534.577114),(398.524900, 658.183912)

   Shape 4 (Loops: 1, Quantity: 10)
      Loop 1 (external): 5 Primitives
         Line: (280.429594, 193.317422),(280.429594, 613.365155)
          Arc: (280.429594, 613.365155),(522.076372, 613.365155),
               Cen: (401.252983, 609.425990), Rad: 120.887586,
               StAng: 3.109002, Offset -3.076410
          Arc: (522.076372, 613.365155),(763.723150, 613.365155),
               Cen: (642.899761, 637.483107), Rad: 123.207008,
               StAng: -2.944569, Offset 2.747545
         Line: (763.723150, 613.365155),(763.723150, 193.317422)
         Line: (763.723150, 193.317422),(280.429594, 193.317422)

   Shape 5 (Loops: 1, Quantity: 10)
      Loop 1 (external): 6 Primitives
         Line: (38.394054, 159.795716),(128.394054, 709.795716)
          Arc: (128.394054, 709.795716),(289.369894, 749.051604),
               Cen: (217.212767, 695.261745), Rad: 90.000000,
               StAng: 2.979394, Offset -2.338808)
         Line: (289.369894, 749.051604),(699.369894, 199.051604)
          Arc: (699.369894, 199.051604),(627.212767, 55.261745),
               Cen: (627.212767, 145.261745), Rad: 90.000000,
               StAng: 0.640586, Offset -2.211382)
         Line: (627.212767, 55.261745),(127.212767, 55.261745)
          Arc: (127.212767, 55.261745),(38.394054, 159.795716),
               Cen: (127.212767, 145.261745), Rad: 90.000000,
               StAng: -1.570796, Offset -1.732995)
```

Profiles6: (10000,5000), Shapes: 9, Rotations: 90 incremental

```
Shape 1 (Loops: 1, Quantity: 8)
   Loop 1 (external): 8 Primitives
      Line: (-569.212411, 1789.976134),(-569.212411,
2429.594272)
      Line: (-569.212411, 2429.594272),(714.797136, 2429.594272)
      Line: (714.797136, 2429.594272),(705.250597, 2229.116945)
      Line: (705.250597, 2229.116945),(-263.723150, 2238.663484)
      Line: (-263.723150, 2238.663484),(-273.269690,
1928.400955)
      Line: (-273.269690, 1928.400955),(681.384248, 1942.720764)
      Line: (681.384248, 1942.720764),(714.797136, 1789.976134)
      Line: (714.797136, 1789.976134),(-569.212411, 1789.976134)

Shape 2 (Loops: 1, Quantity: 8)
   Loop 1 (external): 4 Primitives
      Line: (1125.298329, 2214.797136),(1898.568019, 2262.529833)
      Line: (1898.568019, 2262.529833),(1941.527446, 1971.360382)
      Line: (1941.527446, 1971.360382),(1134.844869, 1952.267303)
      Line: (1134.844869, 1952.267303),(1125.298329, 2214.797136)

Shape 3 (Loops: 2, Quantity: 4)
   Loop 1 (internal): 4 Primitives
      Line: (203.980100, 167.910448),(203.980100, 644.278607)
      Line: (203.980100, 644.278607),(935.323383, 644.278607)
      Line: (935.323383, 644.278607),(935.323383, 167.910448)
      Line: (935.323383, 167.910448),(203.980100, 167.910448)
   Loop 2 (external): 4 Primitives
      Line: (138.059701, 109.452736),(138.059701, 706.467662)
      Line: (138.059701, 706.467662),(1002.487562, 706.467662)
      Line: (1002.487562, 706.467662),(1002.487562, 109.452736)
      Line: (1002.487562, 109.452736),(138.059701, 109.452736)

Shape 4 (Loops: 1, Quantity: 8)
   Loop 1 (external): 13 Primitives
      Line: (338.902148, 745.823389),(348.448687, 807.875895)
      Line: (348.448687, 807.875895),(474.940334, 809.069212)
      Line: (474.940334, 809.069212),(564.439141, 806.682578)
      Line: (564.439141, 806.682578),(637.231504, 725.536993)
      Line: (637.231504, 725.536993),(579.952267, 643.198091)
      Line: (579.952267, 643.198091),(348.448687, 667.064439)
      Line: (348.448687, 667.064439),(350.835322, 601.431981)
      Line: (350.835322, 601.431981),(558.472554, 610.978520)
      Line: (558.472554, 610.978520),(609.785203, 448.687351)
      Line: (609.785203, 448.687351),(371.121718, 335.322196)
      Line: (371.121718, 335.322196),(164.677804, 515.513126)
      Line: (164.677804, 515.513126),(132.458234, 652.744630)
      Line: (132.458234, 652.744630),(338.902148, 745.823389)

Shape 5 (Loops: 1, Quantity: 8)
   Loop 1 (external): 5 Primitives
      Line: (207.637232, 108.591885),(207.637232, 657.517900)
      Line: (207.637232, 657.517900),(412.887828, 147.971360)
      Line: (412.887828, 147.971360),(669.451074, 657.517900)
      Line: (669.451074, 657.517900),(669.451074, 108.591885)
      Line: (669.451074, 108.591885),(207.637232, 108.591885)

Shape 6 (Loops: 1, Quantity: 8)
   Loop 1 (external): 3 Primitives
      Line: (214.797136, 686.157518),(653.937947, 680.190931)
      Line: (653.937947, 680.190931),(414.081146, 182.577566)
      Line: (414.081146, 182.577566),(214.797136, 686.157518)

Shape 7 (Loops: 2, Quantity: 5)
   Loop 1 (internal): 4 Primitives
      Line: (1453.056148, 201.492537),(1453.056148, 467.661692)
      Line: (1453.056148, 467.661692),(1951.812367, 467.661692)
      Line: (1951.812367, 467.661692),(1951.812367, 201.492537)
      Line: (1951.812367, 201.492537),(1453.056148, 201.492537)
   Loop 2 (external): 4 Primitives
      Line: (1404.548685, 150.497512),(1404.548685, 521.144279)
      Line: (1404.548685, 521.144279),(2010.270078, 521.144279)
      Line: (2010.270078, 521.144279),(2010.270078, 150.497512)
      Line: (2010.270078, 150.497512),(1404.548685, 150.497512)

Shape 8 (Loops: 1, Quantity: 4)
   Loop 1 (external): 10 Primitives
      Line: (825.775656, 343.675418),(713.603819, 231.503580)
      Line: (713.603819, 231.503580),(607.398568, 394.988067)
      Line: (607.398568, 394.988067),(588.305489, 305.489260)
      Line: (588.305489, 305.489260),(398.568019, 307.875895)
      Line: (398.568019, 307.875895),(414.081146, 449.880668)
      Line: (414.081146, 449.880668),(559.665871, 449.880668)
      Line: (559.665871, 449.880668),(488.066826, 501.193317)
      Line: (488.066826, 501.193317),(647.971360, 523.866348)
      Line: (647.971360, 523.866348),(769.689737, 479.713604)
      Line: (769.689737, 479.713604),(825.775656, 343.675418)

Shape 9 (Loops: 1, Quantity: 12)
   Loop 1 (external): 8 Primitives
      Line: (468.435543, 34.696133),(21.435543, 34.696133)
      Line: (21.435543, 34.696133),(21.435543, 274.696133)
      Line: (21.435543, 274.696133),(170.435543, 274.696133)
      Line: (170.435543, 274.696133),(170.435543, 64.696133)
      Line: (170.435543, 64.696133),(319.435543, 64.696133)
      Line: (319.435543, 64.696133),(319.435543, 274.696133)
      Line: (319.435543, 274.696133),(468.435543, 274.696133)
      Line: (468.435543, 274.696133),(468.435543, 34.696133)
```

Profiles7: (2500,500), Shapes: 9, Rotations: 90 incremental

```
Shape 1 (Loops: 1, Quantity: 1)
   Loop 1 (external): 4 Primitives
      Line: (48.342460, 448.434459),(48.342460, 698.434459)
      Line: (48.342460, 698.434459),(293.727356, 698.193599)
      Line: (293.727356, 698.193599),(330.361696, 603.837311)
      Line: (330.361696, 603.837311),(48.342460, 448.434459)

Shape 2 (Loops: 1, Quantity: 1)
   Loop 1 (external): 6 Primitives
      Line: (48.342623, 198.434459),(281.342081, 455.735420)
      Line: (281.342081, 455.735420),(48.342460, 448.434459)
      Line: (48.342460, 448.434459),(330.361696, 603.837311)
      Line: (330.361696, 603.837311),(439.352160, 401.135967)
      Line: (439.352160, 401.135967),(548.342623, 198.434622)
      Line: (548.342623, 198.434622),(48.342623, 198.434459)

Shape 3 (Loops: 1, Quantity: 1)
   Loop 1 (external): 3 Primitives
      Line: (48.342623, 198.434459),(48.342460, 448.434459)
      Line: (48.342460, 448.434459),(281.342081, 455.735420)
      Line: (281.342081, 455.735420),(48.342623, 198.434459)

Shape 4 (Loops: 1, Quantity: 1)
   Loop 1 (external): 4 Primitives
      Line: (330.361696, 603.837311),(293.727356, 698.193599)
      Line: (293.727356, 698.193599),(1048.342623, 698.434786)
      Line: (1048.342623, 698.434786),(689.352160, 651.136048)
      Line: (689.352160, 651.136048),(330.361696, 603.837311)

Shape 5 (Loops: 1, Quantity: 1)
   Loop 1 (external): 7 Primitives
      Line: (552.514464, 378.712089),(556.686306, 558.989555)
      Line: (556.686306, 558.989555),(439.352160, 401.135967)
      Line: (439.352160, 401.135967),(330.361696, 603.837311)
      Line: (330.361696, 603.837311),(689.352160, 651.136048)
      Line: (689.352160, 651.136048),(788.225768, 348.309401)
      Line: (788.225768, 348.309401),(778.839033, 348.309401)
      Line: (778.839033, 348.309401),(552.514464, 378.712089)

Shape 6 (Loops: 1, Quantity: 1)
   Loop 1 (external): 4 Primitives
      Line: (439.352160, 401.135967),(556.686306, 558.989555)
      Line: (556.686306, 558.989555),(552.514464, 378.712089)
      Line: (552.514464, 378.712089),(548.342623, 198.434622)
      Line: (548.342623, 198.434622),(439.352160, 401.135967)

Shape 7 (Loops: 1, Quantity: 1)
   Loop 1 (external): 4 Primitives
      Line: (788.225768, 348.309401),(689.352160, 651.136048)
      Line: (689.352160, 651.136048),(1048.342623, 698.434786)
      Line: (1048.342623, 698.434786),(1048.342623, 448.434786)
      Line: (1048.342623, 448.434786),(788.225768, 348.309401)
Shape 8 (Loops: 1, Quantity: 1)
   Loop 1 (external): 5 Primitives
      Line: (552.514464, 378.712089),(778.839033, 348.309401)
      Line: (778.839033, 348.309401),(843.503207, 223.152873)
      Line: (843.503207, 223.152873),(1048.342623, 198.434786)
      Line: (1048.342623, 198.434786),(548.342623, 198.434622)
      Line: (548.342623, 198.434622),(552.514464, 378.712089)

Shape 9 (Loops: 1, Quantity: 1)
   Loop 1 (external): 5 Primitives
      Line: (788.225768, 348.309401),(1048.342623, 448.434786)
      Line: (1048.342623, 448.434786),(1048.342623, 198.434786)
      Line: (1048.342623, 198.434786),(843.503207, 223.152873)
      Line: (843.503207, 223.152873),(778.839033, 348.309401)
      Line: (778.839033, 348.309401),(788.225768, 348.309401)
```

Profiles8: (2500,1000), Shapes: 9, Rotations: 90 incremental

```
Shape 1 (Loops: 1, Quantity: 2)
   Loop 1 (external): 4 Primitives
      Line: (48.342460, 448.434459),(48.342460, 698.434459)
      Line: (48.342460, 698.434459),(293.727356, 698.193599)
      Line: (293.727356, 698.193599),(330.361696, 603.837311)
      Line: (330.361696, 603.837311),(48.342460, 448.434459)

Shape 2 (Loops: 1, Quantity: 2)
   Loop 1 (external): 6 Primitives
      Line: (48.342623, 198.434459),(281.342081, 455.735420)
      Line: (281.342081, 455.735420),(48.342460, 448.434459)
      Line: (48.342460, 448.434459),(330.361696, 603.837311)
      Line: (330.361696, 603.837311),(439.352160, 401.135967)
      Line: (439.352160, 401.135967),(548.342623, 198.434622)
      Line: (548.342623, 198.434622),(48.342623, 198.434459)

Shape 3 (Loops: 1, Quantity: 2)
   Loop 1 (external): 3 Primitives
      Line: (48.342623, 198.434459),(48.342460, 448.434459)
      Line: (48.342460, 448.434459),(281.342081, 455.735420)
      Line: (281.342081, 455.735420),(48.342623, 198.434459)

Shape 4 (Loops: 1, Quantity: 2)
   Loop 1 (external): 4 Primitives
      Line: (330.361696, 603.837311),(293.727356, 698.193599)
      Line: (293.727356, 698.193599),(1048.342623, 698.434786)
      Line: (1048.342623, 698.434786),(689.352160, 651.136048)
      Line: (689.352160, 651.136048),(330.361696, 603.837311)

Shape 5 (Loops: 1, Quantity: 2)
   Loop 1 (external): 7 Primitives
      Line: (552.514464, 378.712089),(556.686306, 558.989555)
      Line: (556.686306, 558.989555),(439.352160, 401.135967)
      Line: (439.352160, 401.135967),(330.361696, 603.837311)
      Line: (330.361696, 603.837311),(689.352160, 651.136048)
      Line: (689.352160, 651.136048),(788.225768, 348.309401)
      Line: (788.225768, 348.309401),(778.839033, 348.309401)
      Line: (778.839033, 348.309401),(552.514464, 378.712089)

  Shape 6 (Loops: 1, Quantity: 2)
    Loop 1 (external): 4 Primitives
      Line: (439.352160, 401.135967),(556.686306, 558.989555)
      Line: (556.686306, 558.989555),(552.514464, 378.712089)
      Line: (552.514464, 378.712089),(548.342623, 198.434622)
      Line: (548.342623, 198.434622),(439.352160, 401.135967)

  Shape 7 (Loops: 1, Quantity: 2)
    Loop 1 (external): 4 Primitives
      Line: (788.225768, 348.309401),(689.352160, 651.136048)
      Line: (689.352160, 651.136048),(1048.342623, 698.434786)
      Line: (1048.342623, 698.434786),(1048.342623, 448.434786)
      Line: (1048.342623, 448.434786),(788.225768, 348.309401)

  Shape 8 (Loops: 1, Quantity: 2)
    Loop 1 (external): 5 Primitives
      Line: (552.514464, 378.712089),(778.839033, 348.309401)
      Line: (778.839033, 348.309401),(843.503207, 223.152873)
      Line: (843.503207, 223.152873),(1048.342623, 198.434786)
      Line: (1048.342623, 198.434786),(548.342623, 198.434622)
      Line: (548.342623, 198.434622),(552.514464, 378.712089)

  Shape 9 (Loops: 1, Quantity: 2)
    Loop 1 (external): 5 Primitives
      Line: (788.225768, 348.309401),(1048.342623, 448.434786)
      Line: (1048.342623, 448.434786),(1048.342623, 198.434786)
      Line: (1048.342623, 198.434786),(843.503207, 223.152873)
      Line: (843.503207, 223.152873),(778.839033, 348.309401)
      Line: (778.839033, 348.309401),(788.225768, 348.309401)
```

Profiles9: (3000,1500), Shapes: 16, Rotations: 90 incremental

```
  Shape 1 (Loops: 2, Quantity: 4)
    Loop 1 (internal): 24 Primitives
      Line: (-677.378261, 221.415713),(-675.544340, 239.043940)
      Line: (-675.544340, 239.043940),(-670.042576, 263.659067)
      Line: (-670.042576, 263.659067),(-660.738634, 288.980848)
      Line: (-660.738634, 288.980848),(-647.899507, 310.632831)
      Line: (-647.899507, 310.632831),(-632.399816, 327.938869)
      Line: (-632.399816, 327.938869),(-616.159420, 340.222816)
      Line: (-616.159420, 340.222816),(-598.947519, 347.552493)
      Line: (-598.947519, 347.552493),(-581.177225, 349.995719)
      Line: (-581.177225, 349.995719),(-554.312876, 344.661407)
      Line: (-554.312876, 344.661407),(-532.146704, 328.658473)
      Line: (-532.146704, 328.658473),(-517.283892, 303.951212)
      Line: (-517.283892, 303.951212),(-512.329621, 271.566001)
      Line: (-512.329621, 271.566001),(-515.925405, 238.345949)
      Line: (-515.925405, 238.345949),(-526.712759, 203.777611)
      Line: (-526.712759, 203.777611),(-544.254302, 172.258543)
      Line: (-544.254302, 172.258543),(-566.101545, 150.412045)
      Line: (-566.101545, 150.412045),(-590.270768, 137.654385)
      Line: (-590.270768, 137.654385),(-614.376922, 133.401832)
      Line: (-614.376922, 133.401832),(-632.023501, 136.189838)
      Line: (-632.023501, 136.189838),(-648.176747, 144.553856)
      Line: (-648.176747, 144.553856),(-661.808235, 158.134199)
      Line: (-661.808235, 158.134199),(-671.810168, 176.571183)
      Line: (-671.810168, 176.571183),(-675.986238, 195.399154)
      Line: (-675.986238, 195.399154),(-677.378261, 221.415713)
    Loop 2 (external): 43 Primitives
      Line: (-739.095874, 67.026286),(-689.247606, 62.368353)
      Line: (-689.247606, 62.368353),(-688.405312, 47.281215)
      Line: (-688.405312, 47.281215),(-685.162972, 36.505273)
      Line: (-685.162972, 36.505273),(-679.124707, 29.350968)
      Line: (-679.124707, 29.350968),(-670.295967, 23.845011)
      Line: (-670.295967, 23.845011),(-654.979333, 19.549745)
      Line: (-654.979333, 19.549745),(-635.764846, 18.117990)
      Line: (-635.764846, 18.117990),(-597.276299, 23.726281)
      Line: (-597.276299, 23.726281),(-571.320852, 40.551154)
      Line: (-571.320852, 40.551154),(-559.023723, 65.491684)
      Line: (-559.023723, 65.491684),(-547.750966, 108.586042)
      Line: (-547.750966, 108.586042),(-542.725429, 131.994950)
      Line: (-542.725429, 131.994950),(-582.414232, 102.919107)
      Line: (-582.414232, 102.919107),(-624.612823, 93.227160)
      Line: (-624.612823, 93.227160),(-664.874692, 101.284406)
      Line: (-664.874692, 101.284406),(-698.037871, 125.456144)
      Line: (-698.037871, 125.456144),(-720.097749, 164.681204)
      Line: (-720.097749, 164.681204),(-727.451042, 216.960493)
      Line: (-727.451042, 216.960493),(-721.681772, 263.833180)
      Line: (-721.681772, 263.833180),(-704.373964, 306.706193)
      Line: (-704.373964, 306.706193),(-678.764276, 342.865029)
      Line: (-678.764276, 342.865029),(-648.490692, 369.193856)
      Line: (-648.490692, 369.193856),(-616.185737, 384.489576)
      Line: (-616.185737, 384.489576),(-582.792782, 389.588150)
      Line: (-582.792782, 389.588150),(-532.389149, 376.259195)
      Line: (-532.389149, 376.259195),(-495.059123, 336.272330)
      Line: (-495.059123, 336.272330),(-485.047734, 383.183492)
      Line: (-485.047734, 383.183492),(-439.543455, 383.183492)
      Line: (-439.543455, 383.183492),(-497.934558, 103.220662)
      Line: (-497.934558, 103.220662),(-509.033900, 61.746333)
      Line: (-509.033900, 61.746333),(-523.000759, 30.581500)
      Line: (-523.000759, 30.581500),(-541.593737, 7.973198)
      Line: (-541.593737, 7.973198),(-566.170110, -8.793312)
      Line: (-566.170110, -8.793312),(-596.048235, -19.177521)
      Line: (-596.048235, -19.177521),(-630.145143, -22.638925)
      Line: (-630.145143, -22.638925),(-662.667950, -20.497999)
      Line: (-662.667950, -20.497999),(-690.415071, -14.075222)
      Line: (-690.415071, -14.075222),(-712.369865, -3.259548)
      Line: (-712.369865, -3.259548),(-727.917021, 12.060068)
      Line: (-727.917021, 12.060068),(-737.174523, 30.734281)
      Line: (-737.174523, 30.734281),(-740.260357, 51.613745)
      Line: (-740.260357, 51.613745),(-739.969237, 59.016971)
      Line: (-739.969237, 59.016971),(-739.095874, 67.026286)

  Shape 2 (Loops: 1, Quantity: 4)
    Loop 1 (external): 6 Primitives
      Line: (-727.937351, 911.991882),(-727.937351, 1110.536278)
      Line: (-727.937351, 1110.536278),(-687.471557,
1110.536278)
      Line: (-687.471557, 1110.536278),(-687.471557, 945.761897)
      Line: (-687.471557, 945.761897),(-587.325997, 945.761897)
      Line: (-587.325997, 945.761897),(-587.325997, 911.991882)
      Line: (-587.325997, 911.991882),(-727.937351, 911.991882)

  Shape 3 (Loops: 1, Quantity: 4)
    Loop 1 (external): 38 Primitives
      Line: (-320.620115, 971.301784),(-315.903134, 970.195269)
      Line: (-315.903134, 970.195269),(-332.884266, 916.270338)
      Line: (-332.884266, 916.270338),(-487.601247, 916.270338)
      Line: (-487.601247, 916.270338),(-487.601247, 921.616250)
      Line: (-487.601247, 921.616250),(-480.204316, 921.616250)
      Line: (-480.204316, 921.616250),(-469.371360, 923.678791)
      Line: (-469.371360, 923.678791),(-462.087687, 929.866413)
      Line: (-462.087687, 929.866413),(-459.760593, 937.714363)
      Line: (-459.760593, 937.714363),(-458.984895, 951.804929)
      Line: (-458.984895, 951.804929),(-458.984895, 1080.735747)
      Line: (-458.984895, 1080.735747),(-460.037222, 1096.139998)
      Line: (-460.037222, 1096.139998),(-463.194203, 1104.359294)
      Line: (-463.194203, 1104.359294),(-470.297851, 1109.283143)
      Line: (-470.297851, 1109.283143),(-480.204316, 1110.924426)
      Line: (-480.204316, 1110.924426),(-487.601247, 1110.924426)
      Line: (-487.601247, 1110.924426),(-487.601247, 1116.270338)
      Line: (-487.601247, 1116.270338),(-397.349675, 1116.270338)
      Line: (-397.349675, 1116.270338),(-397.349675, 1110.924426)
      Line: (-397.349675, 1110.924426),(-410.879461, 1110.223193)
      Line: (-410.879461, 1110.223193),(-419.625197, 1107.926288)
      Line: (-419.625197, 1107.926288),(-425.046228, 1104.499772)
      Line: (-425.046228, 1104.499772),(-428.385146, 1100.409709)
      Line: (-428.385146, 1100.409709),(-430.108542, 1092.359168)
      Line: (-430.108542, 1092.359168),(-430.683008, 1077.905558)
      Line: (-430.683008, 1077.905558),(-430.683008, 951.686891)
      Line: (-430.683008, 951.686891),(-430.106932, 941.347898)
      Line: (-430.106932, 941.347898),(-428.378706, 934.884979)
      Line: (-428.378706, 934.884979),(-425.986530, 932.167621)
      Line: (-425.986530, 932.167621),(-422.957121, 930.253332)
      Line: (-422.957121, 930.253332),(-415.293373, 929.200049)
      Line: (-415.293373, 929.200049),(-399.736593, 928.848954)
      Line: (-399.736593, 928.848954),(-385.296945, 928.848954)
      Line: (-385.296945, 928.848954),(-365.715033, 929.673357)
      Line: (-365.715033, 929.673357),(-352.958530, 932.146564)
      Line: (-352.958530, 932.146564),(-344.097549, 936.840602)
      Line: (-344.097549, 936.840602),(-335.985448, 944.110740)
      Line: (-335.985448, 944.110740),(-328.329473, 955.372200)
      Line: (-328.329473, 955.372200),(-320.620115, 971.301784)


  Shape 4 (Loops: 1, Quantity: 4)
    Loop 1 (external): 12 Primitives
      Line: (-219.013076, 1404.317897),(-219.013076, 1504.317897)
      Line: (-219.013076, 1504.317897),(-205.767079, 1504.317897)
      Line: (-205.767079, 1504.317897),(-205.767079, 1463.269862)
      Line: (-205.767079, 1463.269862),(-153.802013, 1463.269862)
      Line: (-153.802013, 1463.269862),(-153.802013, 1504.317897)
      Line: (-153.802013, 1504.317897),(-140.556016, 1504.317897)
      Line: (-140.556016, 1504.317897),(-140.556016, 1404.317897)
      Line: (-140.556016, 1404.317897),(-153.802013, 1404.317897)
      Line: (-153.802013, 1404.317897),(-153.802013, 1451.479469)
      Line: (-153.802013, 1451.479469),(-205.767079, 1451.479469)
      Line: (-205.767079, 1451.479469),(-205.767079, 1404.317897)
      Line: (-205.767079, 1404.317897),(-219.013076, 1404.317897)

  Shape 5 (Loops: 1, Quantity: 4)
    Loop 1 (external): 67 Primitives
      Line: (-45.880464, 1644.169894),(-45.880464, 1495.652611)
      Line: (-45.880464, 1495.652611),(-23.885520, 1517.801102)
      Line: (-23.885520, 1517.801102),(-7.053898, 1530.260460)
      Line: (-7.053898, 1530.260460),(7.280781, 1536.093479)
      Line: (7.280781, 1536.093479),(21.697951, 1538.037818)
      Line: (21.697951, 1538.037818),(37.685989, 1535.626611)
      Line: (37.685989, 1535.626611),(51.349725, 1528.392988)
      Line: (51.349725, 1528.392988),(62.175649, 1516.200875)
      Line: (62.175649, 1516.200875),(69.975385, 1498.589064)
      Line: (69.975385, 1498.589064),(72.941989, 1478.566045)
      Line: (72.941989, 1478.566045),(73.930857, 1446.798875)
      Line: (73.930857, 1446.798875),(73.930857, 1375.301969)
      Line: (73.930857, 1375.301969),(74.684291, 1359.099630)
      Line: (74.684291, 1359.099630),(76.944593, 1348.988309)
      Line: (76.944593, 1348.988309),(79.964140, 1344.382422)
      Line: (79.964140, 1344.382422),(84.489348, 1340.565366)
      Line: (84.489348, 1340.565366),(92.035914, 1338.315932)
      Line: (92.035914, 1338.315932),(104.119536, 1337.566120)
      Line: (104.119536, 1337.566120),(104.119536, 1329.547252)
      Line: (104.119536, 1329.547252),(4.591234, 1329.547252)
      Line: (4.591234, 1329.547252),(4.591234, 1337.566120)
      Line: (4.591234, 1337.566120),(9.082857, 1337.566120)
      Line: (9.082857, 1337.566120),(21.044744, 1338.724045)
      Line: (21.044744, 1338.724045),(28.734781, 1342.197818)
      Line: (28.734781, 1342.197818),(33.341498, 1347.265328)
      Line: (33.341498, 1347.265328),(36.378555, 1354.595554)
      Line: (36.378555, 1354.595554),(36.948442, 1361.478649)
      Line: (36.948442, 1361.478649),(37.138404, 1375.301969)
      Line: (37.138404, 1375.301969),(37.138404, 1446.755403)
      Line: (37.138404, 1446.755403),(36.252366, 1474.168837)
      Line: (36.252366, 1474.168837),(33.594253, 1490.226498)
      Line: (33.594253, 1490.226498),(29.177914, 1499.309102)
      Line: (29.177914, 1499.309102),(22.692065, 1505.797366)
      Line: (22.692065, 1505.797366),(14.421046, 1509.812611)
      Line: (14.421046, 1509.812611),(4.649197, 1511.151026)
      Line: (4.649197, 1511.151026),(-6.406124, 1509.681403)
      Line: (-6.406124, 1509.681403),(-17.912615, 1505.272535)
      Line: (-17.912615, 1505.272535),(-30.770615, 1496.622385)
      Line: (-30.770615, 1496.622385),(-45.880464, 1482.103781)
      Line: (-45.880464, 1482.103781),(-45.880464, 1375.301969)
      Line: (-45.880464, 1375.301969),(-45.339332, 1358.482837)
      Line: (-45.339332, 1358.482837),(-43.715935, 1349.438271)
      Line: (-43.715935, 1349.438271),(-40.320879, 1344.758120)
      Line: (-40.320879, 1344.758120),(-34.789898, 1340.945290)
      Line: (-34.789898, 1340.945290),(-26.424917, 1338.410913)
      Line: (-26.424917, 1338.410913),(-13.333294, 1337.566120)
      Line: (-13.333294, 1337.566120),(-13.333294, 1329.547252)
      Line: (-13.333294, 1329.547252),(-113.804992, 1329.547252)
      Line: (-113.804992, 1329.547252),(-113.804992, 1337.566120)
      Line: (-113.804992, 1337.566120),(-101.759369, 1338.578347)
      Line: (-101.759369, 1338.578347),(-92.501294, 1341.615026)
      Line: (-92.501294, 1341.615026),(-88.534917, 1344.848686)
      Line: (-88.534917, 1344.848686),(-85.396841, 1350.035554)
      Line: (-85.396841, 1350.035554),(-83.353898, 1359.433894)
      Line: (-83.353898, 1359.433894),(-82.672917, 1375.301969)
      Line: (-82.672917, 1375.301969),(-82.672917, 1559.264234)
      Line: (-82.672917, 1559.264234),(-83.106577, 1586.932649)
      Line: (-83.106577, 1586.932649),(-84.407558, 1601.443403)
      Line: (-84.407558, 1601.443403),(-86.552728, 1608.105328)
      Line: (-86.552728, 1608.105328),(-89.518954, 1612.227252)
      Line: (-89.518954, 1612.227252),(-93.478690, 1614.366196)
      Line: (-93.478690, 1614.366196),(-98.604388, 1615.079177)
      Line: (-98.604388, 1615.079177),(-104.694200, 1614.215064)
      Line: (-104.694200, 1614.215064),(-113.635935, 1611.622724)
      Line: (-113.635935, 1611.622724),(-116.635181, 1619.641592)
      Line: (-116.635181, 1619.641592),(-56.122577, 1644.169894)
      Line: (-56.122577, 1644.169894),(-45.880464, 1644.169894)

Shape 6 (Loops: 1, Quantity: 4)
   Loop 1 (external): 13 Primitives
      Line: (-303.830940, 523.279099),(-351.129642, 723.279099)
      Line: (-351.129642, 723.279099),(-309.623546, 723.279099)
      Line: (-309.623546, 723.279099),(-279.709647, 586.048939)
      Line: (-279.709647, 586.048939),(-243.325508, 723.279099)
      Line: (-243.325508, 723.279099),(-194.957717, 723.279099)
      Line: (-194.957717, 723.279099),(-160.058992, 583.779594)
      Line: (-160.058992, 583.779594),(-129.524099, 723.279099)
      Line: (-129.524099, 723.279099),(-88.829787, 723.279099)
      Line: (-88.829787, 723.279099),(-137.051505, 523.279099)
      Line: (-137.051505, 523.279099),(-180.642119, 523.279099)
      Line: (-180.642119, 523.279099),(-220.029880, 672.740339)
      Line: (-220.029880, 672.740339),(-259.622311, 523.279099)
      Line: (-259.622311, 523.279099),(-303.830940, 523.279099)

Shape 7 (Loops: 1, Quantity: 4)
   Loop 1 (external): 19 Primitives
      Line: (-1.353104, 520.752469),(-45.600487, 665.730635)
      Line: (-45.600487, 665.730635),(-20.023590, 665.730635)
      Line: (-20.023590, 665.730635),(3.260113, 582.033913)
      Line: (3.260113, 582.033913),(11.494966, 550.857156)
      Line: (11.494966, 550.857156),(13.651542, 559.497878)
      Line: (13.651542, 559.497878),(19.028334, 580.774035)
      Line: (19.028334, 580.774035),(42.011973, 665.730635)
      Line: (42.011973, 665.730635),(67.125871, 665.730635)
      Line: (67.125871, 665.730635),(89.093056, 581.715962)
      Line: (89.093056, 581.715962),(96.175350, 553.830967)
      Line: (96.175350, 553.830967),(104.407222, 581.917651)
      Line: (104.407222, 581.917651),(129.357161, 665.730635)
      Line: (129.357161, 665.730635),(153.235030, 665.730635)
      Line: (153.235030, 665.730635),(107.882785, 520.752469)
      Line: (107.882785, 520.752469),(82.528445, 520.752469)
      Line: (82.528445, 520.752469),(59.461336, 607.774769)
      Line: (59.461336, 607.774769),(53.671711, 632.460417)
      Line: (53.671711, 632.460417),(24.316206, 520.752469)
      Line: (24.316206, 520.752469),(-1.353104, 520.752469)

Shape 8 (Loops: 1, Quantity: 4)
   Loop 1 (external): 28 Primitives
      Line: (344.874051, 934.129998),(349.677545, 901.352077)
      Line: (349.677545, 901.352077),(334.898575, 898.945474)
      Line: (334.898575, 898.945474),(321.797021, 898.143274)
      Line: (321.797021, 898.143274),(303.951221, 899.710225)
      Line: (303.951221, 899.710225),(290.561667, 904.411081)
      Line: (290.561667, 904.411081),(281.189405, 911.885527)
      Line: (281.189405, 911.885527),(275.094488, 921.069806)
      Line: (275.094488, 921.069806),(271.752304, 937.158819)
      Line: (271.752304, 937.158819),(270.638243, 964.644025)
      Line: (270.638243, 964.644025),(270.638243, 1089.846330)
      Line: (270.638243, 1089.846330),(243.564007, 1089.846330)
      Line: (243.564007, 1089.846330),(243.564007, 1118.667291)
      Line: (243.564007, 1118.667291),(270.638243, 1118.667291)
      Line: (270.638243, 1118.667291),(270.638243, 1172.459570)
      Line: (270.638243, 1172.459570),(307.319466, 1194.472496)
      Line: (307.319466, 1194.472496),(307.319466, 1118.667291)
      Line: (307.319466, 1118.667291),(344.874051, 1118.667291)
      Line: (344.874051, 1118.667291),(344.874051, 1089.846330)
      Line: (344.874051, 1089.846330),(307.319466, 1089.846330)
      Line: (307.319466, 1089.846330),(307.319466, 962.594767)
      Line: (307.319466, 962.594767),(307.793038, 949.670191)
      Line: (307.793038, 949.670191),(309.213754, 942.391169)
      Line: (309.213754, 942.391169),(311.894383, 938.447623)
      Line: (311.894383, 938.447623),(315.685719, 935.529474)
      Line: (315.685719, 935.529474),(321.081457, 933.363186)
      Line: (321.081457, 933.363186),(328.414313, 932.641090)
      Line: (328.414313, 932.641090),(335.621789, 933.013317)
      Line: (335.621789, 933.013317),(344.874051, 934.129998)

Shape 9 (Loops: 1, Quantity: 4)
   Loop 1 (external): 8 Primitives
      Line: (405.382646, 981.289111),(405.382646, 1069.498718)
      Line: (405.382646, 1069.498718),(372.485994, 1069.498718)
      Line: (372.485994, 1069.498718),(372.485994, 1081.289111)
      Line: (372.485994, 1081.289111),(451.525295, 1081.289111)
      Line: (451.525295, 1081.289111),(451.525295, 1069.498718)
      Line: (451.525295, 1069.498718),(418.628643, 1069.498718)
      Line: (418.628643, 1069.498718),(418.628643, 981.289111)
      Line: (418.628643, 981.289111),(405.382646, 981.289111)

Shape 10 (Loops: 1, Quantity: 4)
   Loop 1 (external): 4 Primitives
      Line: (541.980923, 1481.914894),(541.980923, 1581.914894)
      Line: (541.980923, 1581.914894),(555.226920, 1581.914894)
      Line: (555.226920, 1581.914894),(555.226920, 1481.914894)
      Line: (555.226920, 1481.914894),(541.980923, 1481.914894)

Shape 11 (Loops: 1, Quantity: 4)
   Loop 1 (external): 32 Primitives
      Line: (687.322990, 1424.745161),(687.322990, 1419.399249)
      Line: (687.322990, 1419.399249),(601.788399, 1419.399249)
      Line: (601.788399, 1419.399249),(601.788399, 1424.745161)
      Line: (601.788399, 1424.745161),(608.812902, 1424.745161)
      Line: (608.812902, 1424.745161),(619.480047, 1426.562192)
      Line: (619.480047, 1426.562192),(626.785732, 1432.013287)
      Line: (626.785732, 1432.013287),(629.499997, 1440.116406)
      Line: (629.499997, 1440.116406),(630.404751, 1454.933840)
      Line: (630.404751, 1454.933840),(630.404751, 1583.864658)
      Line: (630.404751, 1583.864658),(629.901129, 1596.531828)
      Line: (629.901129, 1596.531828),(628.390261, 1604.054847)
      Line: (628.390261, 1604.054847),(626.184701, 1607.644381)
      Line: (626.184701, 1607.644381),(622.367116, 1610.414997)
      Line: (622.367116, 1610.414997),(615.796877, 1613.143752)
      Line: (615.796877, 1613.143752),(608.812902, 1614.053337)
      Line: (608.812902, 1614.053337),(601.788399, 1614.053337)
      Line: (601.788399, 1614.053337),(601.788399, 1619.399249)
      Line: (601.788399, 1619.399249),(687.322990, 1619.399249)
      Line: (687.322990, 1619.399249),(687.322990, 1614.053337)
      Line: (687.322990, 1614.053337),(680.380097, 1614.053337)
      Line: (680.380097, 1614.053337),(669.626009, 1612.209211)
      Line: (669.626009, 1612.209211),(662.304223, 1606.676834)
      Line: (662.304223, 1606.676834),(659.606034, 1598.600809)
      Line: (659.606034, 1598.600809),(658.706638, 1583.864658)
      Line: (658.706638, 1583.864658),(658.706638, 1454.933840)
      Line: (658.706638, 1454.933840),(659.210537, 1442.185387)
      Line: (659.210537, 1442.185387),(660.722235, 1434.635274)
      Line: (660.722235, 1434.635274),(662.858563, 1431.176155)
      Line: (662.858563, 1431.176155),(666.706739, 1428.275123)
      Line: (666.706739, 1428.275123),(673.276173, 1425.627652)
      Line: (673.276173, 1425.627652),(680.380097, 1424.745161)
      Line: (680.380097, 1424.745161),(687.322990, 1424.745161)

Shape 12 (Loops: 2, Quantity: 2)
   Loop 1 (internal): 13 Primitives
      Line: (763.797818, 619.325467),(763.797818, 751.494317)
      Line: (763.797818, 751.494317),(890.436397, 751.494317)
      Line: (890.436397, 751.494317),(929.032496, 746.897496)
      Line: (929.032496, 746.897496),(955.584857, 733.107033)
      Line: (955.584857, 733.107033),(971.133224, 712.365909)
      Line: (971.133224, 712.365909),(976.316013, 686.515778)
      Line: (976.316013, 686.515778),(973.618930, 667.986381)
      Line: (973.618930, 667.986381),(965.527681, 651.005001)
      Line: (965.527681, 651.005001),(952.477410, 636.496658)
      Line: (952.477410, 636.496658),(934.501934, 626.646247)
      Line: (934.501934, 626.646247),(910.272857, 621.155662)
      Line: (910.272857, 621.155662),(877.603233, 619.325467)
      Line: (877.603233, 619.325467),(763.797818, 619.325467)
   Loop 2 (external): 30 Primitives
      Line: (710.813830, 395.744681),(710.813830, 795.744681)
      Line: (710.813830, 795.744681),(888.457148, 795.744681)
      Line: (888.457148, 795.744681),(935.511914, 793.027476)
      Line: (935.511914, 793.027476),(969.662994, 784.875860)
      Line: (969.662994, 784.875860),(994.773399, 770.078864)
      Line: (994.773399, 770.078864),(1014.304811, 746.916873)
      Line: (1014.304811, 746.916873),(1026.861247, 718.186090)
      Line: (1026.861247, 718.186090),(1031.046726, 686.575400)
      Line: (1031.046726, 686.575400),(1024.117212, 647.406981)
      Line: (1024.117212, 647.406981),(1003.328671, 614.920925)
      Line: (1003.328671, 614.920925),(968.277166, 591.114835)
      Line: (968.277166, 591.114835),(918.157434, 577.986311)
      Line: (918.157434, 577.986311),(936.660001, 567.839330)
      Line: (936.660001, 567.839330),(950.174761, 557.882113)
      Line: (950.174761, 557.882113),(972.599029, 534.121484)
      Line: (972.599029, 534.121484),(993.920391, 504.713298)
      Line: (993.920391, 504.713298),(1063.236956, 395.744681)
      Line: (1063.236956, 395.744681),(996.998540, 395.744681)
      Line: (996.998540, 395.744681),(943.996665, 479.482206)
      Line: (943.996665, 479.482206),(922.884444, 511.063086)
      Line: (922.884444, 511.063086),(906.043640, 534.084733)
      Line: (906.043640, 534.084733),(891.621981, 550.340547)
      Line: (891.621981, 550.340547),(878.812432, 560.793321)
      Line: (878.812432, 560.793321),(866.891617, 567.193508)
```

```
         Line: (866.891617, 567.193508),(854.734831, 571.291557)
         Line: (854.734831, 571.291557),(842.781504, 572.819173)
         Line: (842.781504, 572.819173),(825.159751, 573.328378)
         Line: (825.159751, 573.328378),(763.797818, 573.328378)
         Line: (763.797818, 573.328378),(763.797818, 395.744681)
         Line: (763.797818, 395.744681),(710.813830, 395.744681)

   Shape 13 (Loops: 2, Quantity: 2)
      Loop 1 (internal): 17 Primitives
         Line: (1232.136879, 612.351320),(1260.523628, 747.431378)
         Line: (1260.523628, 747.431378),(1390.411208, 747.431378)
         Line: (1390.411208, 747.431378),(1416.705986, 746.289253)
         Line: (1416.705986, 746.289253),(1433.753462, 742.862877)
         Line: (1433.753462, 742.862877),(1445.336437, 736.115442)
         Line: (1445.336437, 736.115442),(1454.514428, 725.010138)
         Line: (1454.514428, 725.010138),(1460.496053, 710.946906)
         Line: (1460.496053, 710.946906),(1462.489927, 694.602406)
         Line: (1462.489927, 694.602406),(1459.792844, 674.465649)
         Line: (1459.792844, 674.465649),(1451.701596, 655.726365)
         Line: (1451.701596, 655.726365),(1438.931267, 639.546056)
         Line: (1438.931267, 639.546056),(1421.795616, 627.086225)
         Line: (1421.795616, 627.086225),(1399.585753, 618.393823)
         Line: (1399.585753, 618.393823),(1371.994114, 613.515803)
         Line: (1371.994114, 613.515803),(1347.716315, 612.642441)
         Line: (1347.716315, 612.642441),(1305.576833, 612.351320)
         Line: (1305.576833, 612.351320),(1232.136879, 612.351320)
      Loop 2 (external): 28 Primitives
         Line: (1132.358923, 391.099500),(1216.201718, 791.099500)
         Line: (1216.201718, 791.099500),(1383.603173, 791.099500)
         Line: (1383.603173, 791.099500),(1427.173433, 788.983914)
         Line: (1427.173433, 788.983914),(1458.771965, 782.637154)
         Line: (1458.771965, 782.637154),(1481.624204, 770.798318)
         Line: (1481.624204, 770.798318),(1498.955581, 751.805177)
         Line: (1498.955581, 751.805177),(1510.034288, 725.595361)
         Line: (1510.034288, 725.595361),(1513.727191, 693.581014)
         Line: (1513.727191, 693.581014),(1505.803348, 648.912601)
         Line: (1505.803348, 648.912601),(1482.031820, 612.604711)
         Line: (1482.031820, 612.604711),(1439.959878, 586.142761)
         Line: (1439.959878, 586.142761),(1378.315302, 571.012164)
         Line: (1378.315302, 571.012164),(1397.789607, 554.938429)
         Line: (1397.789607, 554.938429),(1411.389794, 539.143890)
         Line: (1411.389794, 539.143890),(1433.899488, 503.312694)
         Line: (1433.899488, 503.312694),(1451.510807, 466.506778)
         Line: (1451.510807, 466.506778),(1482.345764, 391.099500)
         Line: (1482.345764, 391.099500),(1422.500082, 391.099500)
         Line: (1422.500082, 391.099500),(1393.785415, 465.336333)
         Line: (1393.785415, 465.336333),(1377.240251, 503.012117)
         Line: (1377.240251, 503.012117),(1357.972617, 535.109934)
         Line: (1357.972617, 535.109934),(1344.617066, 552.129835)
         Line: (1344.617066, 552.129835),(1330.622166, 562.074895)
         Line: (1330.622166, 562.074895),(1312.222679, 567.031122)
         Line: (1312.222679, 567.031122),(1285.144716, 568.683198)
         Line: (1285.144716, 568.683198),(1222.928332, 568.683198)
         Line: (1222.928332, 568.683198),(1185.694679, 391.099500)
         Line: (1185.694679, 391.099500),(1132.358923, 391.099500)

   Shape 14 (Loops: 1, Quantity: 4)
      Loop 1 (external): 20 Primitives
         Line: (1608.863518, 441.554308),(1608.863518, 586.532474)
         Line: (1608.863518, 586.532474),(1630.988699, 586.532474)
         Line: (1630.988699, 586.532474),(1630.988699, 564.496725)
         Line: (1630.988699, 564.496725),(1639.224554, 577.278093)
         Line: (1639.224554, 577.278093),(1646.570138, 584.817051)
         Line: (1646.570138, 584.817051),(1654.180012, 588.505365)
         Line: (1654.180012, 588.505365),(1662.578801, 589.734803)
         Line: (1662.578801, 589.734803),(1674.991424, 587.792096)
         Line: (1674.991424, 587.792096),(1687.757258, 581.963973)
         Line: (1687.757258, 581.963973),(1678.884548, 559.208853)
         Line: (1678.884548, 559.208853),(1669.954196, 563.107682)
         Line: (1669.954196, 563.107682),(1661.153008, 564.407292)
         Line: (1661.153008, 564.407292),(1653.379430, 563.186797)
         Line: (1653.379430, 563.186797),(1646.522440, 559.525313)
         Line: (1646.522440, 559.525313),(1641.107896, 553.727118)
         Line: (1641.107896, 553.727118),(1637.661654, 546.096492)
         Line: (1637.661654, 546.096492),(1634.403663, 532.367118)
         Line: (1634.403663, 532.367118),(1633.317666, 517.379869)
         Line: (1633.317666, 517.379869),(1633.317666, 441.554308)
         Line: (1633.317666, 441.554308),(1608.863518, 441.554308)

   Shape 15 (Loops: 2, Quantity: 1)
      Loop 1 (internal): 9 Primitives
         Line: (1260.398733, 1357.025795),(1273.602588, 1416.977096)
         Line: (1273.602588, 1416.977096),(1304.770623, 1463.536327)
         Line: (1304.770623, 1463.536327),(1349.941837, 1493.697096)
         Line: (1349.941837, 1493.697096),(1404.452905, 1503.750685)
         Line: (1404.452905, 1503.750685),(1463.990361, 1491.162831)
         Line: (1463.990361, 1491.162831),(1511.888457, 1453.399268)
         Line: (1511.888457, 1453.399268),(1532.785016, 1413.303920)
         Line: (1532.785016, 1413.303920),(1543.659287, 1357.025795)
         Line: (1543.659287, 1357.025795),(1260.398733, 1357.025795)
      Loop 2 (external): 28 Primitives
         Line: (1543.057715, 1219.471209),(1631.411857, 1208.263058)
         Line: (1631.411857, 1208.263058),(1601.320620, 1138.923728)
         Line: (1601.320620, 1138.923728),(1553.340847, 1087.172962)
         Line: (1553.340847, 1087.172962),(1488.618378, 1054.955600)
         Line: (1488.618378, 1054.955600),(1408.299054, 1044.216479)
         Line: (1408.299054, 1044.216479),(1308.581965, 1061.321259)
         Line: (1308.581965, 1061.321259),(1231.868894, 1112.635600)
         Line: (1231.868894, 1112.635600),(1182.960597, 1194.827134)
         Line: (1182.960597, 1194.827134),(1166.657831, 1304.563495)
         Line: (1166.657831, 1304.563495),(1183.077977, 1418.257271)
         Line: (1183.077977, 1418.257271),(1232.338413, 1503.175198)
         Line: (1232.338413, 1503.175198),(1308.122961, 1556.336071)
         Line: (1308.122961, 1556.336071),(1403.413115, 1574.056362)
         Line: (1403.413115, 1574.056362),(1495.859496, 1556.640362)
         Line: (1495.859496, 1556.640362),(1569.601878, 1504.392362)
         Line: (1569.601878, 1504.392362),(1618.158035, 1421.008036)
         Line: (1618.158035, 1421.008036),(1634.343421, 1309.480735)
         Line: (1634.343421, 1309.480735),(1634.177377, 1300.104362)
         Line: (1634.177377, 1300.104362),(1633.679246, 1286.720118)
         Line: (1633.679246, 1286.720118),(1255.304119, 1286.720118)
         Line: (1255.304119, 1286.720118),(1269.366070, 1213.199046)
         Line: (1269.366070, 1213.199046),(1302.482367, 1158.695803)
         Line: (1302.482367, 1158.695803),(1350.328701, 1125.565568)
         Line: (1350.328701, 1125.565568),(1408.580766, 1114.522156)
         Line: (1408.580766, 1114.522156),(1452.846116, 1120.730901)
         Line: (1452.846116, 1120.730901),(1489.279129, 1139.357137)
         Line: (1489.279129, 1139.357137),(1519.486442, 1172.401294)
         Line: (1519.486442, 1172.401294),(1543.057715, 1219.471209)

   Shape 16 (Loops: 1, Quantity: 4)
      Loop 1 (external): 12 Primitives
         Line: (1719.395136, 1384.292866),(1719.395136, 1484.292866)
         Line: (1719.395136, 1484.292866),(1791.593098, 1484.292866)
         Line: (1791.593098, 1484.292866),(1791.593098, 1472.502473)
         Line: (1791.593098, 1472.502473),(1732.641133, 1472.502473)
         Line: (1732.641133, 1472.502473),(1732.641133, 1441.934787)
         Line: (1732.641133, 1441.934787),(1787.808528, 1441.934787)
         Line: (1787.808528, 1441.934787),(1787.808528, 1430.144394)
         Line: (1787.808528, 1430.144394),(1732.641133, 1430.144394)
         Line: (1732.641133, 1430.144394),(1732.641133, 1396.083259)
         Line: (1732.641133, 1396.083259),(1793.922065, 1396.083259)
         Line: (1793.922065, 1396.083259),(1793.922065, 1384.292866)
         Line: (1793.922065, 1384.292866),(1719.395136, 1384.292866)
```

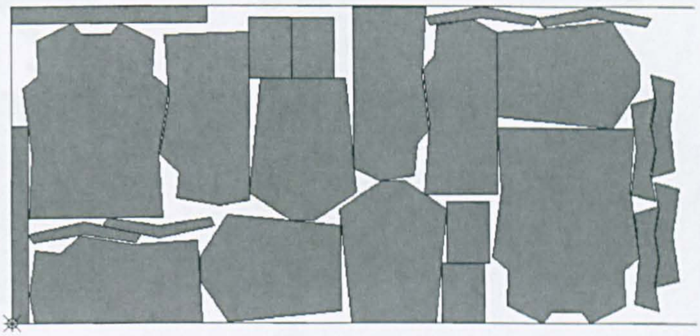
Profiles10: (15000,3000), Shapes: 13, Rotations: 0 absolute

```
   Shape 1 (Loops: 1, Quantity: 7)
      Loop 1 (external): 8 Primitives
         Line: (200.000000, -0.000000),(100.000000, 200.000000)
         Line: (100.000000, 200.000000),(0.000000, 300.000000)
         Line: (0.000000, 300.000000),(100.000000, 400.000000)
         Line: (100.000000, 400.000000),(100.000000, 500.000000)
         Line: (100.000000, 500.000000),(200.000000, 700.000000)
         Line: (200.000000, 700.000000),(900.000000, 500.000000)
         Line: (900.000000, 500.000000),(900.000000, 100.000000)
         Line: (900.000000, 100.000000),(200.000000, -0.000000)

   Shape 2 (Loops: 1, Quantity: 7)
      Loop 1 (external): 6 Primitives
         Line: (74.000000, -0.000000),(0.000000, 125.000000)
         Line: (0.000000, 125.000000),(870.000000, 305.000000)
         Line: (870.000000, 305.000000),(1740.000000, 125.000000)
         Line: (1740.000000, 125.000000),(1666.000000, -0.000000)
         Line: (1666.000000, -0.000000),(870.000000, 119.000000)
         Line: (870.000000, 119.000000),(74.000000, -0.000000)

   Shape 3 (Loops: 1, Quantity: 7)
      Loop 1 (external): 11 Primitives
         Line: (-0.000000, 200.000000),(-0.000000, 600.000000)
         Line: (-0.000000, 600.000000),(200.000000, 600.000000)
         Line: (200.000000, 600.000000),(200.000000, 300.000000)
         Line: (200.000000, 300.000000),(800.000000, 300.000000)
         Line: (800.000000, 300.000000),(800.000000, 600.000000)
         Line: (800.000000, 600.000000),(1100.000000, 600.000000)
         Line: (1100.000000, 600.000000),(1100.000000, 400.000000)
         Line: (1100.000000, 400.000000),(700.000000, -0.000000)
         Line: (700.000000, -0.000000),(600.000000, -0.000000)
         Line: (600.000000, -0.000000),(600.000000, 200.000000)
         Line: (600.000000, 200.000000),(-0.000000, 200.000000)

   Shape 4 (Loops: 1, Quantity: 7)
      Loop 1 (external): 8 Primitives
         Line: (-0.000000, 0.000000),(100.000000, 300.000000)
         Line: (100.000000, 300.000000),(0.000000, 500.000000)
         Line: (0.000000, 500.000000),(200.000000, 400.000000)
         Line: (200.000000, 400.000000),(400.000000, 500.000000)
         Line: (400.000000, 500.000000),(300.000000, 200.000000)
         Line: (300.000000, 200.000000),(400.000000, 0.000000)
         Line: (400.000000, 0.000000),(200.000000, 100.000000)
         Line: (200.000000, 100.000000),(-0.000000, 0.000000)

   Shape 5 (Loops: 1, Quantity: 7)
      Loop 1 (external): 8 Primitives
         Line: (0.000000, 0.000000),(200.000000, 200.000000)
         Line: (200.000000, 200.000000),(200.000000, 400.000000)
         Line: (200.000000, 400.000000),(0.000000, 600.000000)
         Line: (0.000000, 600.000000),(600.000000, 600.000000)
         Line: (600.000000, 600.000000),(400.000000, 400.000000)
         Line: (400.000000, 400.000000),(400.000000, 200.000000)
         Line: (400.000000, 200.000000),(600.000000, 0.000000)
         Line: (600.000000, 0.000000),(0.000000, 0.000000)

   Shape 6 (Loops: 1, Quantity: 7)
      Loop 1 (external): 4 Primitives
         Line: (0.000000, 0.000000),(0.000000, 660.000000)
         Line: (0.000000, 660.000000),(200.000000, 260.000000)
         Line: (200.000000, 260.000000),(390.000000, 230.000000)
         Line: (390.000000, 230.000000),(0.000000, 0.000000)


   Shape 7 (Loops: 1, Quantity: 7)
      Loop 1 (external): 14 Primitives
         Line: (0.000000, 232.000000),(0.000000, 426.000000)
```

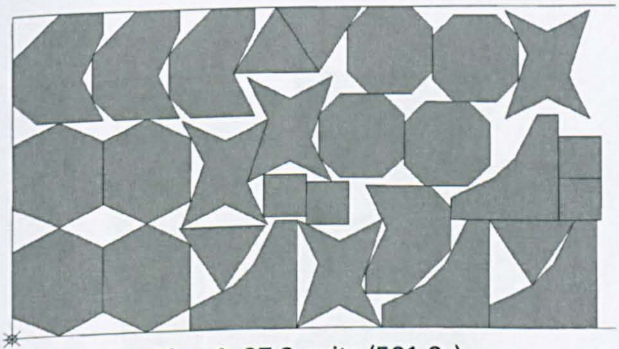
```
          Line: (0.000000, 426.000000),(539.000000, 414.000000)
          Line: (539.000000, 414.000000),(616.000000, 477.000000)
          Line: (616.000000, 477.000000),(694.000000, 544.000000)
          Line: (694.000000, 544.000000),(732.000000, 547.000000)
          Line: (732.000000, 547.000000),(761.000000, 490.000000)
          Line: (761.000000, 490.000000),(845.000000, 453.000000)
          Line: (845.000000, 453.000000),(884.000000, 463.000000)
          Line: (884.000000, 463.000000),(1003.000000, 500.000000)
          Line: (1003.000000, 500.000000),(1097.000000, 260.000000)
          Line: (1097.000000, 260.000000),(975.000000, 258.000000)
          Line: (975.000000, 258.000000),(894.000000, 20.000000)
          Line: (894.000000, 20.000000),(827.000000, -0.000000)
          Line: (827.000000, -0.000000),(0.000000, 232.000000)



Shape 8 (Loops: 1, Quantity: 7)
       Loop 1 (external): 15 Primitives
          Line: (882.000000, 623.000000),(963.000000, 713.000000)
          Line: (963.000000, 713.000000),(1147.000000, 666.000000)
          Line: (1147.000000, 666.000000),(1098.000000, 518.000000)
          Line: (1098.000000, 518.000000),(1052.000000, 396.000000)
          Line: (1052.000000, 396.000000),(998.000000, 265.000000)
          Line: (998.000000, 265.000000),(954.000000, 169.000000)
          Line: (954.000000, 169.000000),(868.000000, 0.000000)
          Line: (868.000000, 0.000000),(689.000000, 49.000000)
          Line: (689.000000, 49.000000),(592.000000, 148.000000)
          Line: (592.000000, 148.000000),(4.000000, 165.000000)
          Line: (4.000000, 165.000000),(0.000000, 325.000000)
          Line: (0.000000, 325.000000),(546.000000, 334.000000)
          Line: (546.000000, 334.000000),(688.000000, 402.000000)
          Line: (688.000000, 402.000000),(780.000000, 508.000000)
          Line: (780.000000, 508.000000),(882.000000, 623.000000)

   Shape 9 (Loops: 1, Quantity: 7)
       Loop 1 (external): 6 Primitives
          Line: (0.000000, 0.000000),(10.000000, 50.000000)
          Line: (10.000000, 50.000000),(30.000000, 300.000000)
          Line: (30.000000, 300.000000),(470.000000, 390.000000)
          Line: (470.000000, 390.000000),(400.000000, 200.000000)
```

```
      Line: (400.000000, 200.000000),(520.000000, 70.000000)
      Line: (520.000000, 70.000000),(0.000000, 0.000000)

Shape 10 (Loops: 1, Quantity: 7)
   Loop 1 (external): 10 Primitives
      Line: (100.000000, 0.000000),(-0.000000, 600.000000)
      Line: (-0.000000, 600.000000),(100.000000, 600.000000)
      Line: (100.000000, 600.000000),(100.000000, 500.000000)
      Line: (100.000000, 500.000000),(200.000000, 400.000000)
      Line: (200.000000, 400.000000),(300.000000, 400.000000)
      Line: (300.000000, 400.000000),(400.000000, 500.000000)
      Line: (400.000000, 500.000000),(400.000000, 600.000000)
      Line: (400.000000, 600.000000),(500.000000, 600.000000)
      Line: (500.000000, 600.000000),(400.000000, 0.000000)
      Line: (400.000000, 0.000000),(100.000000, 0.000000)

Shape 11 (Loops: 1, Quantity: 7)
   Loop 1 (external): 3 Primitives
      Line: (0.000000, 800.000000),(800.000000, 800.000000)
      Line: (800.000000, 800.000000),(400.000000, 0.000000)
      Line: (400.000000, 0.000000),(0.000000, 800.000000)

Shape 12 (Loops: 1, Quantity: 7)
   Loop 1 (external): 6 Primitives
      Line: (0.000000, 0.000000),(-0.000000, 600.000000)
      Line: (-0.000000, 600.000000),(400.000000, 600.000000)
      Line: (400.000000, 600.000000),(400.000000, 300.000000)
      Line: (400.000000, 300.000000),(600.000000, 300.000000)
      Line: (600.000000, 300.000000),(600.000000, -0.000000)
      Line: (600.000000, -0.000000),(0.000000, 0.000000)

Shape 13 (Loops: 1, Quantity: 7)
   Loop 1 (external): 8 Primitives
      Line: (100.000000, 0.000000),(0.000000, 100.000000)
      Line: (0.000000, 100.000000),(-0.000000, 300.000000)
      Line: (-0.000000, 300.000000),(200.000000, 500.000000)
      Line: (200.000000, 500.000000),(400.000000, 500.000000)
      Line: (400.000000, 500.000000),(400.000000, 400.000000)
      Line: (400.000000, 400.000000),(300.000000, 200.000000)
      Line: (300.000000, 200.000000),(400.000000, -0.000000)
      Line: (400.000000, -0.000000),(100.000000, 0.000000)
```

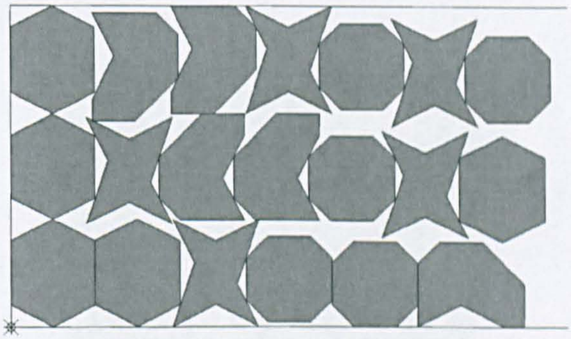
Appendix B – Best Layouts from Chapter 4



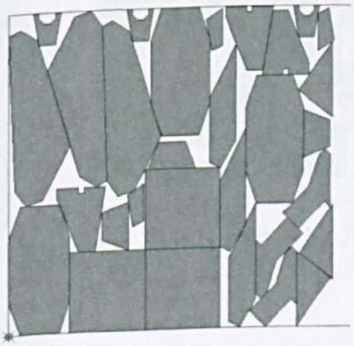
Albano: 10292.9 units (93.3s)



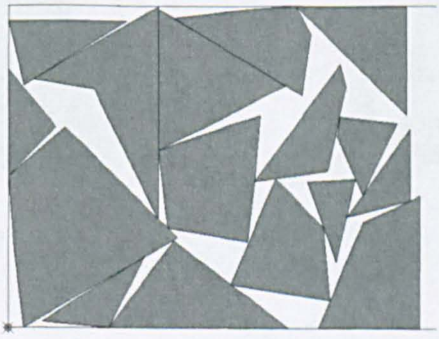
Blasz1: 27.2 units (501.9s)



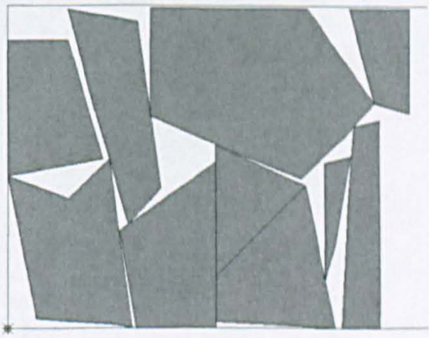
Blasz2: 25.28 units (10.9s)



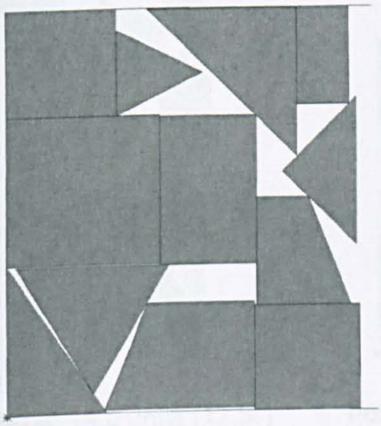
Dagli: 60.57 units (188.8s)



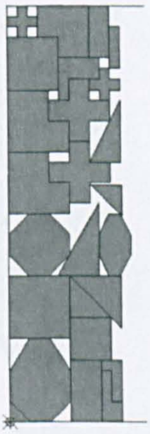
Dighe1: 1292.3 units (8.8s)



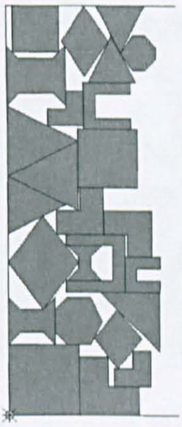
Dighe2: 1260.0 units (7.1s)



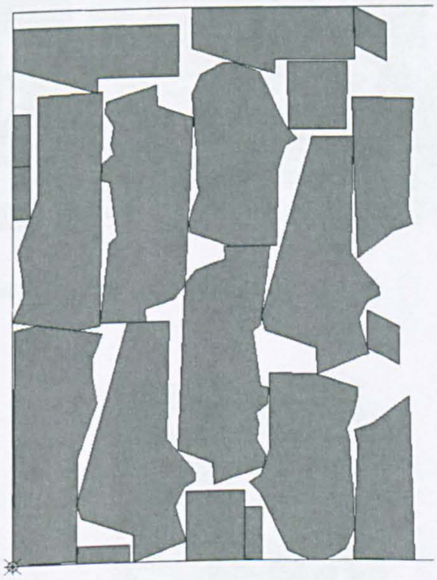
Fu: 32.8 units (20.7s)



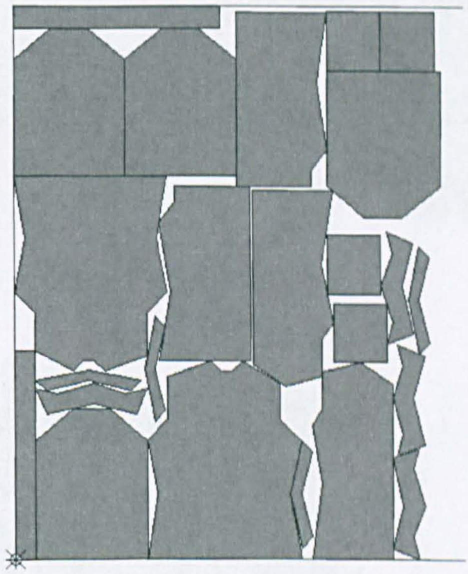
Jakobs1: 11.86 units (43.4s)



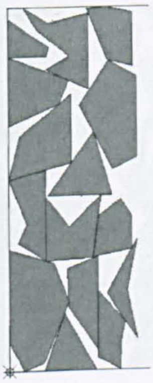
Jakobs2: 25.8 units (81.4s)



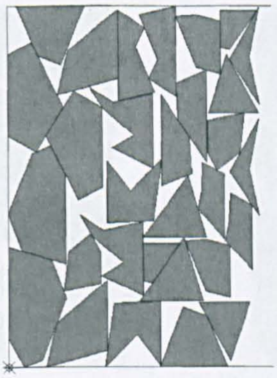
Mao: 1854.3 units (29.7s)



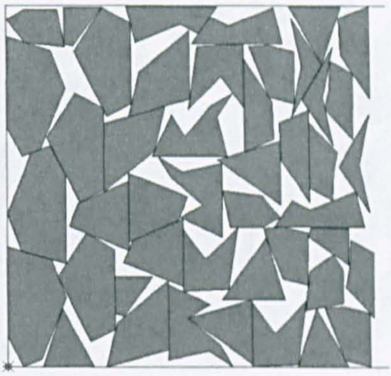
Marques: 80.0 units (4.8s)



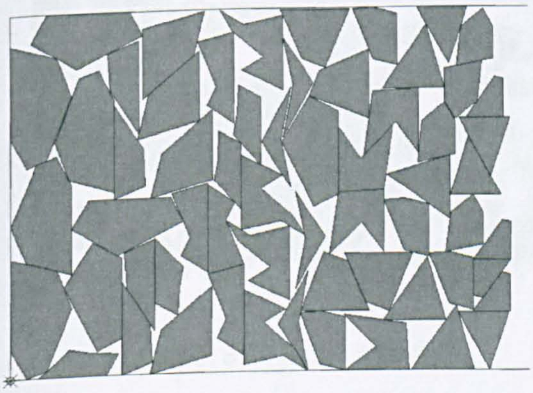
Poly1A: 14.0 units (12.5s)



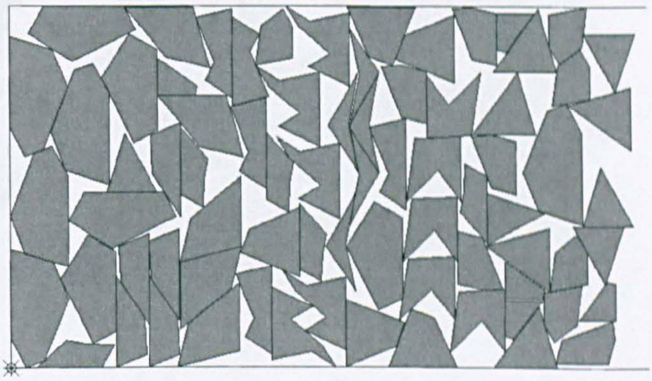
Poly2A: 28.17 units (120.6s)



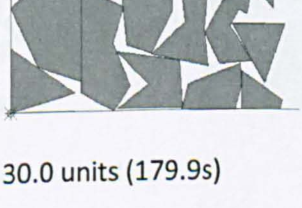
Poly3A: 40.33 units (1515.5s)



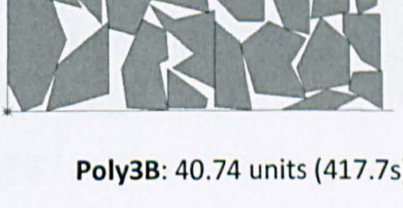
Poly4A: 54.93 units (203.2s)



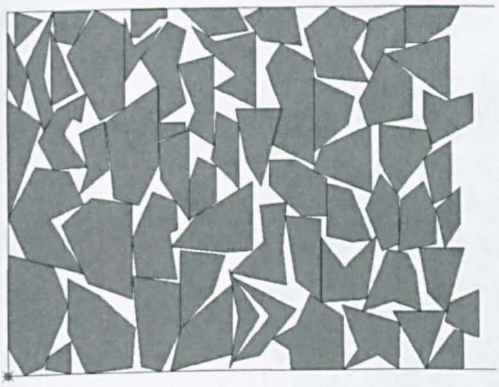
Poly5A: 69.37 units (475.6s)



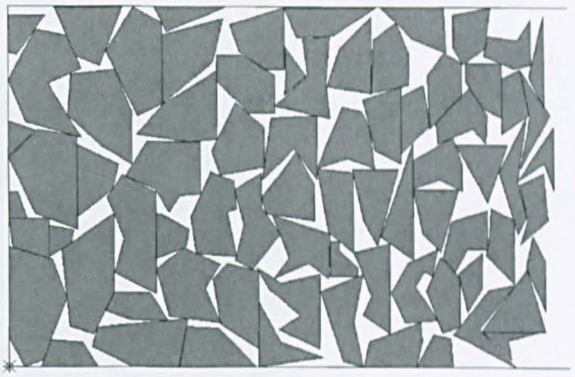
Poly2B: 30.0 units (179.9s)



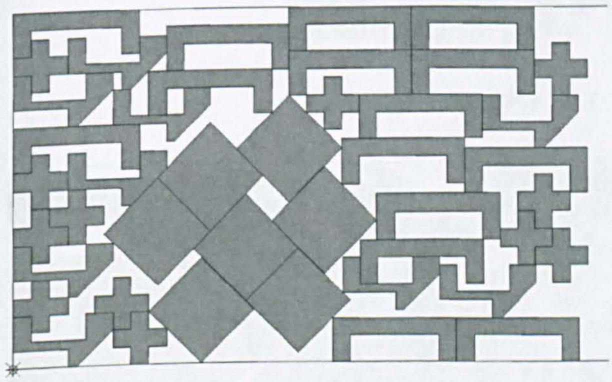
Poly3B: 40.74 units (417.7s)



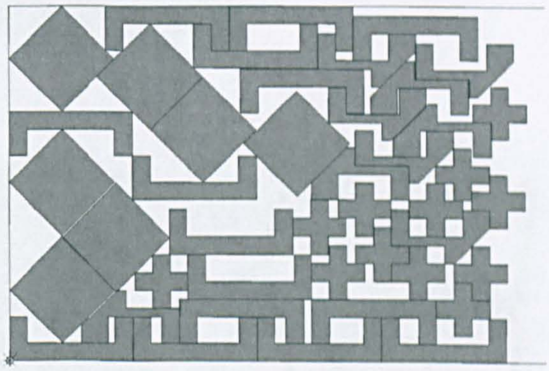
Poly4B: 51.73 units (95.7s)



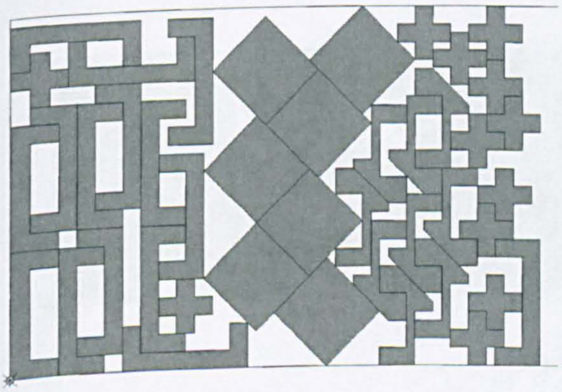
Poly5B: 60.54 units (676.6s)



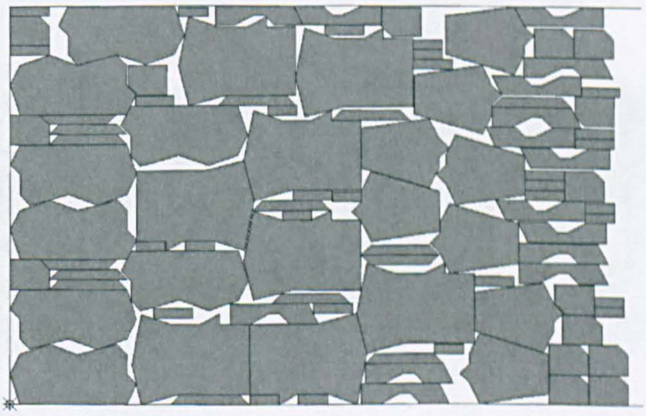
SHAPES0: 65.0 units (332.4s)



SHAPES1: 58.4 units (1810.1s)

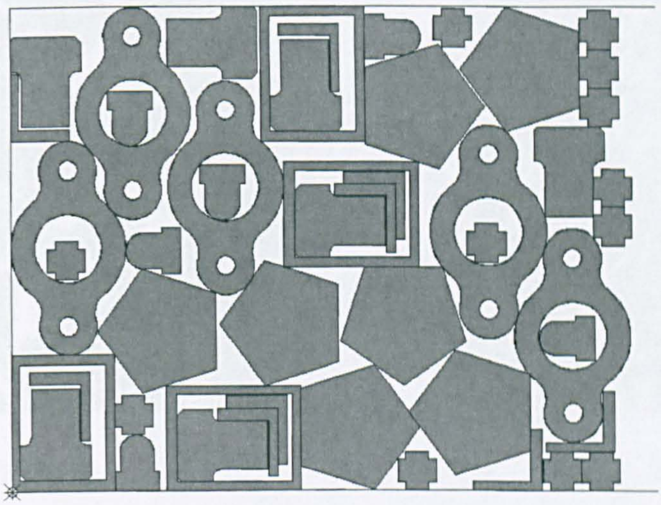


SHAPES: 59.0 units (31.4s)

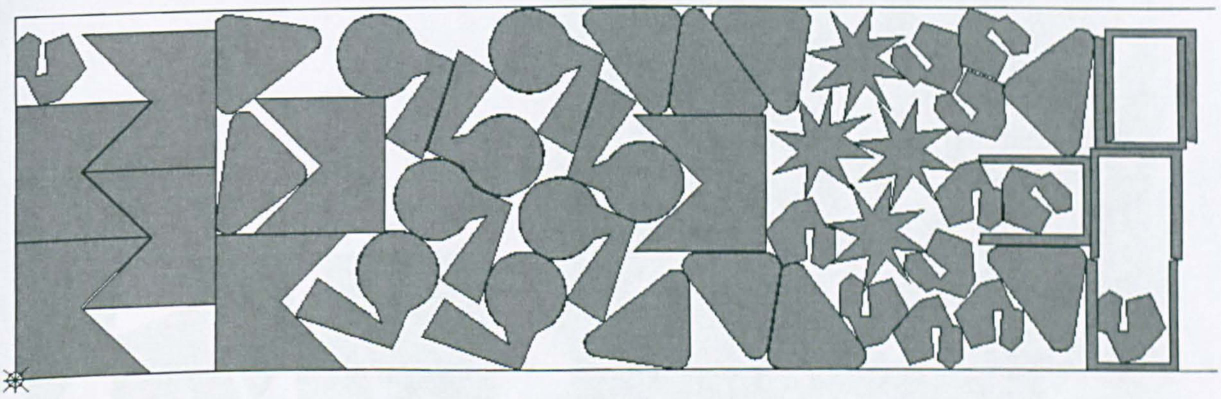


SHIRTS: 63.0 units (806.5s)

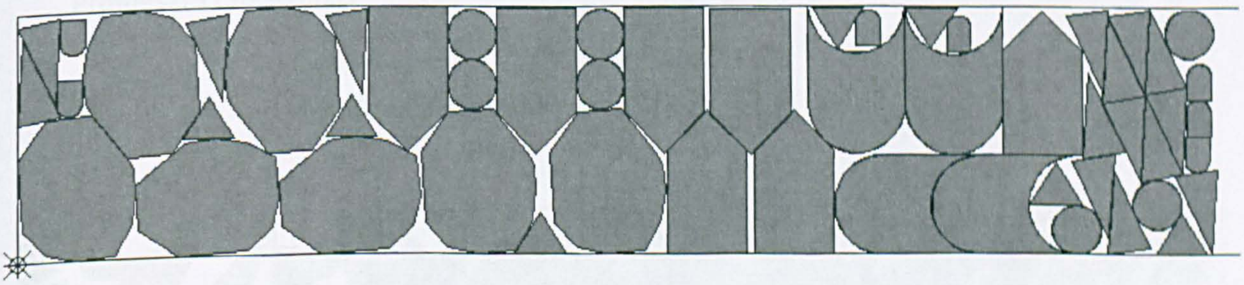
Profiles1: 1377.74 units – shown in the chapter 4 (see Figure 26)



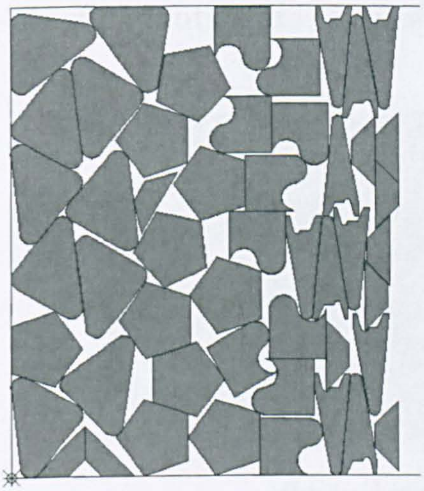
Profiles2: 3216.10 units



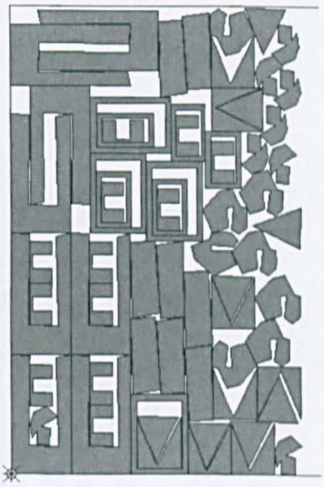
Profiles3: 8193.89 units



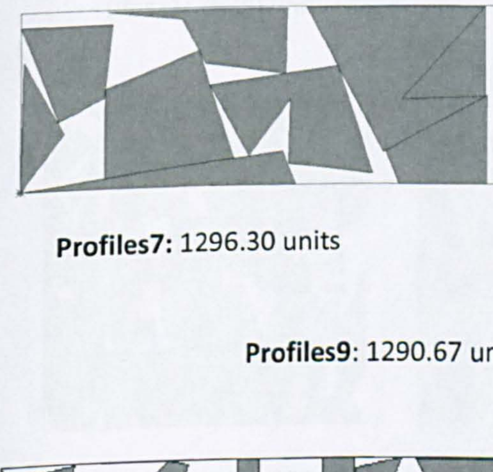
Profiles4: 2453.12 units



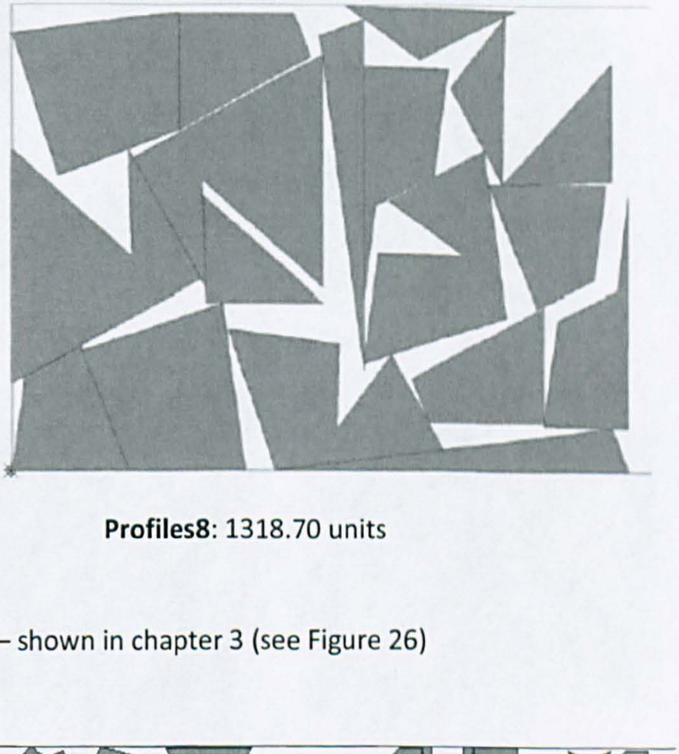
Profiles5: 3332.70 units



Profiles6: 3097.86 units

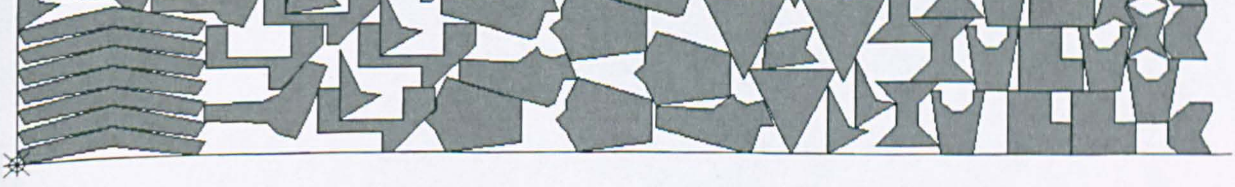


Profiles7: 1296.30 units



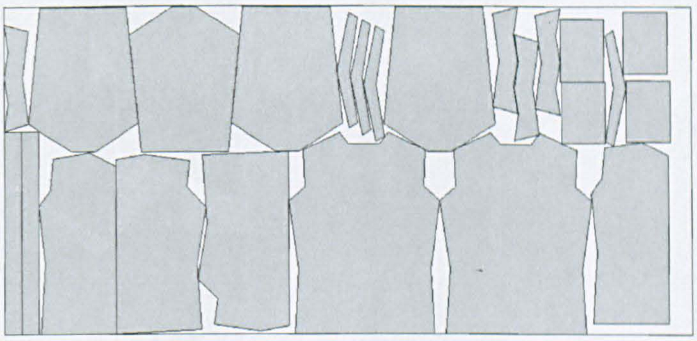
Profiles8: 1318.70 units

Profiles9: 1290.67 units – shown in chapter 3 (see Figure 26)

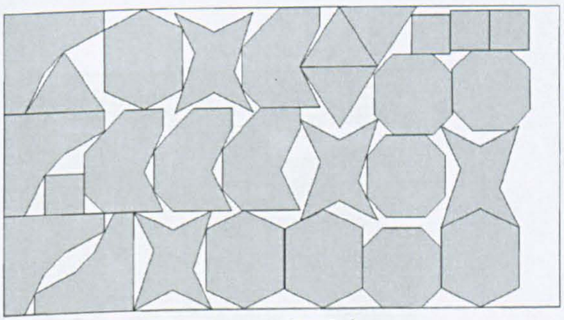


Profiles10: 11160.10 units

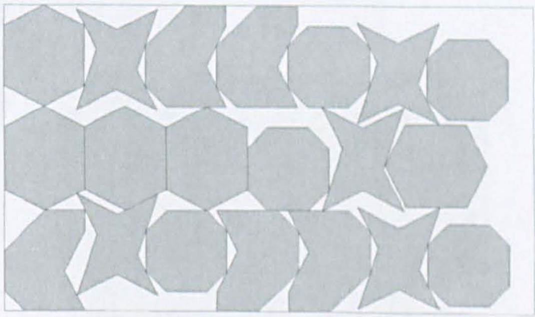
Appendix C – Best Layouts from Chapter 6



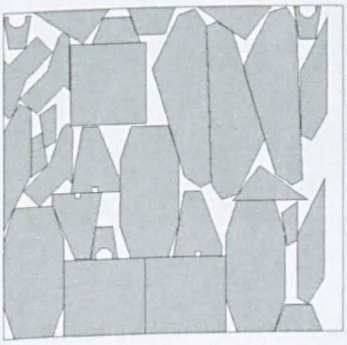
Albano: 9980.86 units (299s)



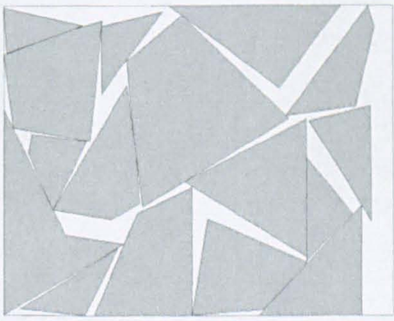
Blasz1: 26.80 units (281s)



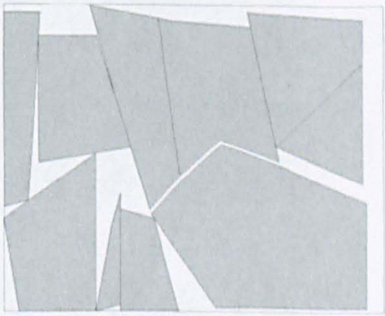
Blasz2: 24.80 units (14s)



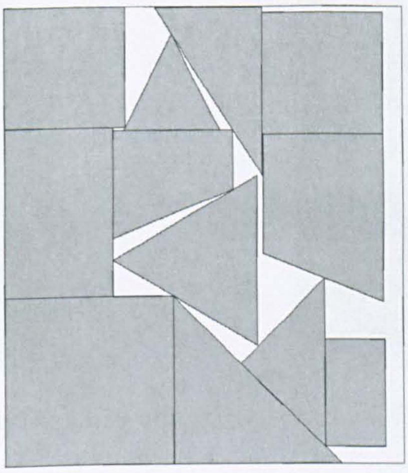
Dagli: 59.94 units (252s)



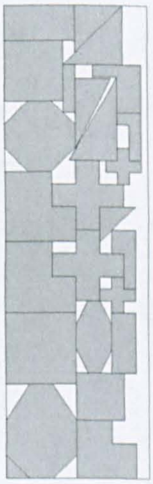
Dighe1: 1210.00 units (3s)



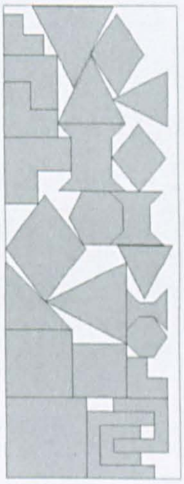
Dighe2: 1180.00 units (148s)



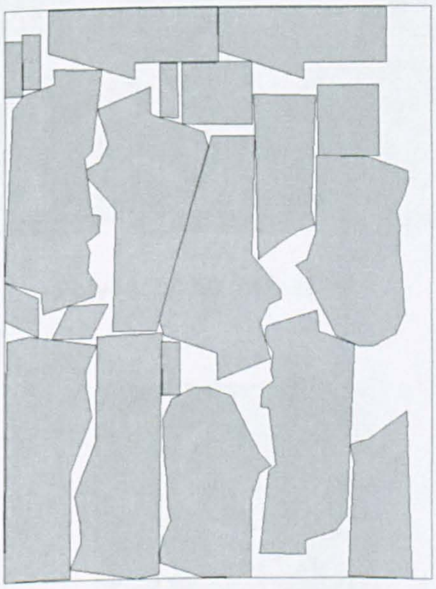
Fu: 31.57 units (139s)



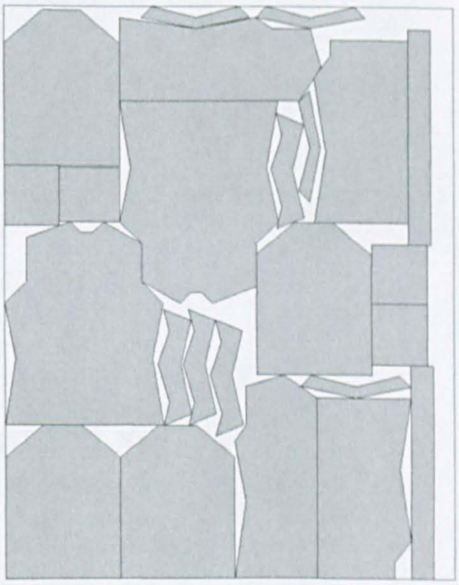
Jakobs1: 11.50 units (29s)



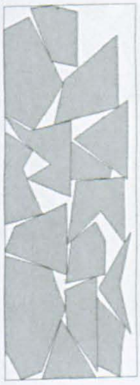
Jakobs2: 24.70 units (51s)



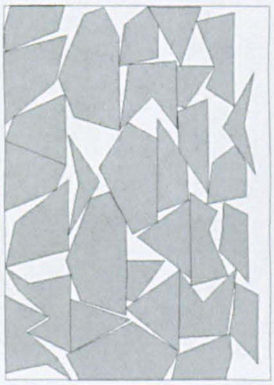
Mao: 1821.70 units (152s)



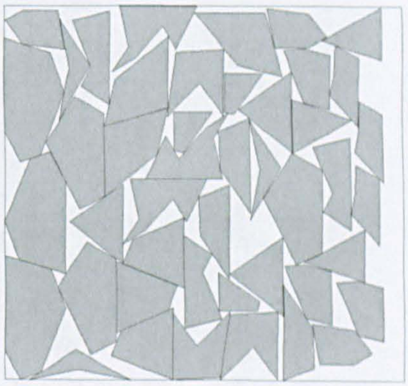
Marques: 78.00 units (21s)



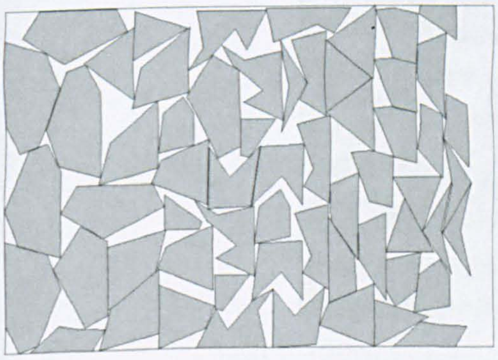
Poly1A: 13.30 units (254s)



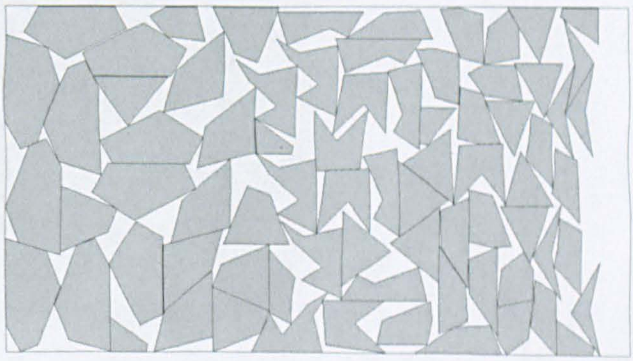
Poly2A: 27.09 units (239s)



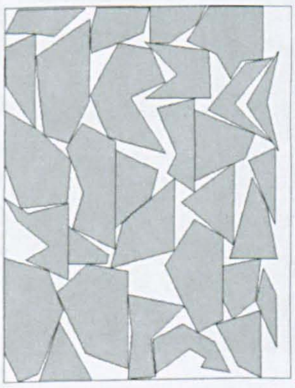
Poly3A: 40.45 units (429s)



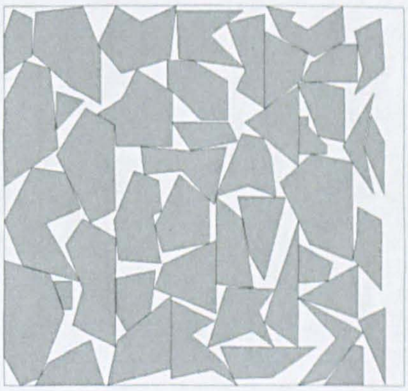
Poly4A: 54.60 units (224s)



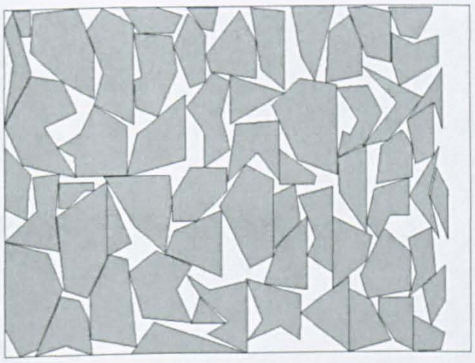
Poly5A: 68.84 units (300s)



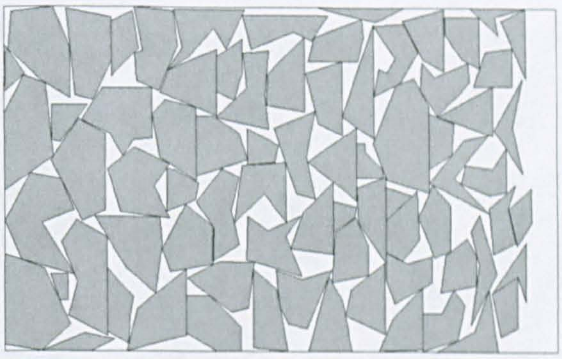
Poly2B: 29.63 units (189s)



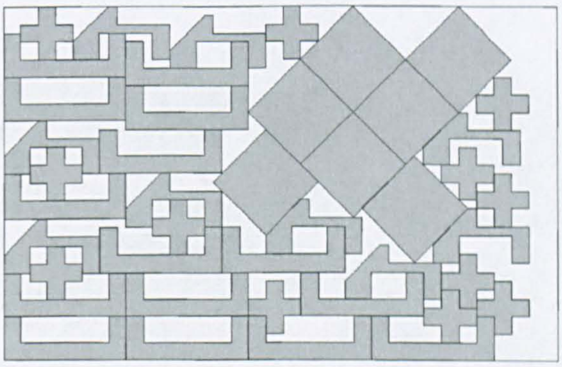
Poly3B: 40.50 units (114s)



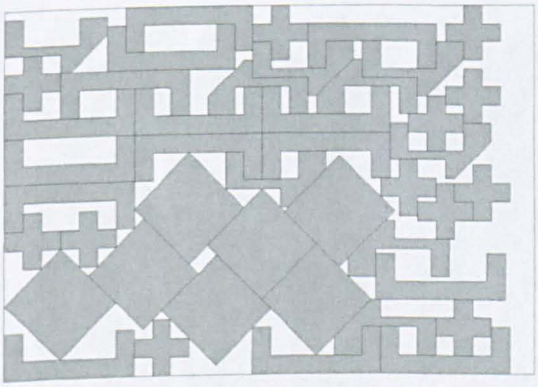
Poly4B: 51.18 units (176s)



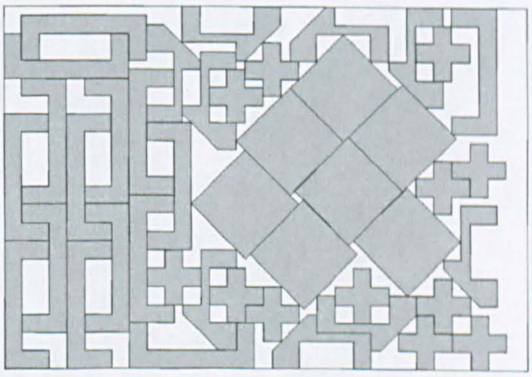
Poly5B: 60.86 units (299s)



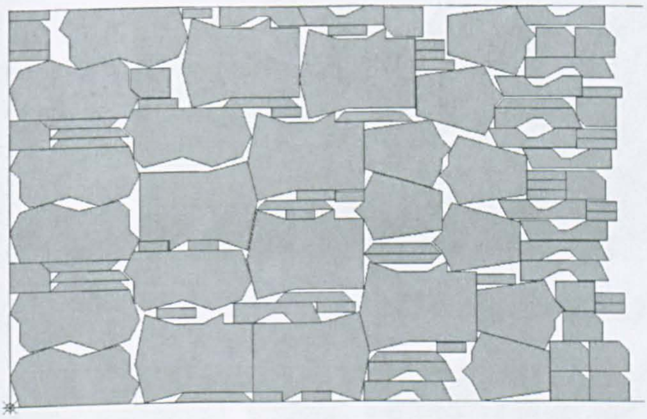
SHAPES0: 60.00 units (274s)

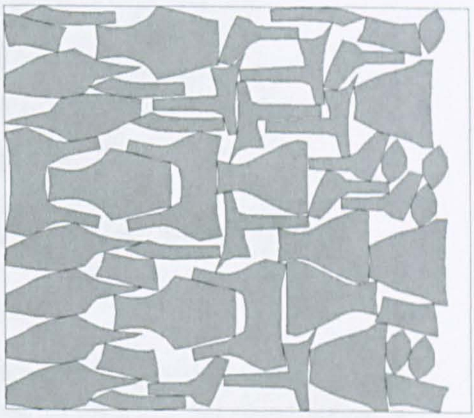


SHAPES1: 55.00 units (166s)

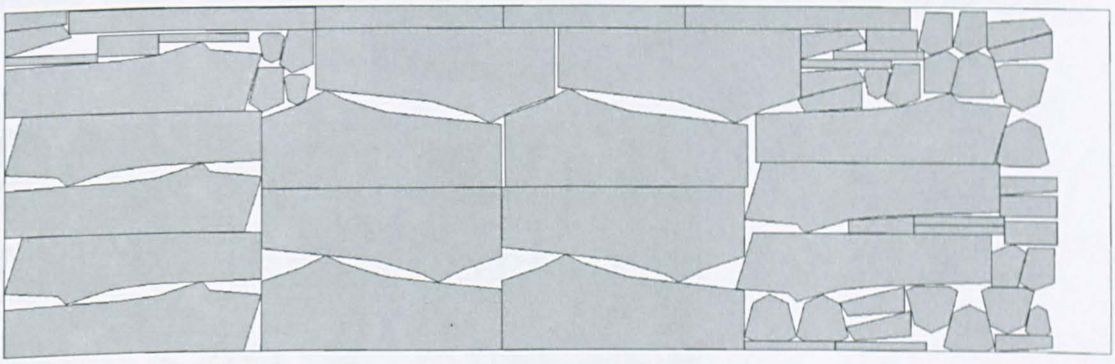


SHAPES: 56.00 units (226s)

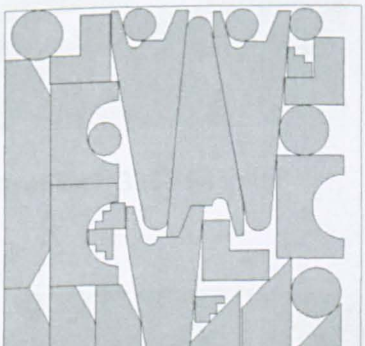


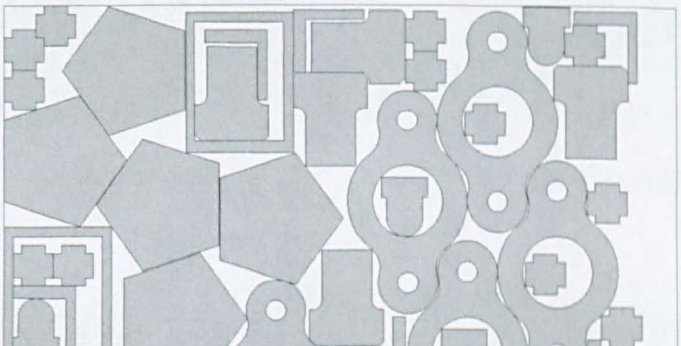


SHIRTS: 63.16 units (806s) **SWIM**: 6270.88 units (141s)

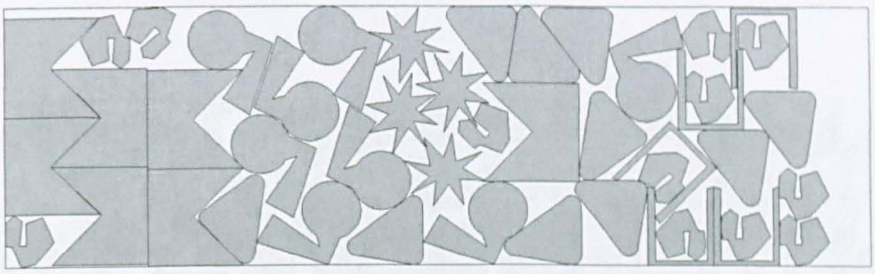


TROUSERS: 243.00 units (12484s)

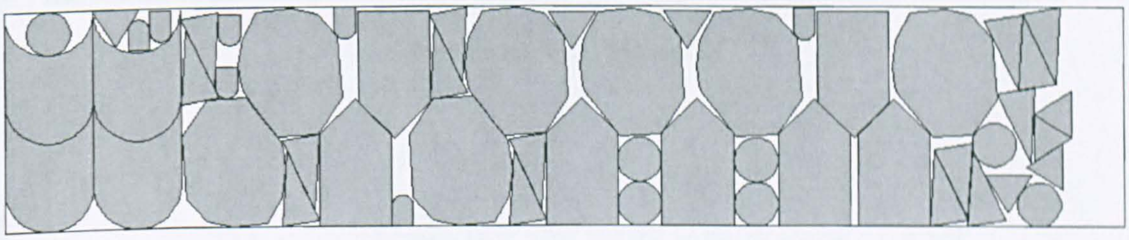




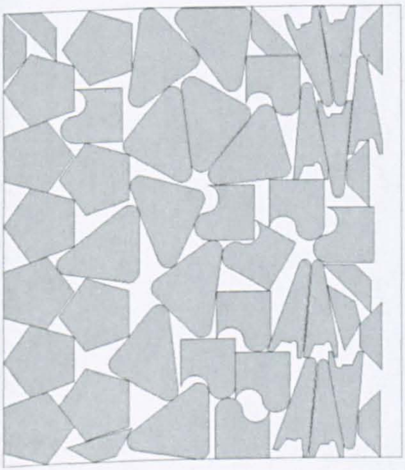
Profiles1: 1359.90 units **Profiles2**: 3194.19 units



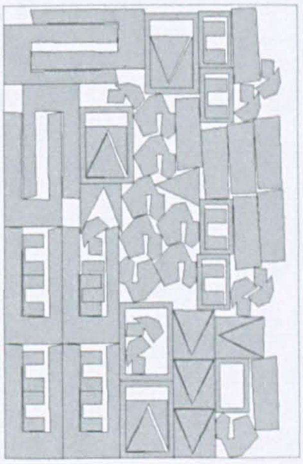
Profiles3: 7881.13 units



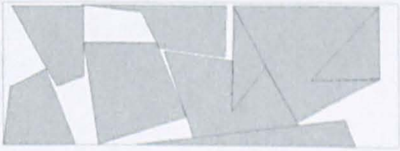
Profiles4: 2425.26 units



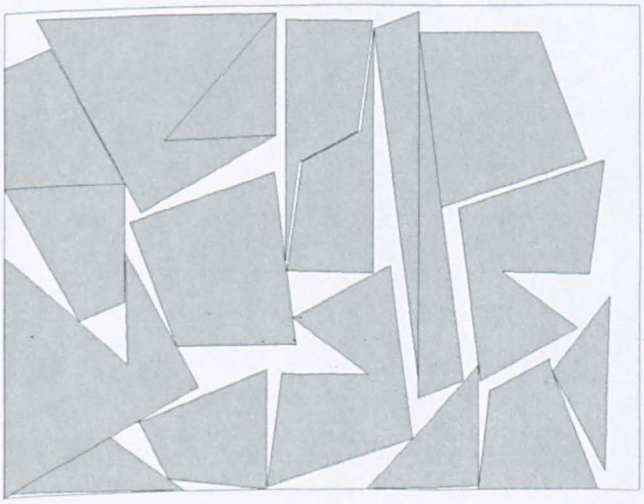
Profiles5: 3351.94 units



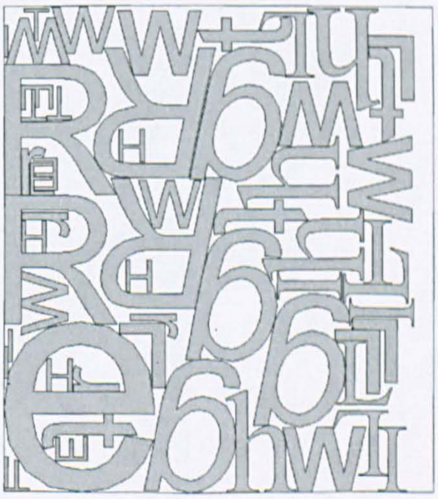
Profiles6: 3121.36 units



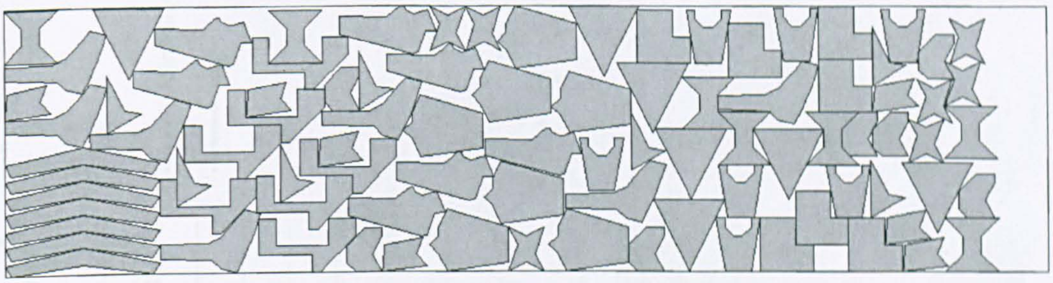
Profiles7: 1292.30 units



Profiles8: 1263.11 units



Profiles9: 1278.21 units

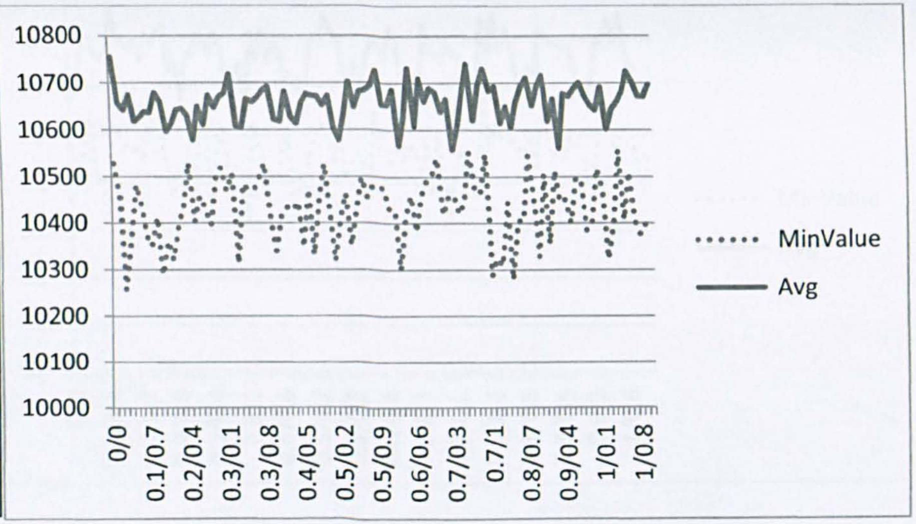


Profiles10: 11219.60 units

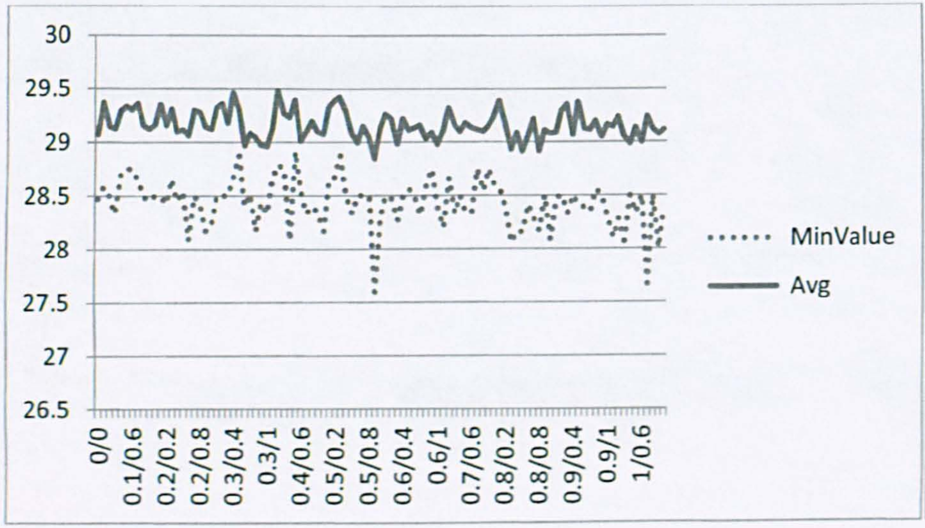
Appendix D – Result Graphs from Chapter 7

See Chapter 7 (Figure 81)

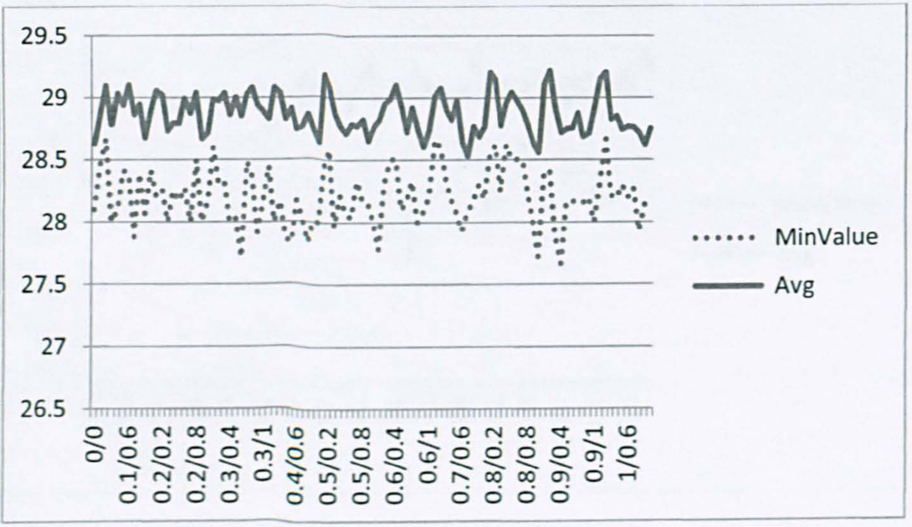
Albano (A)



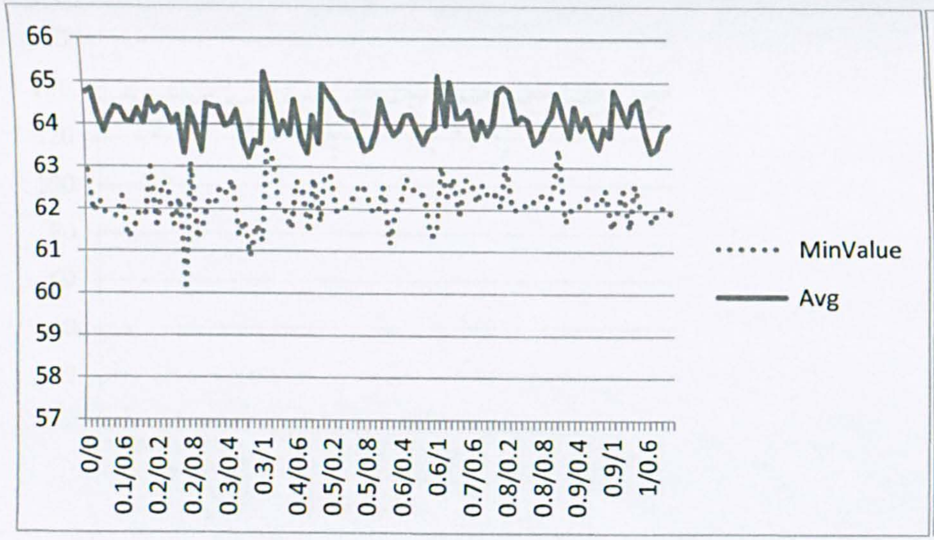
Albano (L)



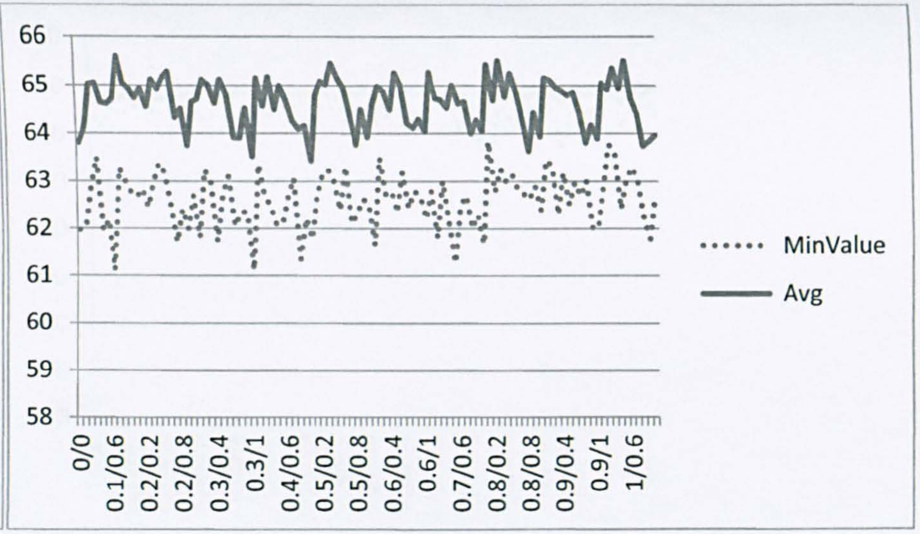
Blaz (A)



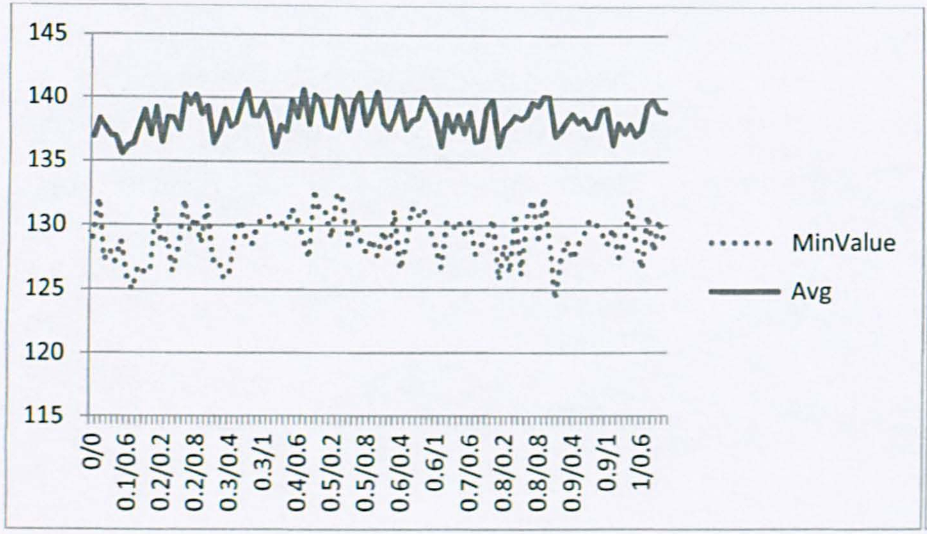
Blaz (L)



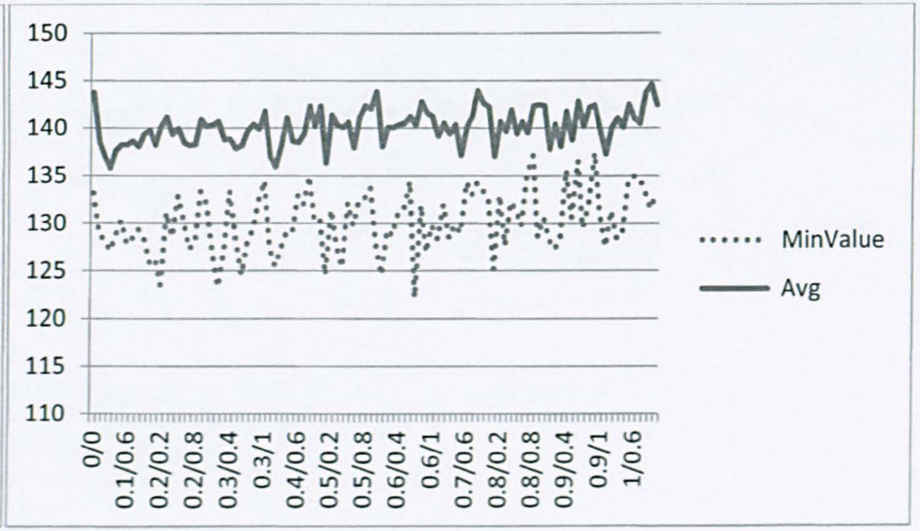
Dagli (A)



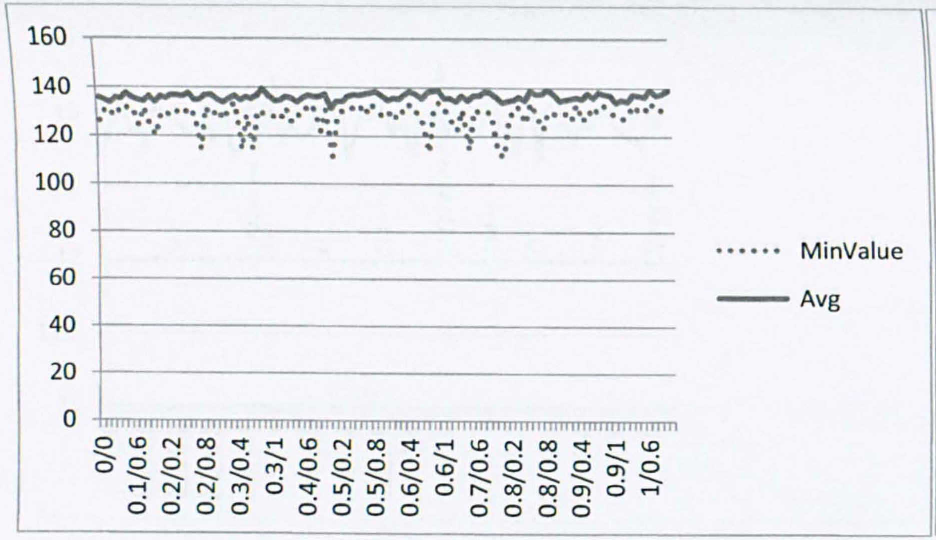
Dagli (L)



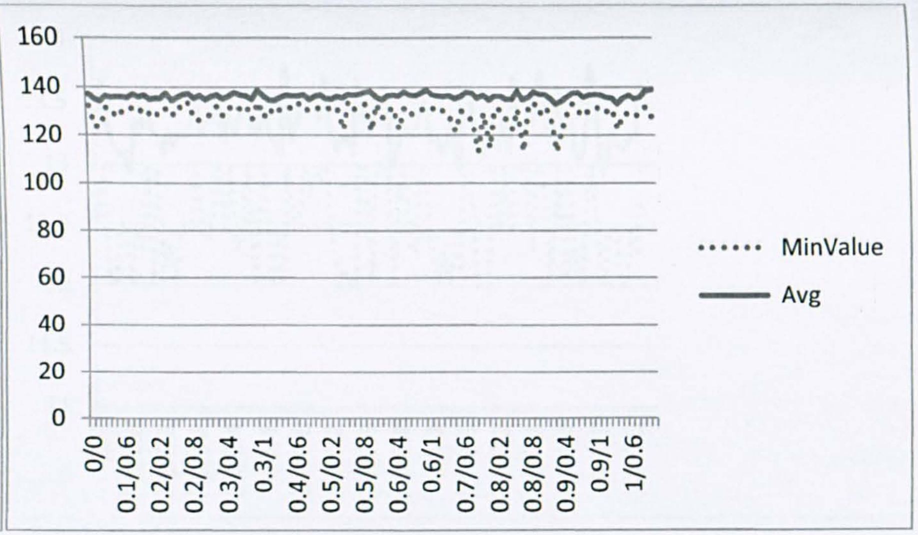
Dighe1 (A)



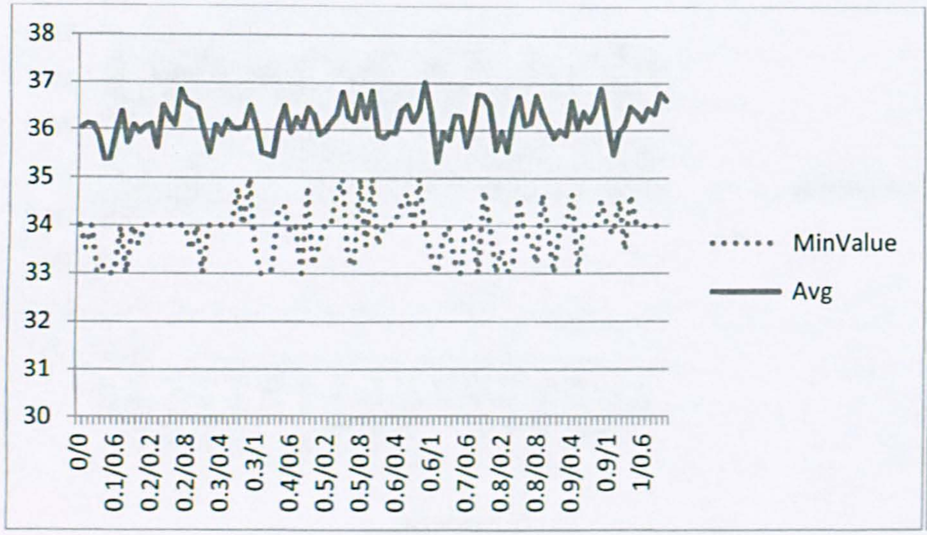
Dighe1 (L)



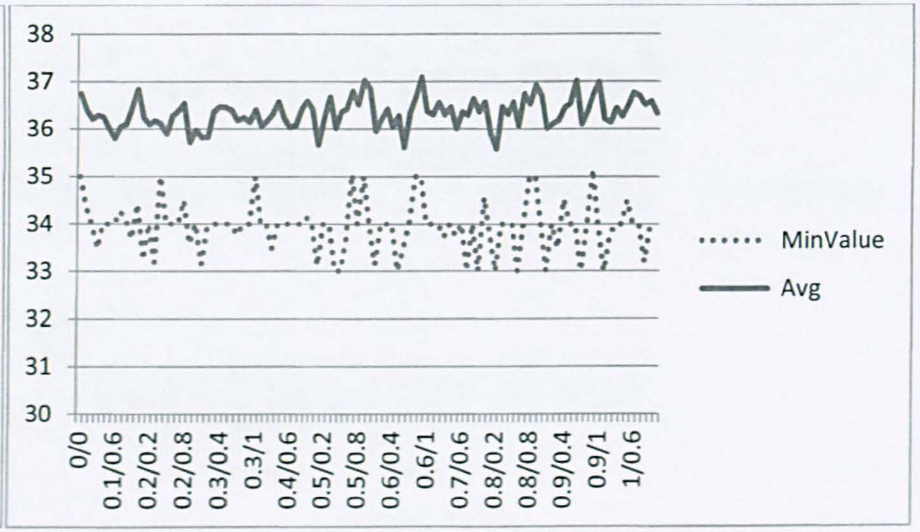
Dighe2 (A)



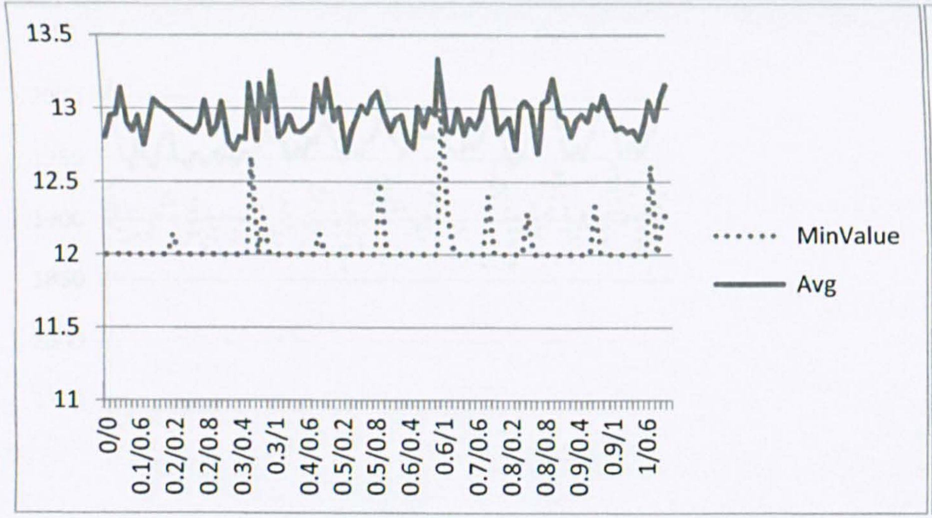
Dighe2 (L)



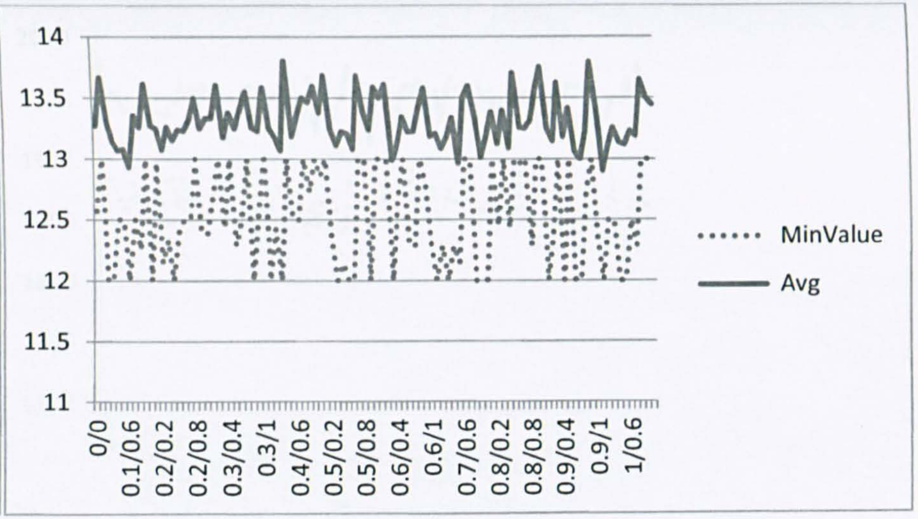
Fu (A)



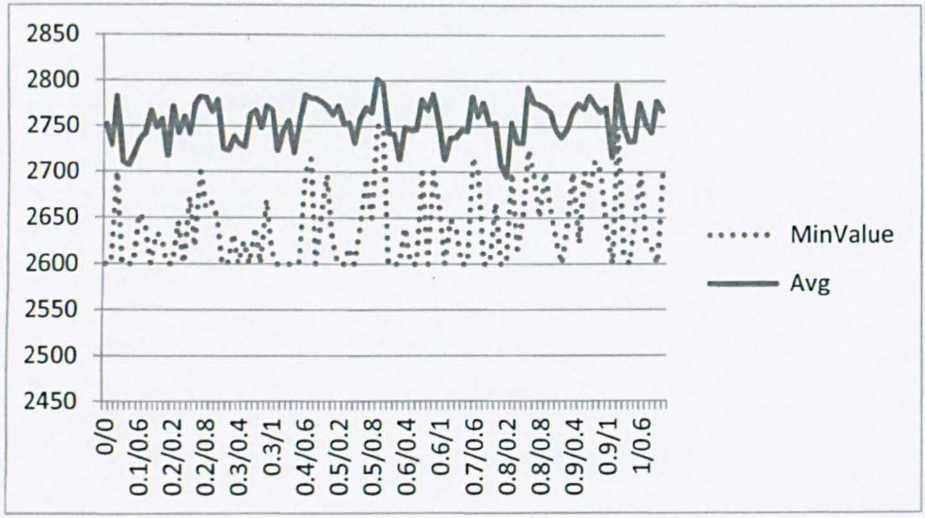
Fu (L)



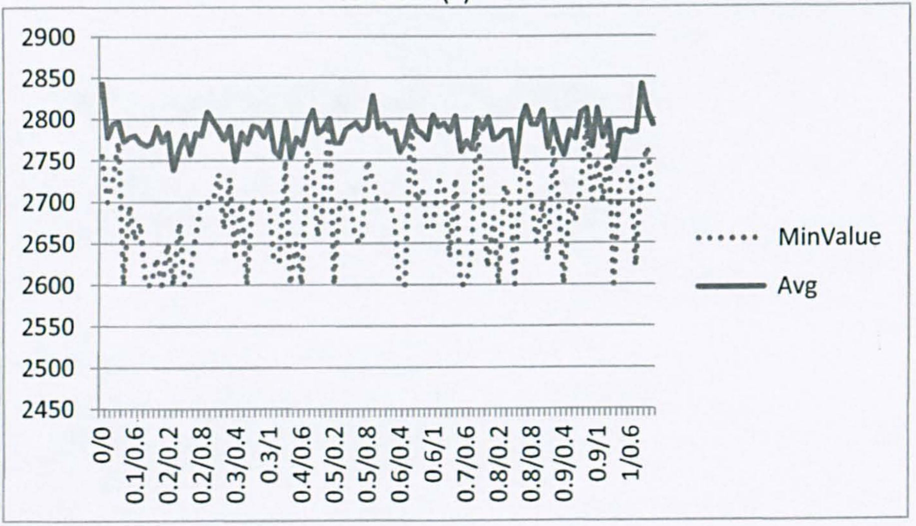
Jakobs1 (A)



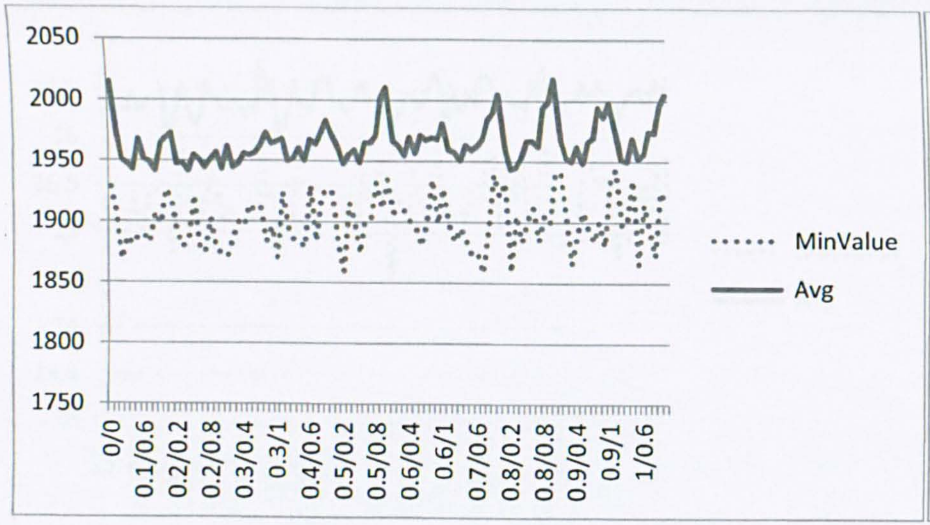
Jakobs1 (L)



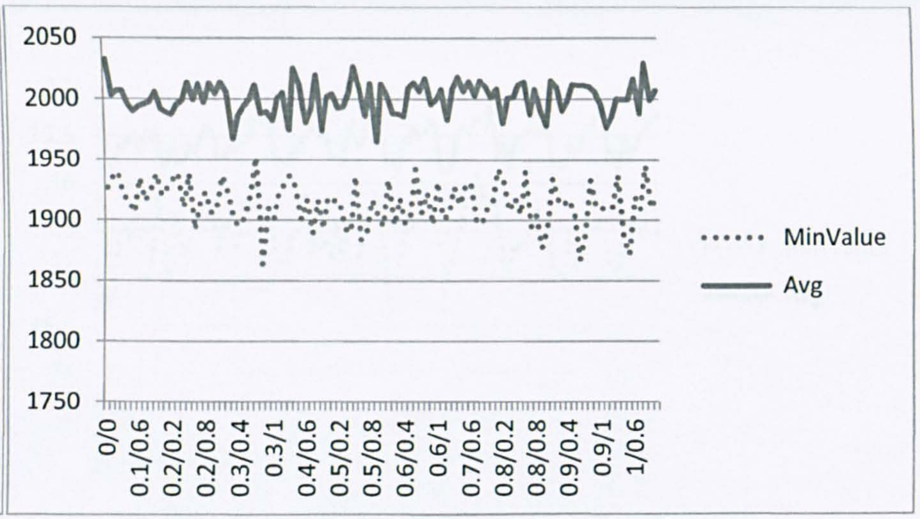
Jakobs2 (A)



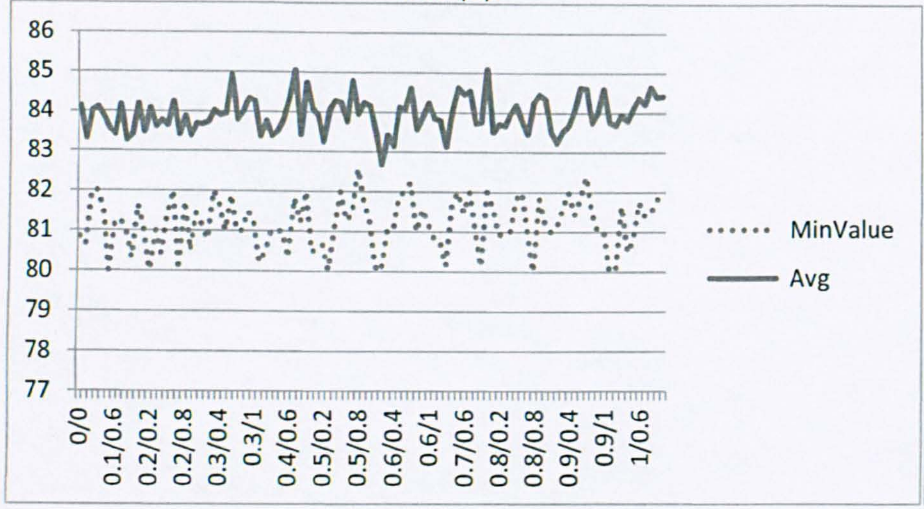
Jakobs2 (L)



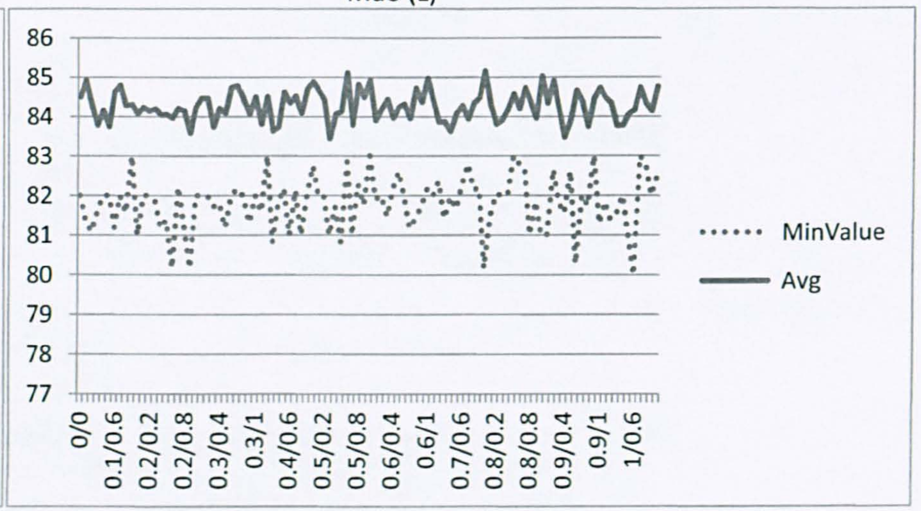
Mao (A)



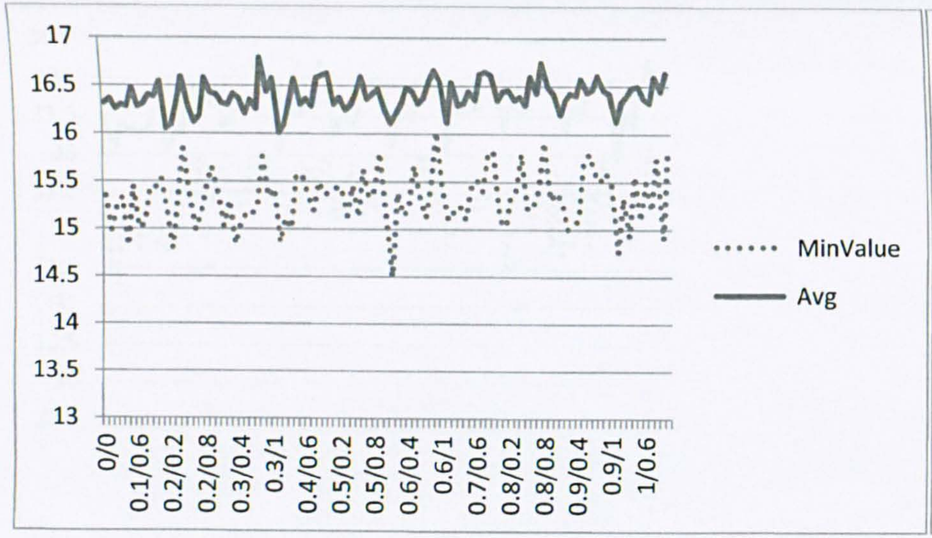
Mao (L)



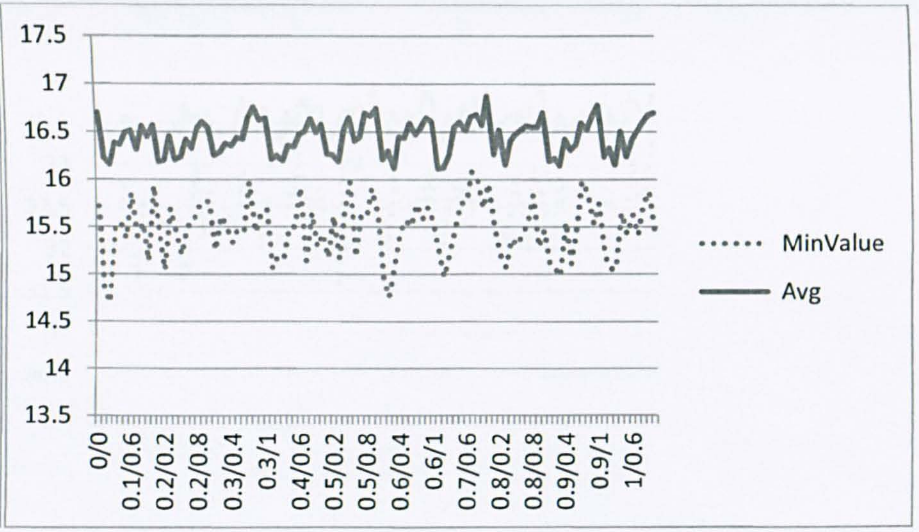
Marques (A)



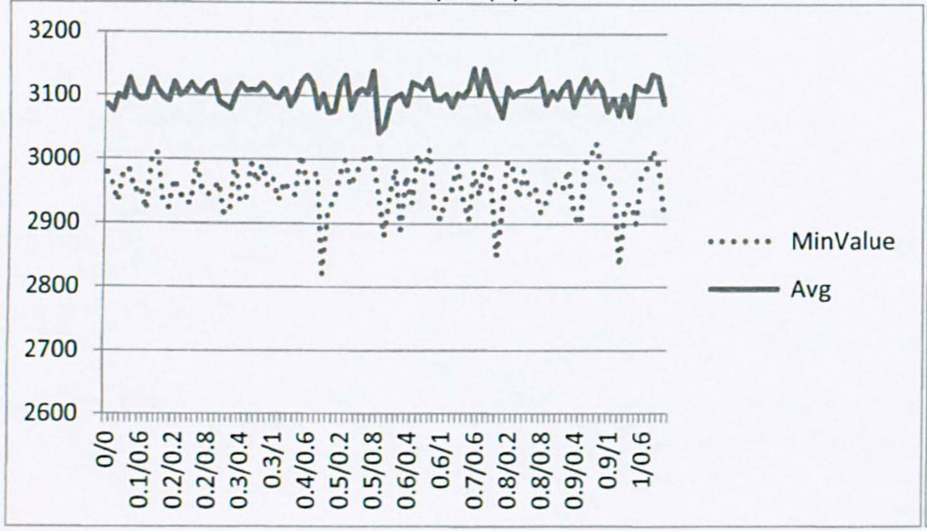
Marques (L)



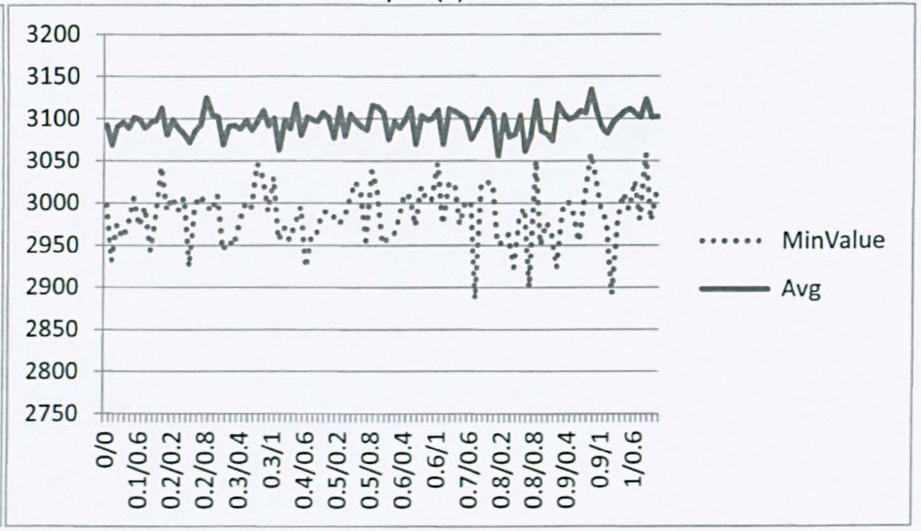
Poly1A (A)



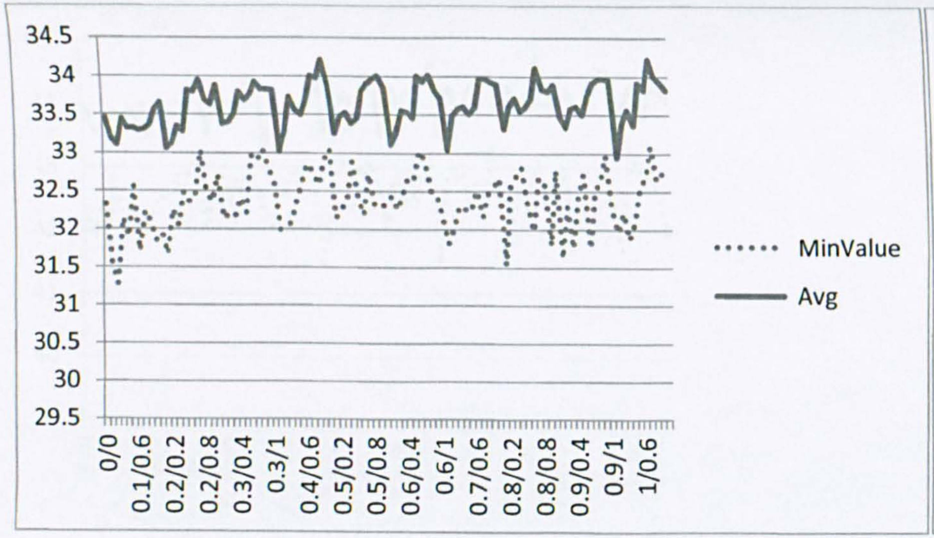
Poly1A (L)



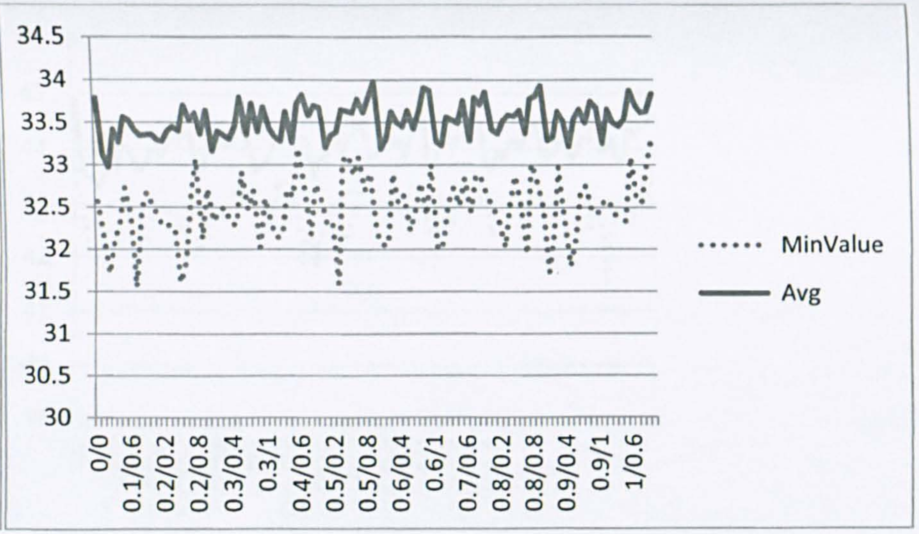
Poly2A (A)



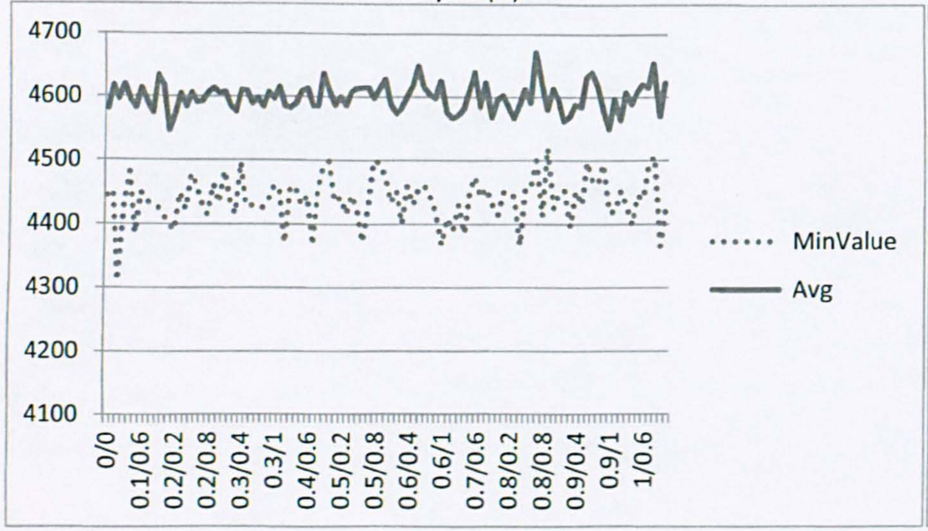
Poly2A (L)



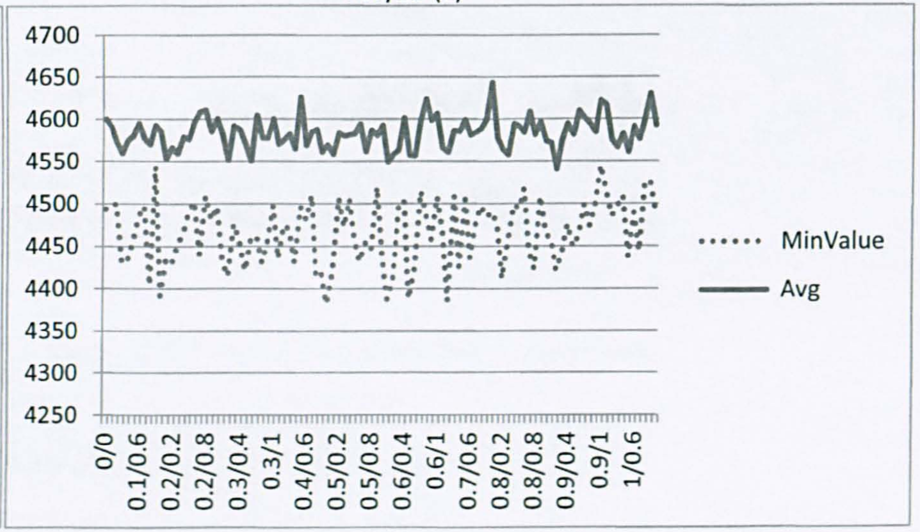
Poly2B (A)



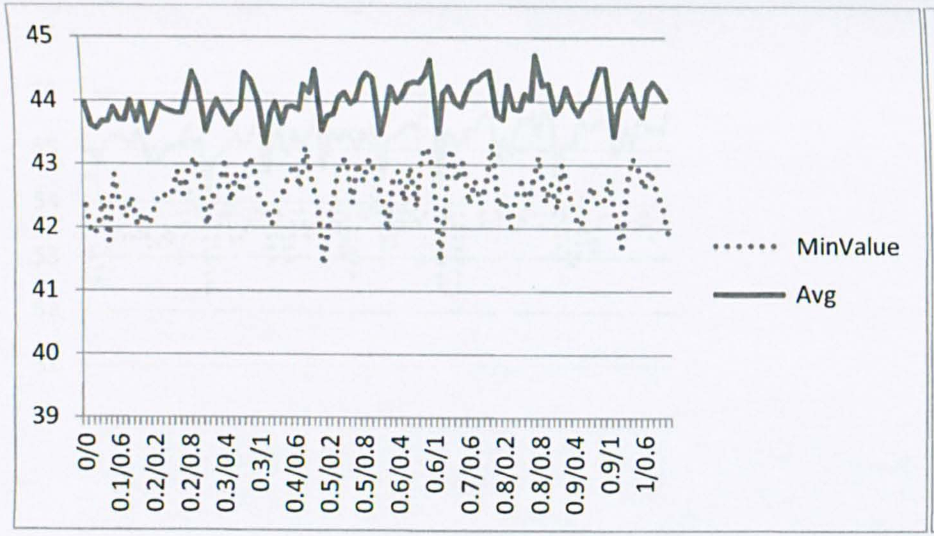
Poly2B (L)



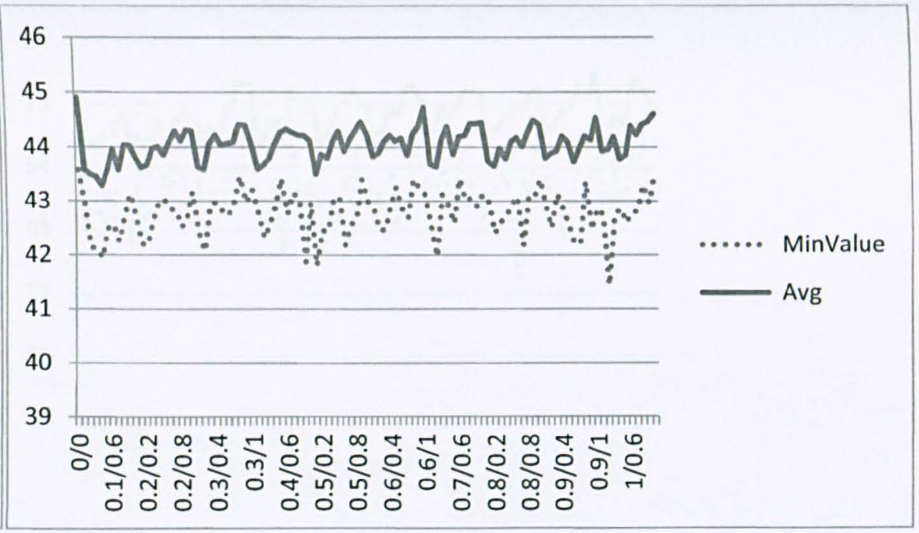
Poly3A (A)



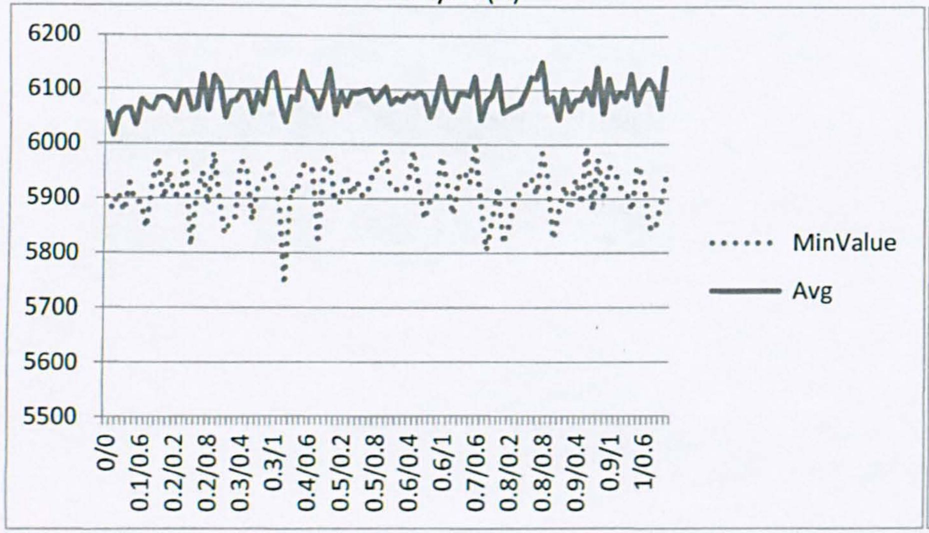
Poly3A (L)



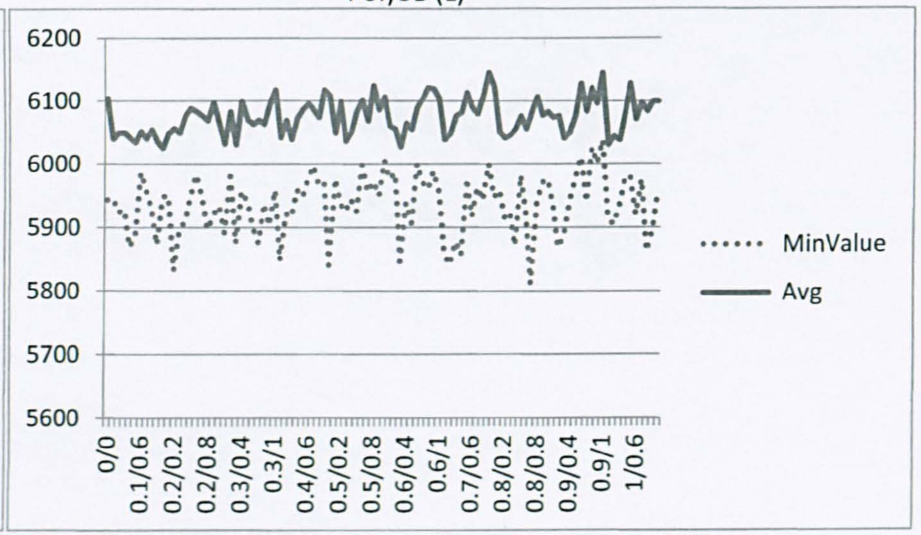
Poly3B (A)



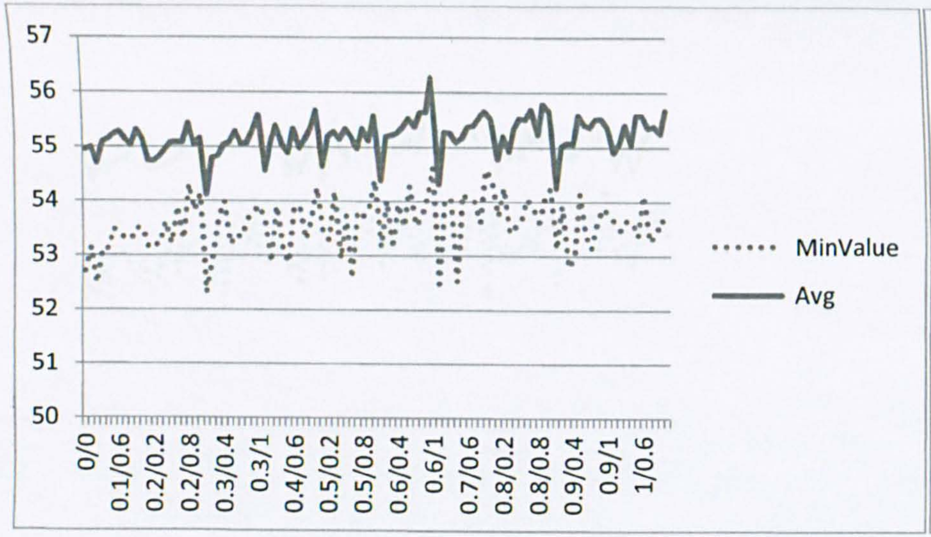
Poly3B (L)



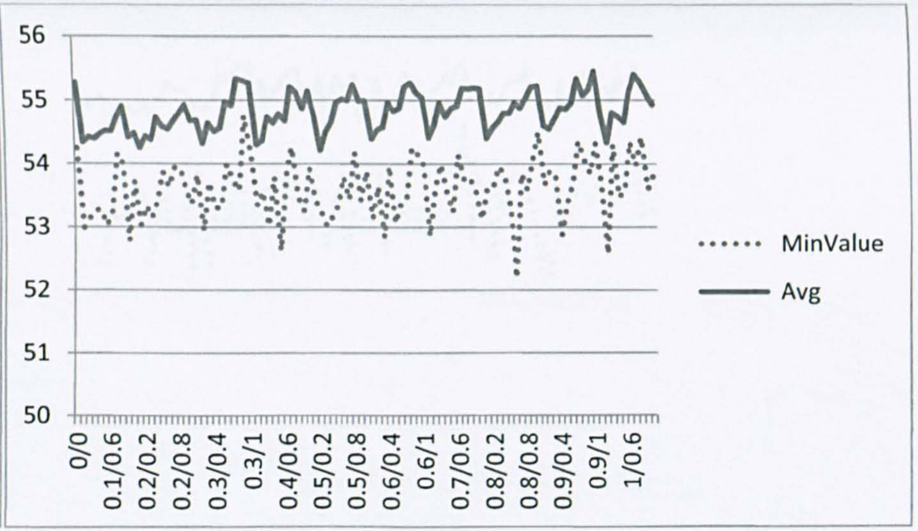
Poly4A (A)



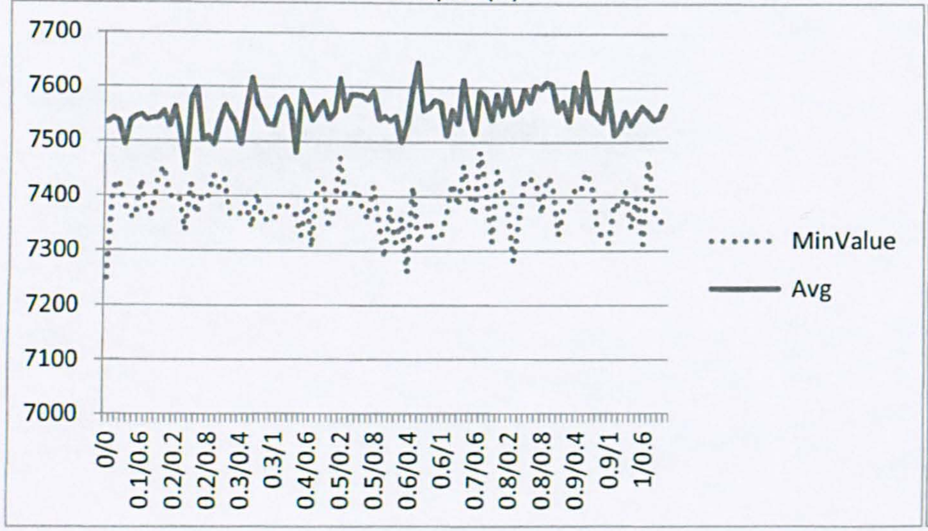
Poly4B (L)



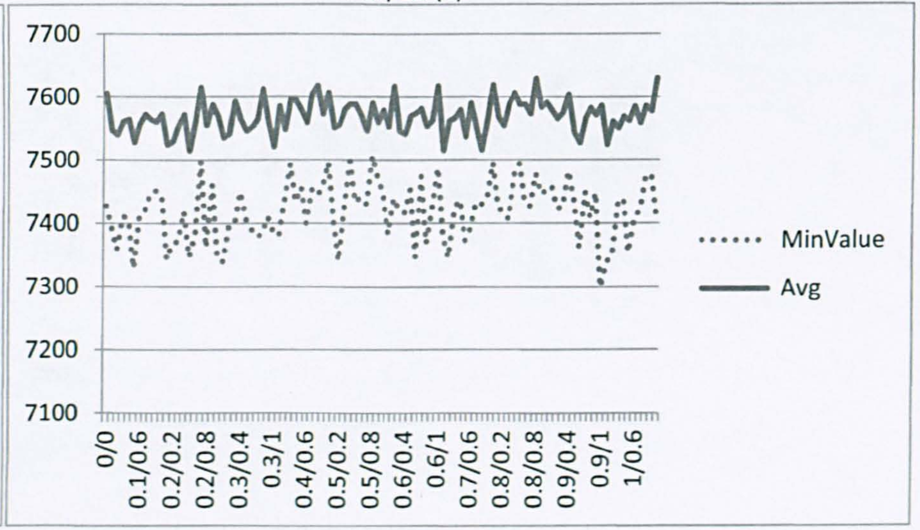
Poly4B (A)



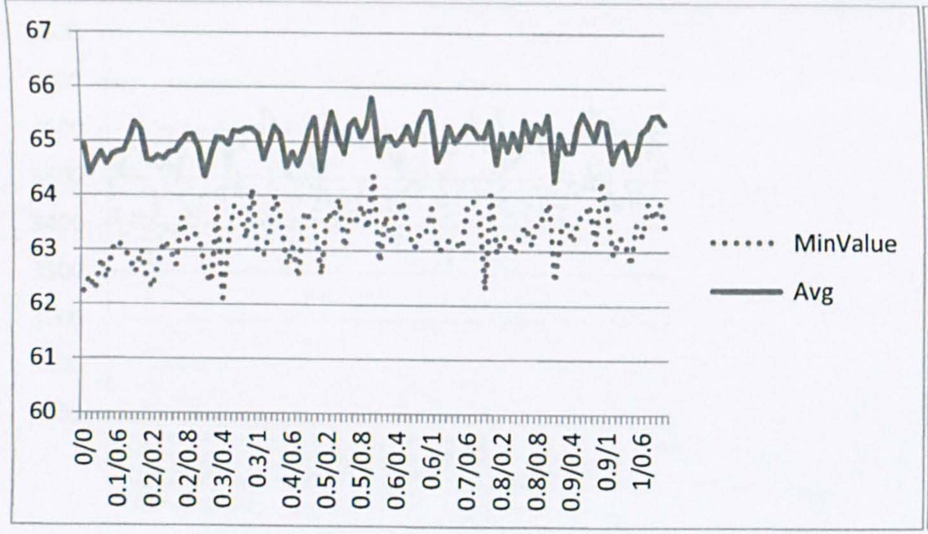
Poly4B (L)



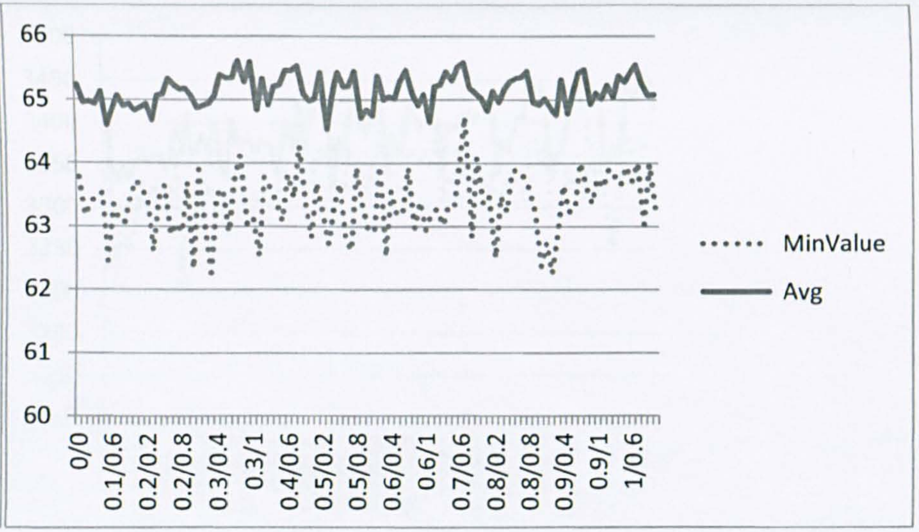
Poly5A (A)



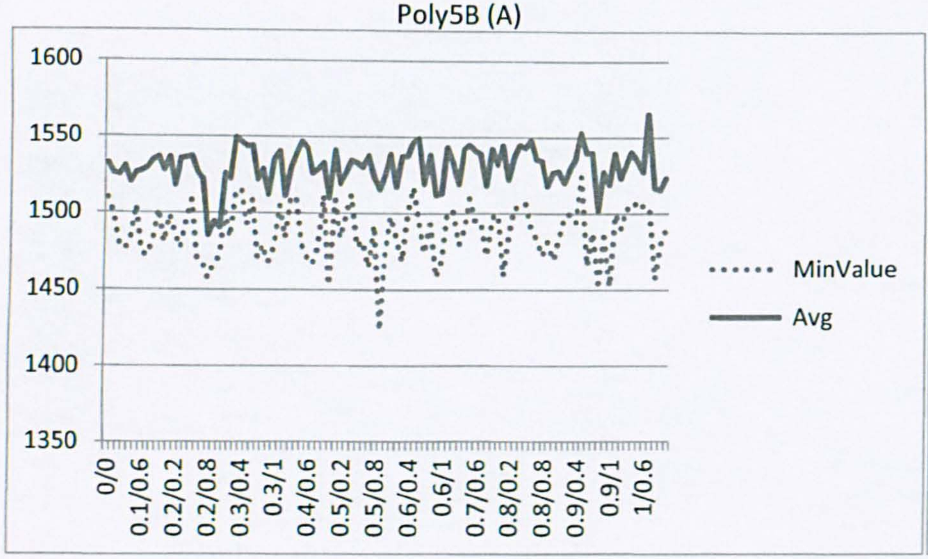
Poly5A (L)



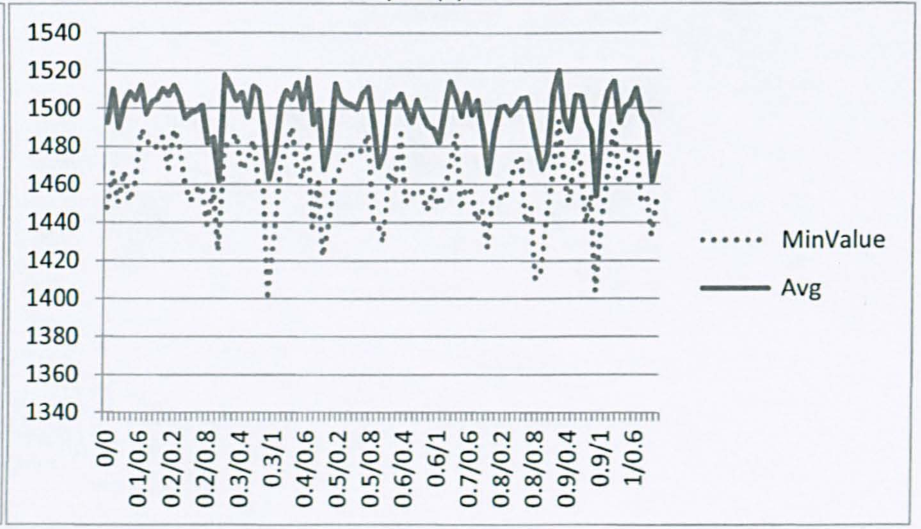
Poly5B (A)



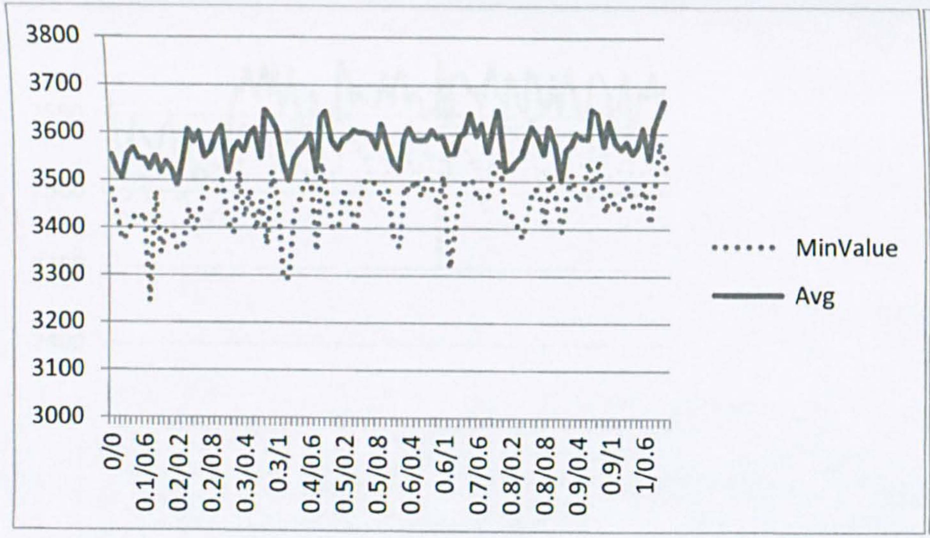
Poly5B (L)



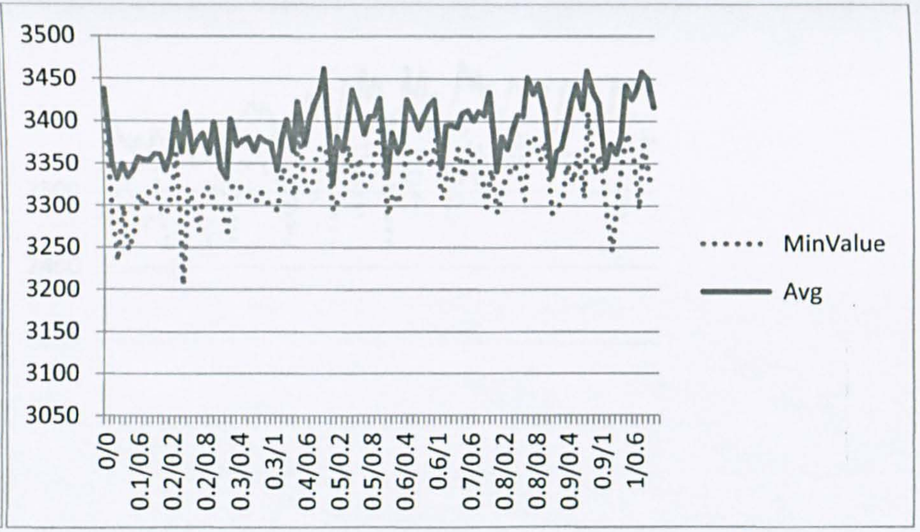
Profiles1 (A)



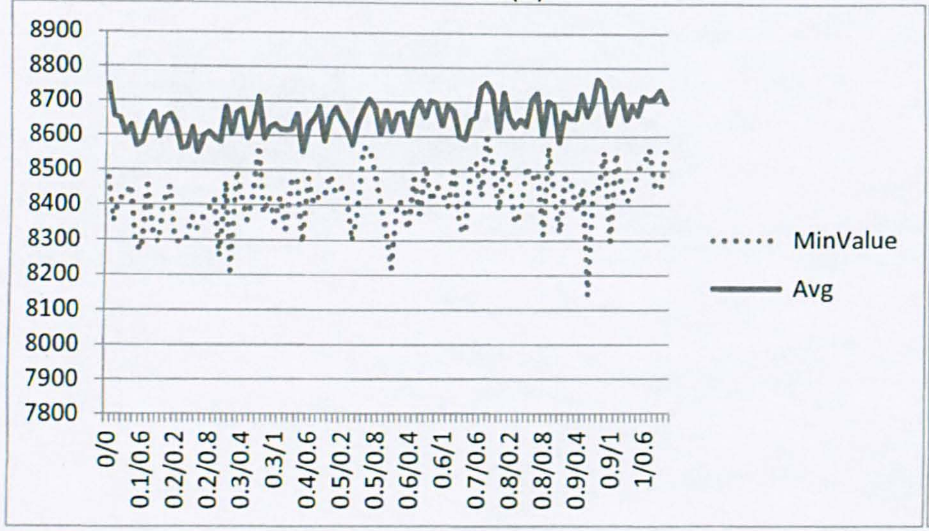
Profiles1 (L)



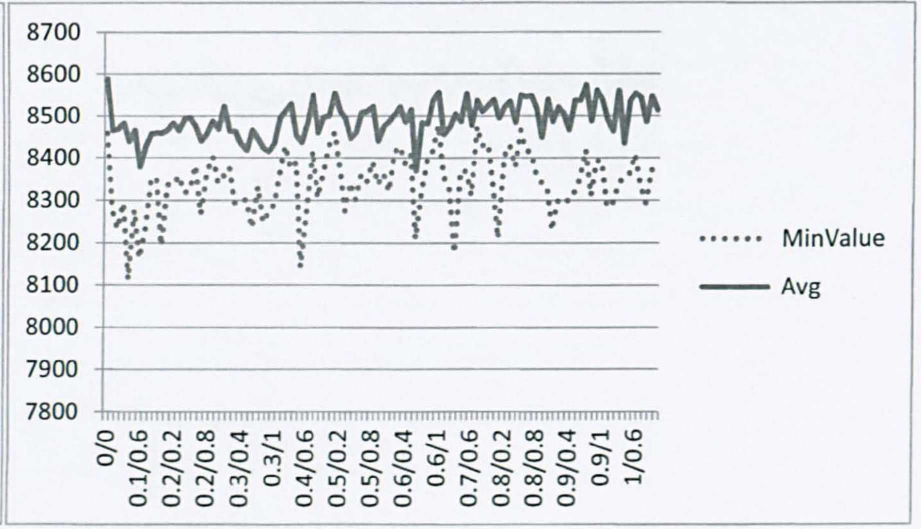
Profiles2 (A)



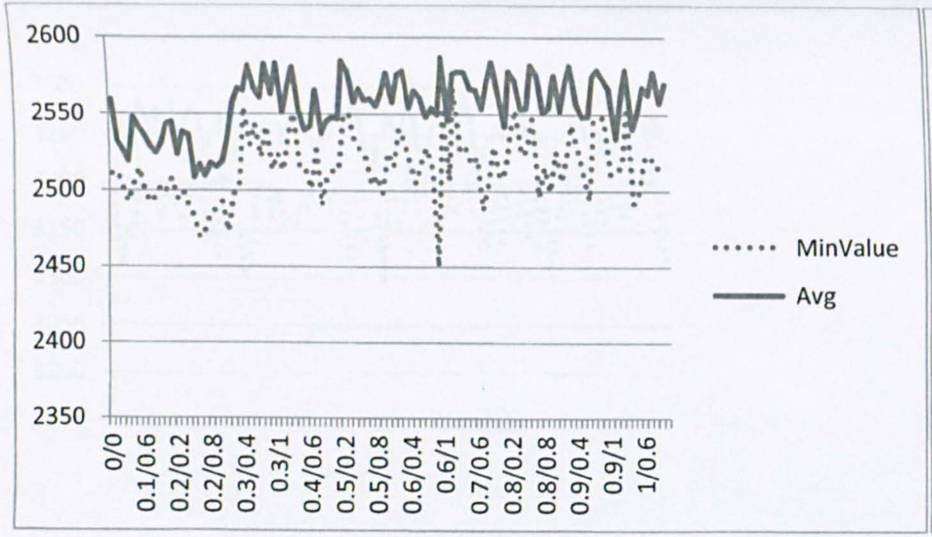
Profiles2 (L)



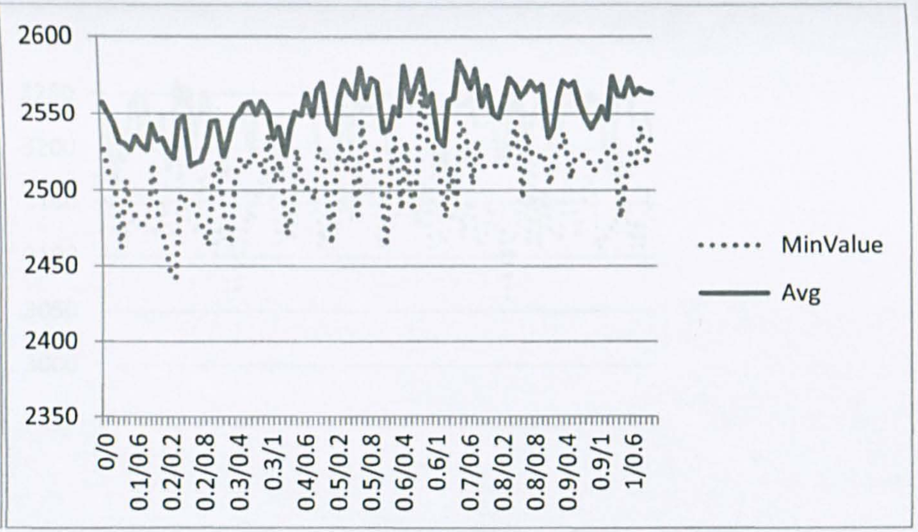
Profiles3 (A)



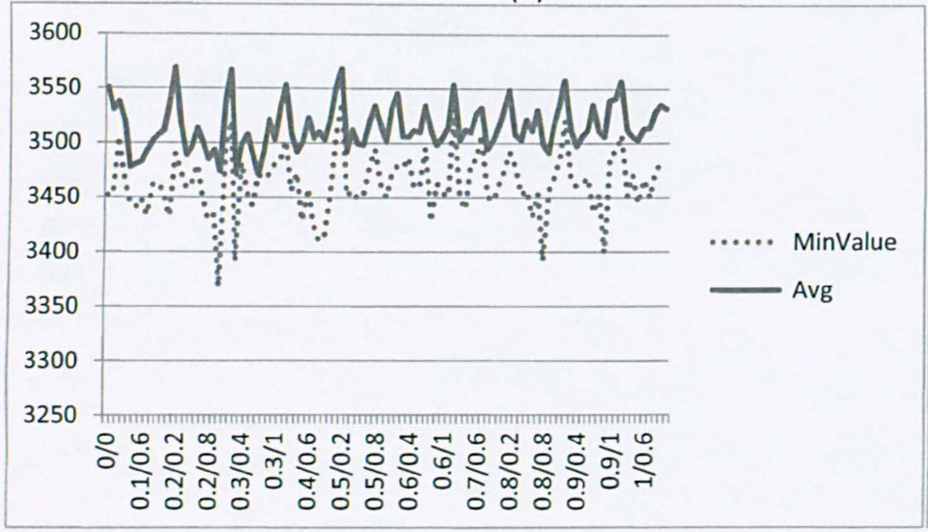
Profiles3 (L)



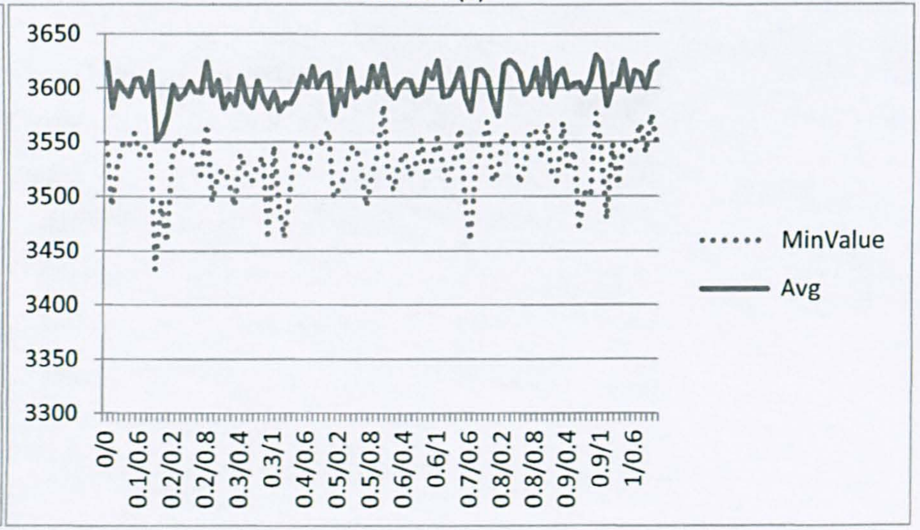
Profiles4 (A)



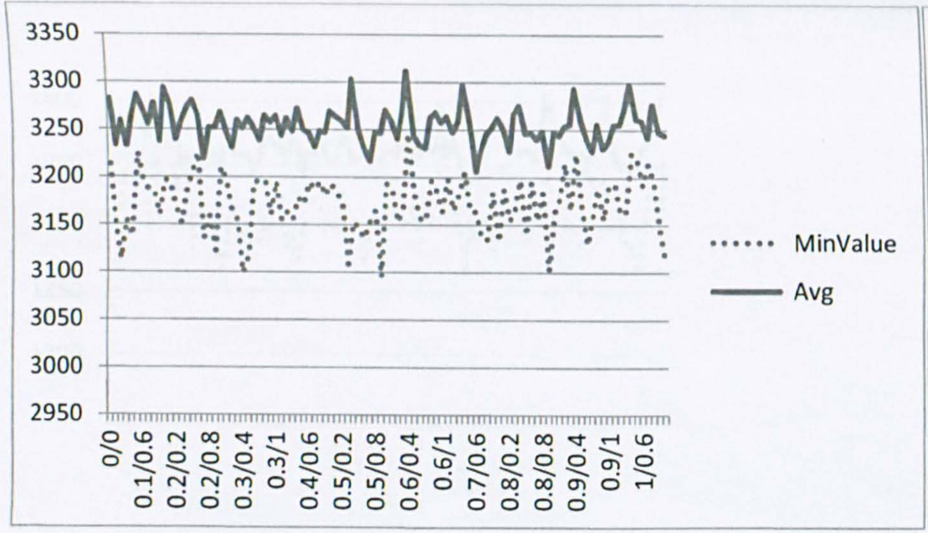
Profiles4 (L)



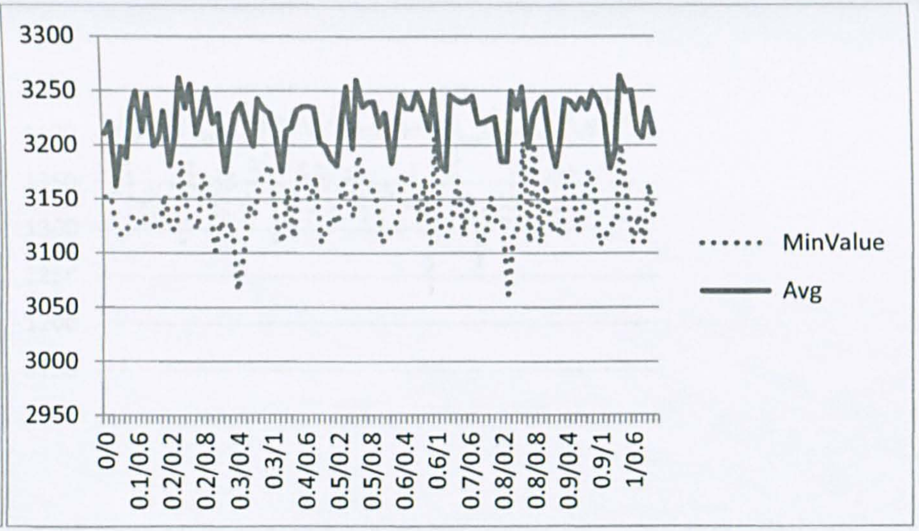
Profiles5 (A)



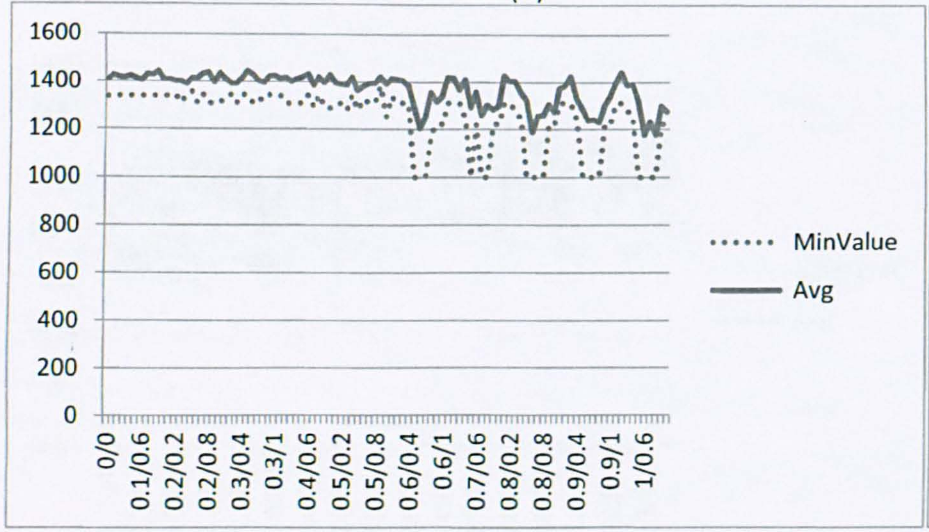
Profiles5 (L)



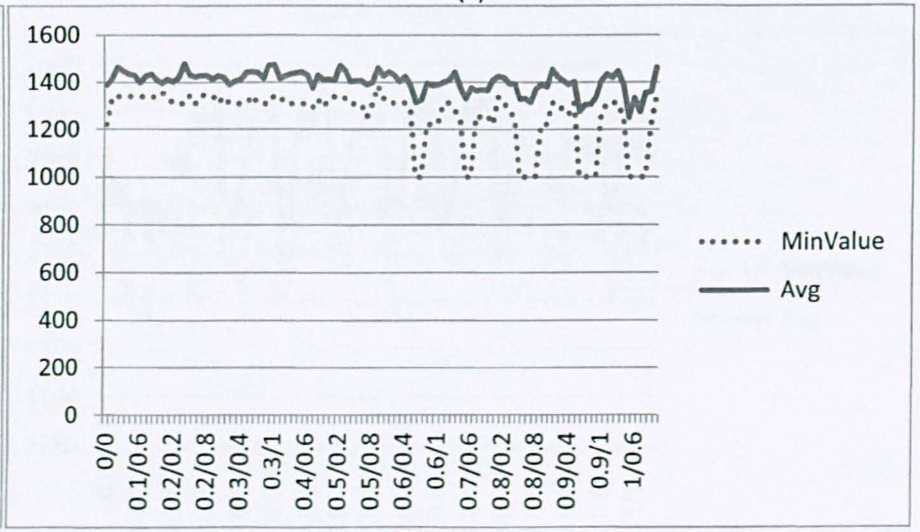
Profiles6 (A)



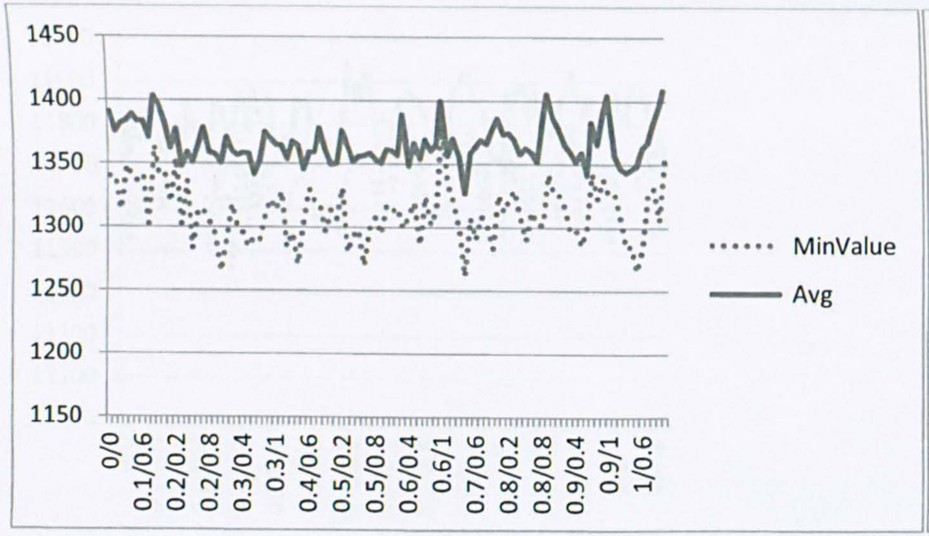
Profiles6 (L)



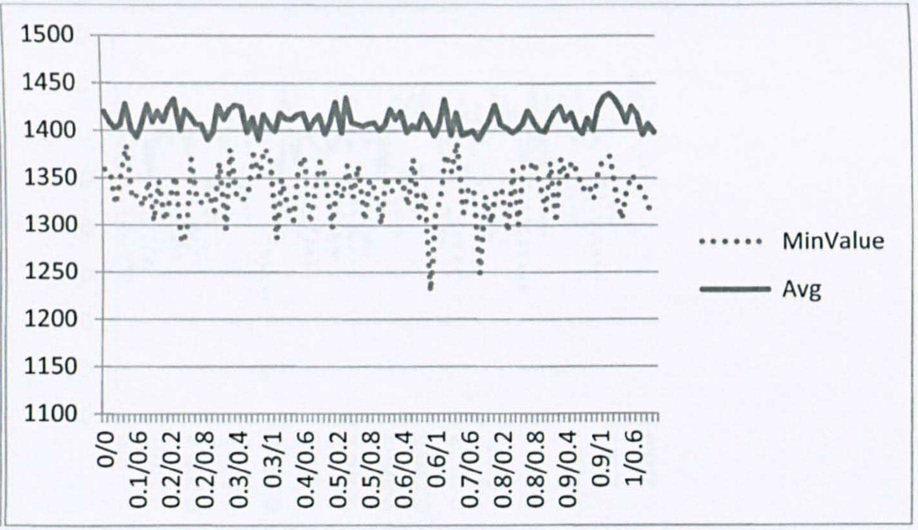
Profiles7 (A)



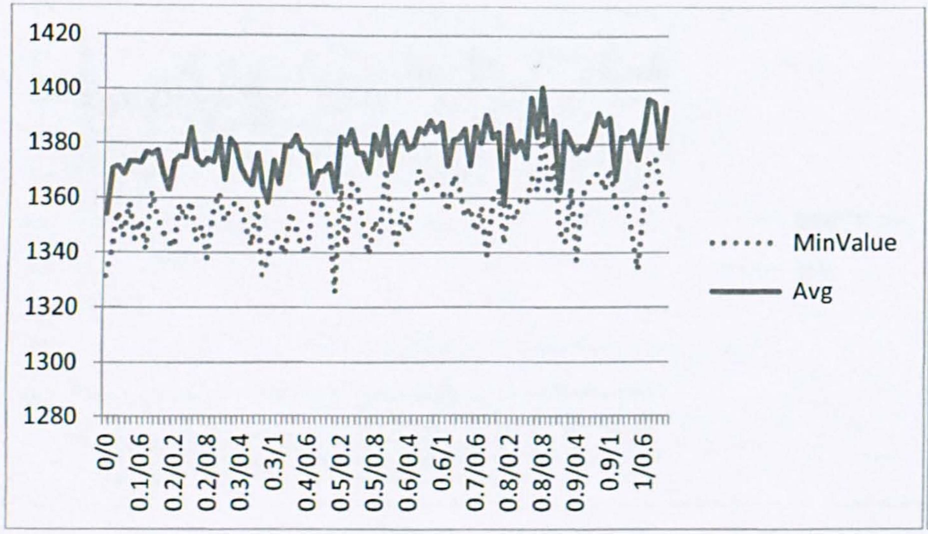
Profiles7 (L)



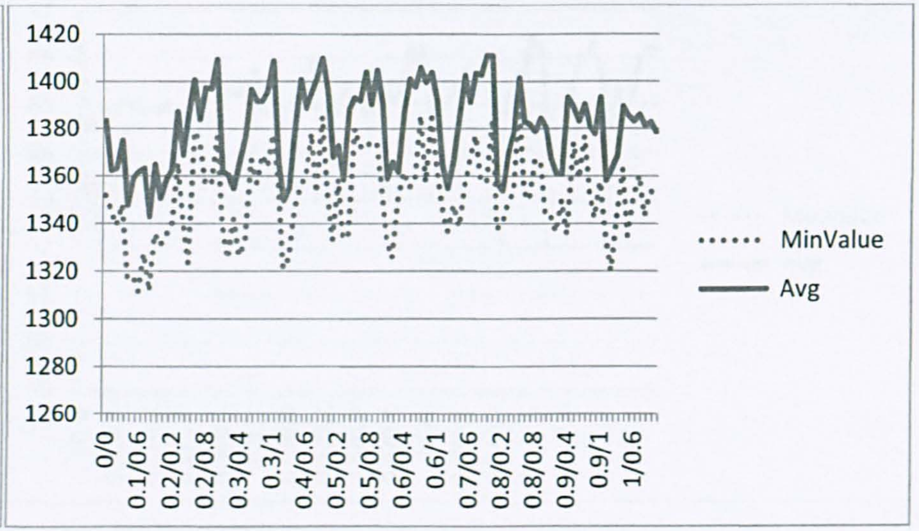
Profiles8 (A)



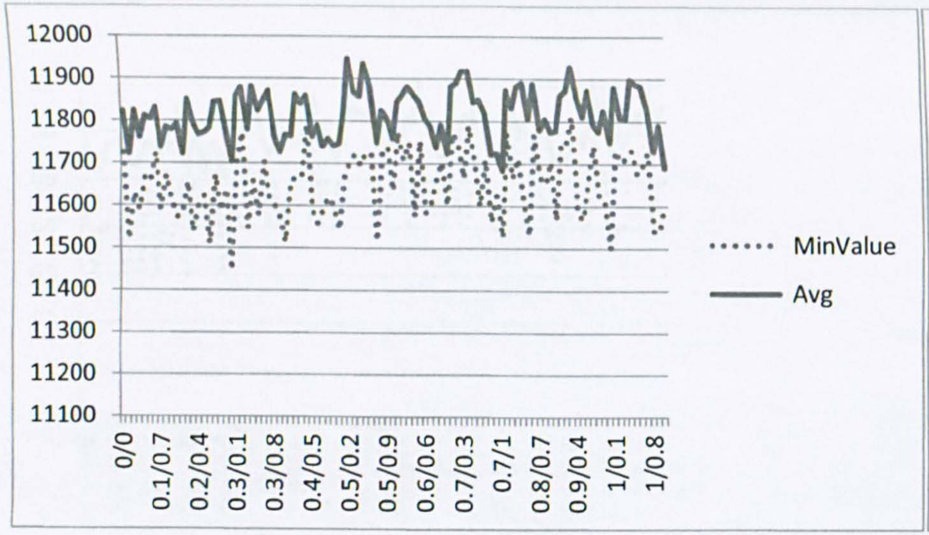
Profiles8 (L)



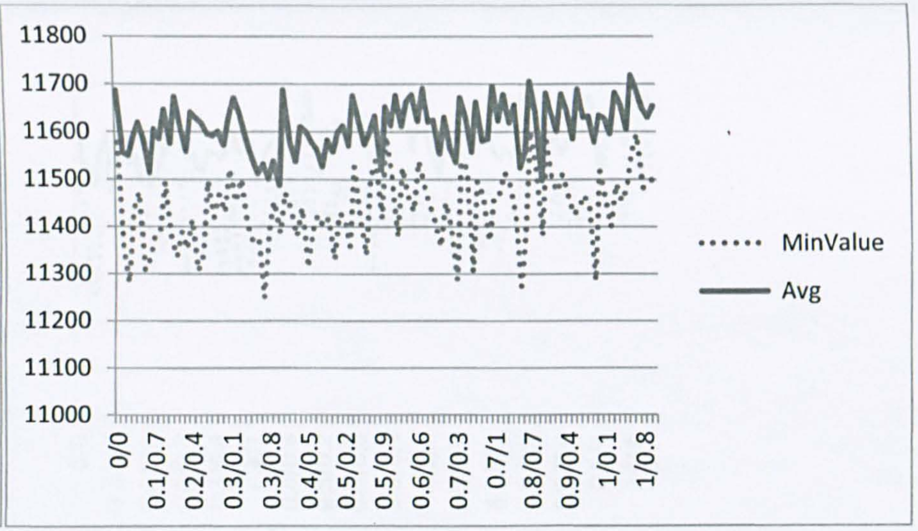
Profiles9 (A)



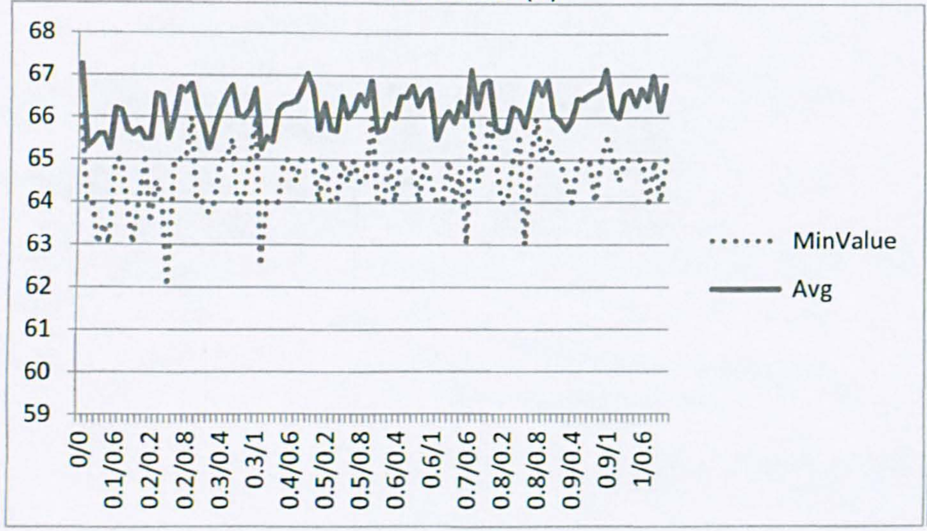
Profiles9 (L)



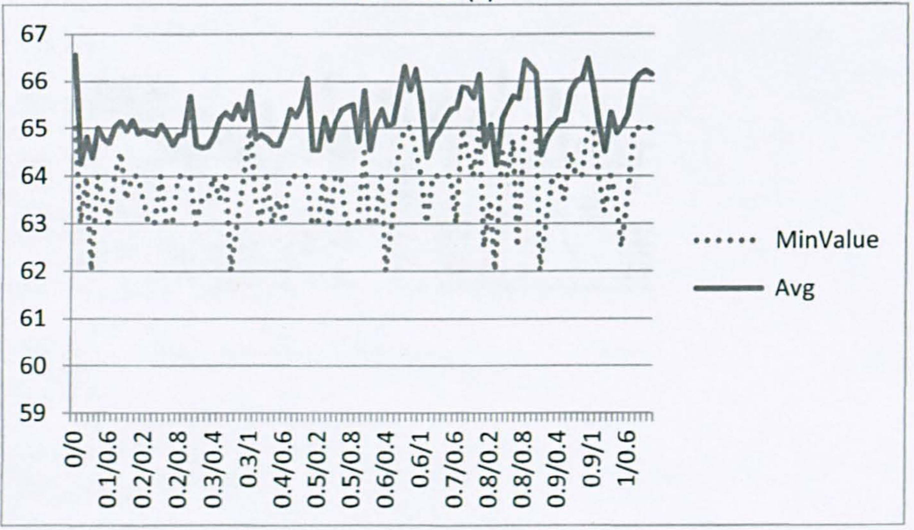
Profiles10 (A)



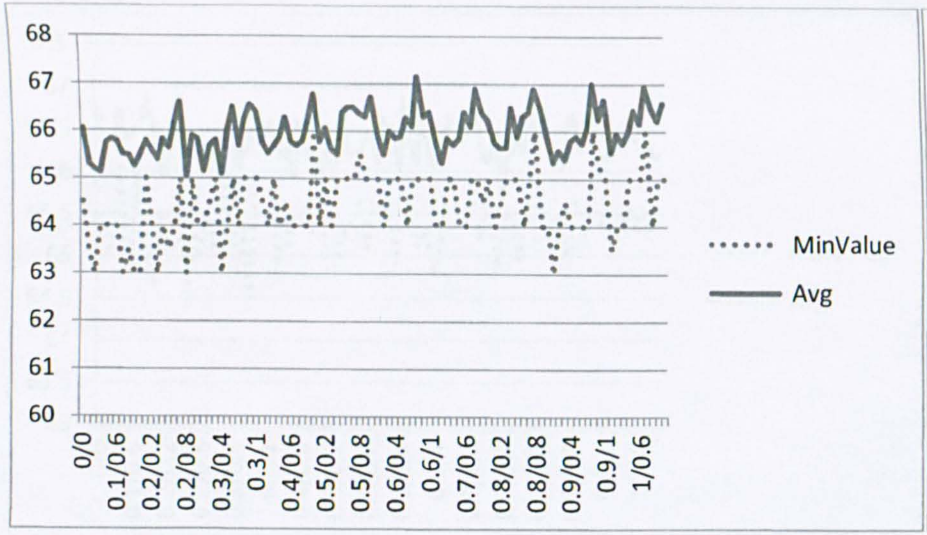
Profiles10 (L)



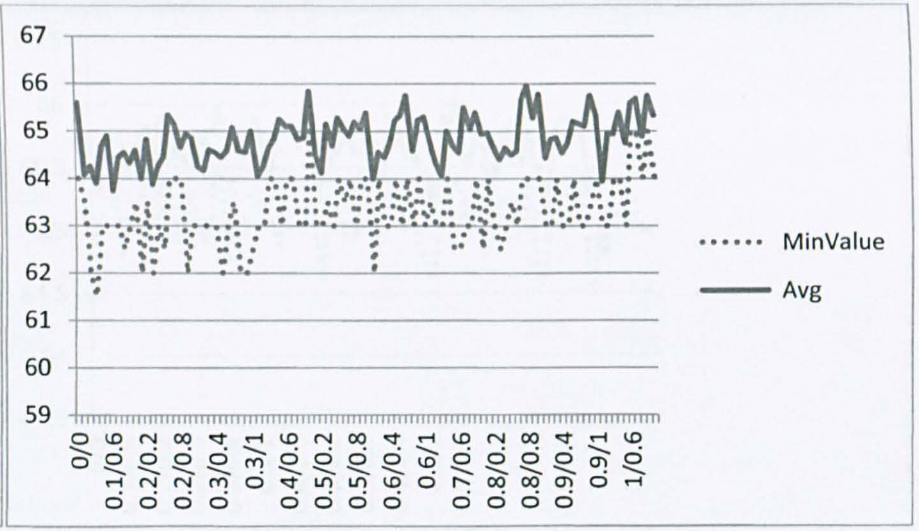
Shapes0 (A)



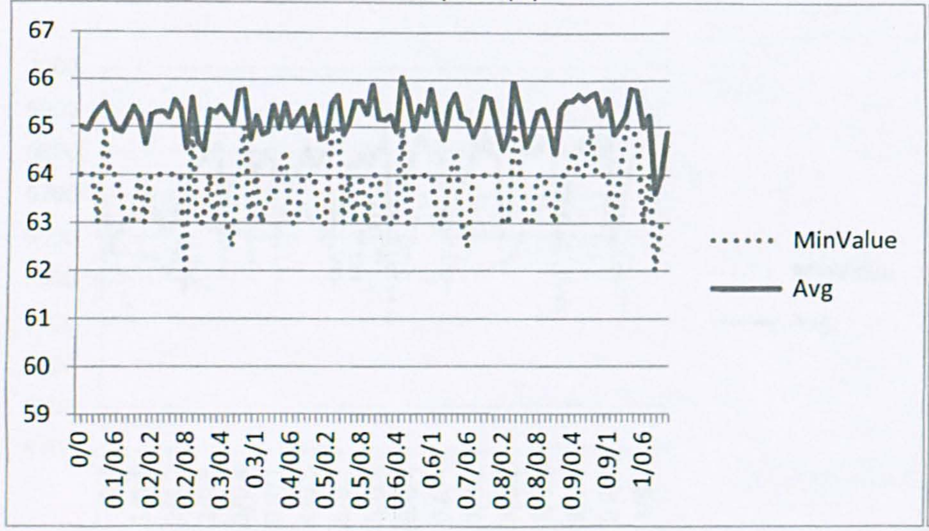
Shapes0 (L)



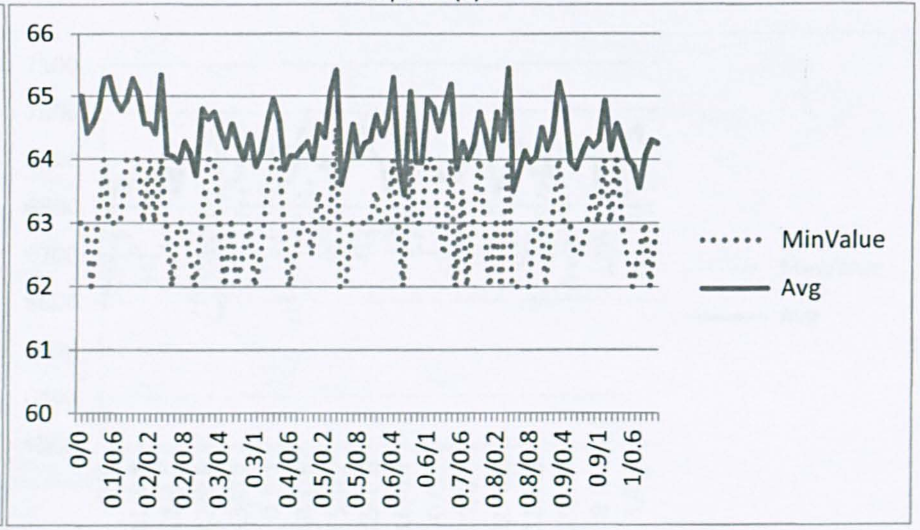
Shapes1 (A)



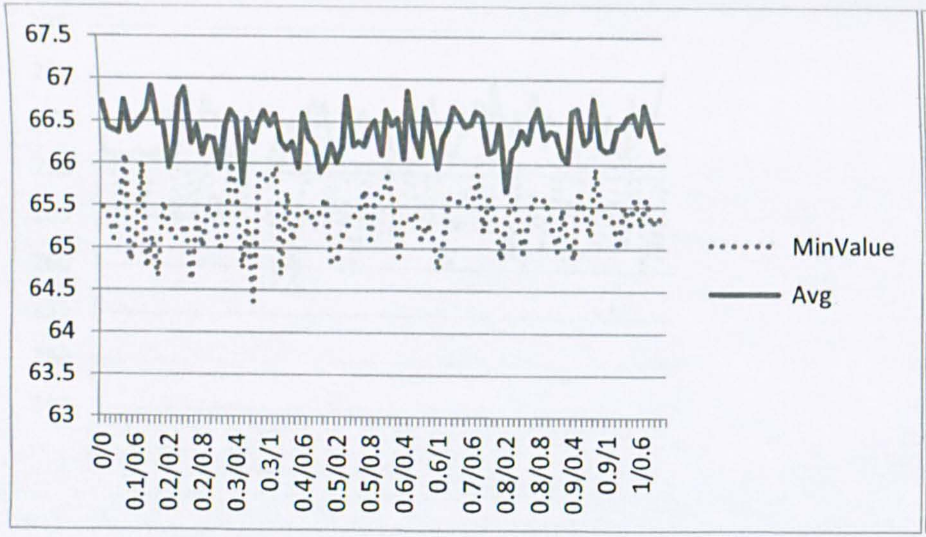
Shapes1 (L)



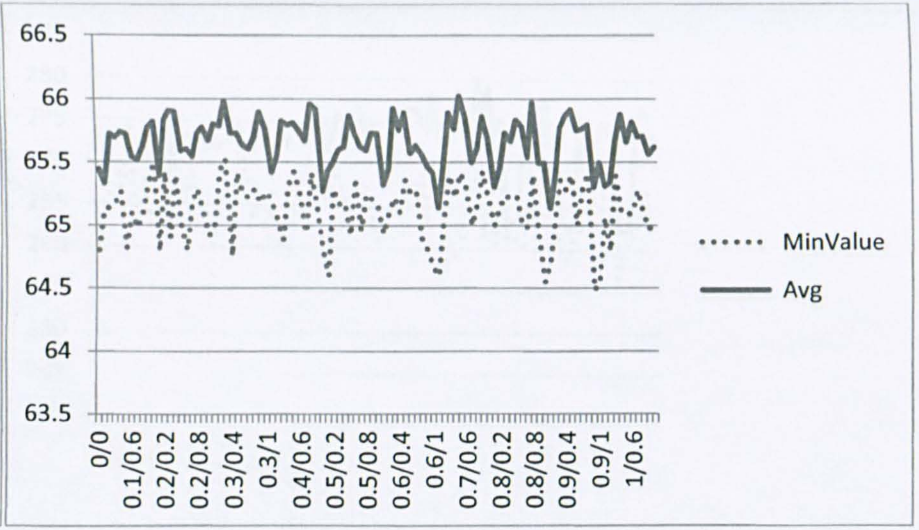
Shapes (A)



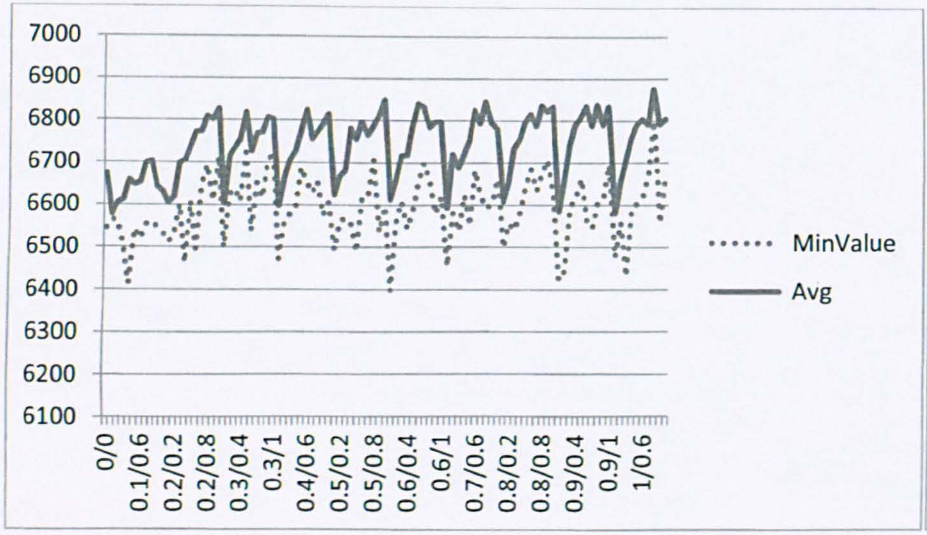
Shapes (L)



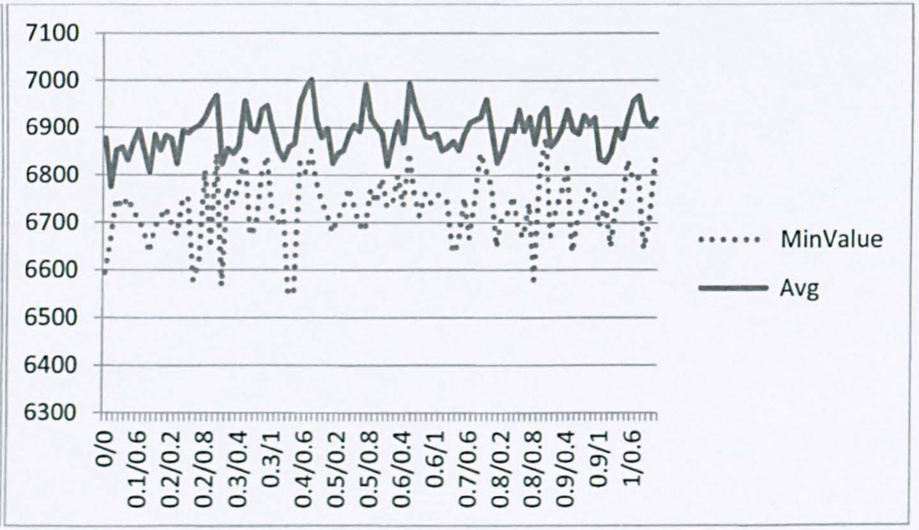
Shirts (A)



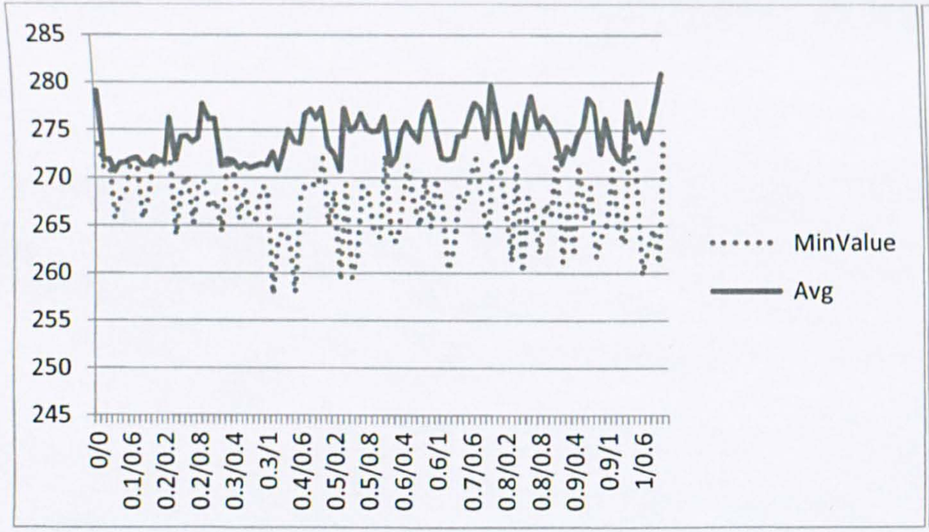
Shirts (L)



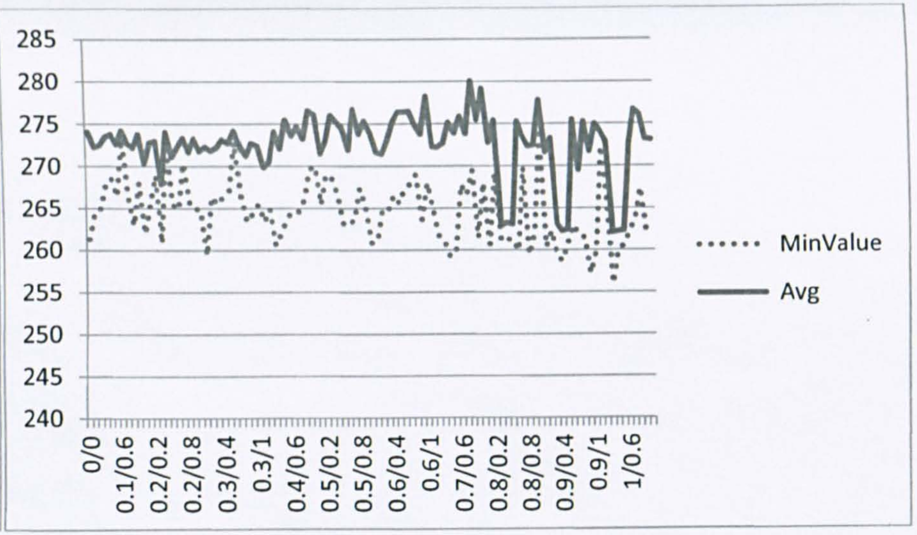
Swim (A)



Swim (L)



Trousers (A)



Trousers (L)