UNIVERSITY OF SOUTHAMPTON

FACULTY OF BUSINESS, LAW AND ART

Southampton Business School

Cutting Patterns for Efficient Production of Irregular-shaped Pieces

by

Ranga Prasad Abeysooriya

Thesis for the degree of Doctor of Philosophy

August 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF BUSINESS, LAW AND ART
Southampton Business School

Doctor of Philosophy in Operational Research

CUTTING PATTERNS FOR EFFICIENT PRODUCTION OF IRREGULAR-SHAPED PIECES

by Ranga Prasad Abeysooriya

The research presented in this thesis belongs to the subject area of operations research. The study investigates and utilises the solution methodologies known as heuristics and local search for three practical problems related to cutting and packing using irregular shapes and multiple bins. From an application point of view, the problems domains remain in manufacturing, specifically where minimising the resources is required to meet a particular outcome. Many manufacturing processes begin with cutting desired items from a stock sheet of material, hence this study focuses on generating efficient cutting patterns, which is applicable in the manufacture of furniture, shoes, tools, ships, and garments.

First, we consider designing an efficient solution procedure for solving two-dimensional irregular shape single bin size bin packing problem and two-dimensional irregular shape multiple bin size bin packing problem. Our intention is to consider alternative strategies such as placement policies, hole-filling and handling rotation of pieces; particularly with unrestricted rotations. Despite the fact that both problems are widely applicable in sheet cutting, their consideration in the literature is limited. To our knowledge, only a few authors have attempted to incorporate the first problem with the unrestricted rotation of pieces while the second problem with unrestricted rotation has not been considered at all. Being applicable in the real world, both the problems require powerful algorithms to determine the arrangement of irregular pieces on stock sheets in order to minimise the material waste. In this thesis, our focus is on developing algorithms to solve each problem efficiently. These algorithms draw on concepts in computational geometry, computer science as well as operations research.

We investigate a set of newly proposed single-pass constructive algorithms that builds a feasible solution by adding pieces sequentially to a packing area defined by a set of bins which can either be homogeneous or heterogeneous. Each problem has a large solution space due to the different combinations of bins and arrangements of irregular pieces. We adopt the optimisation power of local search methods and metaheuristics to find good solutions. As one of the useful heuristic procedures, we use the Jostle heuristic (JS) to solve irregular shape single bin size bin packing problem and irregular shape multiple bin size bin packing problem due to its promising performances in handling both allocation and placement decisions of the pieces together. The Jostle was used in earlier studies for solving irregular strip packing problems and in this study we adopt it first time to solve irregular bin packing problems. Also, our implementation of Jostle handles identifying promising orientation angles of the pieces using the newly proposed angled tuning mechanism when placing pieces. Experimental results reveal that the proposed algorithms can manage different variants of the problem and find solutions with good utilisation of material.

For the third problem of this study, we consider multi-period irregular bin packing problem with use of residuals. This allows using leftovers of a certain period which are usable as input stock material for the next periods. Here, we expand the previous work on irregular bin packing algorithms for heterogeneous stock sheets to consider the inventory and production process of sheet cutting. We propose two models to test the impact of a variety of operational policies around the retention and reuse of residual materials in the sheet cutting process. It also examines the cost sensitivity of using residuals with respect to nine practical scenarios within those operational policies. The computational results demonstrate that the proposed multi-period approach with residuals derives better results than solving each order individually for a selected set of operational scenarios and disclose which policy would be more advantageous to operate in each scenario. The results facilitate developing a tool to guide the manufacturers to take effective decisions based on the scenarios applicable to their sheet cutting process.

Key words: *Irregular shape bin packing, Jostle, Usable leftovers*

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Ranga Prasad Abeysooriya , declare that the thesis entitled *Cutting Patterns for Efficient Production of Irregular-shaped Pieces* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- none of this work has been published before submission

Signed:.........*R. Abeysooriya*..

Date:............19.07.2017..

Acknowledgements

Undertaking this PhD has been a life-changing experience for me which would not have been possible without the support and guidance from numerous people.

I would first like to express my heartfelt gratitude to my supervisors Professor Julia Bennell and Dr Antonio Martinez-Sykora, for bearing all my concerns and guiding me throughout the study. I have been lucky to have such enthusiastic supervisors who have been a tremendous support for me in many ways throughout this massive learning experience.

Next, my sincere gratitude goes to the Head of School of Business and Law of University of Southampton and the Director of the postgraduate research programme and all the members of staff at the University of Southampton for their enormous support in several aspects throughout this PhD journey. In particular, I would like to thank the examiners of MPhil to PhD upgrade and anonymous reviewers of research articles, for their invaluable reviews and suggestions on my work.

I am indebted to the Faculty of Business, Law and Arts of University of Southampton for funding my PhD with the aid of Faculty scholarship. The faculty funds provided me with the opportunity to attend conferences which provided me the opportunity to meet my research community all over the world, including experts in my research area. Also, I appreciate the financial support by the Association of European Operational Research Societies (EURO) when I was attending some internationally recognised workshops in operations research.

Completing this PhD would have been much more difficult without the support and friendship provided by the other colleagues of the Business School and our research group CORMSIS in Southampton. A very special gratitude goes to my friend Carlos Lamas-Fernandez for his invaluable feedback and advises especially during the hard times.

I am also grateful to all the staff of University of Moratuwa, Sri Lanka for allowing me to work through my PhD by releasing me from all my duties and offering me a study leave to complete it without any trouble.

Finally, but by no means least, I express my gratitude to my beloved wife Tolusha, for her continued support, encouragement and patience. Throughout this journey, I was so lucky to have invaluable blessings of my dearest family and in-laws who have always have faith in me and motivating me to pursue my dreams.

Southampton, United Kingdom, May 2017

Abbreviation

1D :	One-dimensional
2D :	Two-dimensional
2D IBPPUL :	**T**wo **D**imensional **I**rregular shape **B**in **P**acking **P**roblem with **U**sable **L**eftovers
2D IMBSBPP :	**T**wo **D**imensional **I**rregular shape **M**ultiple **B**in **S**ize **B**in **P**acking **P**roblem
2D ISBSBPP :	**T**wo **D**imensional **I**rregular shape **S**ingle **B**in **S**ize **B**in **P**acking **P**roblem
BFD :	**B**est **F**it **D**ecreasing
BL :	**B**ottom **L**eft (placement rule)
BPP :	**B**in **P**acking **P**roblem
C&P :	**C**utting and **P**acking
CA :	**C**onstructive **A**pproach
CAA :	**C**onstructive **A**pproach with minimum **A**rea
CAD :	**C**onstructive **A**pproach with maximum **A**djacency
CC :	**C**ircle **C**overing
CSP :	**C**utting **S**tock **P**roblem
DJD :	**D**jang & **F**inch
FF :	**F**irst **F**it
FFD :	**F**irst **F**it **D**ecreasing
FL :	**F**iller
GA :	**G**enetic **A**lgorithm
GD :	**G**reedy **D**decisions
GRASP :	**G**reedy **R**andomised **A**daptive **P**rocedure
HGA :	**H**ybrid **G**enetic **A**lgorithm
HGAIJ :	**H**ybrid approach of **G**enetic **A**lgorithm and **I**terated **J**ostling
HGAJ :	**H**ybrid approach of **G**enetic **A**lgorithm and **J**ostling
IBPP :	**I**rregular **B**in **P**acking **P**roblem
IFP :	**I**nner-**F**it **P**olygon
IJRAB :	**I**terated **J**ostling with **R**andom **A**ssignment of **B**ins
IJS :	**I**terated **J**ostling
ILS :	**I**terated **L**ocal **S**earch
IP :	**I**nteger **P**rogramming
JP :	**J**igsaw **P**uzzle
JS :	**J**ostling

LB :	**L**arge **B**ins
MB :	**M**edium **B**ins
MBSBPP :	**M**ultiple **B**in **S**ize **B**in **P**acking **P**roblem
MIP :	**M**ixed **I**nteger **P**rogramming
ML :	**M**inimum **L**ength (placement rule)
MSSCSP :	**M**ultiple **S**tock **S**ize **C**utting **S**tock **P**roblem
MU :	**M**aximum **U**tilisation (placement rule)
NF :	**N**ext **F**it
NFD :	**N**ext **F**it **D**ecreasing
NFP :	**N**o-**F**it **P**olygon
ODP :	**O**pen **D**imension **P**roblem
PBP :	**P**artial **P**acking **P**roblem
RBPP :	**R**esidual **B**in **P**acking **P**roblem
RCSP :	**R**esidual **C**utting **S**tock **P**roblem
RR :	**R**estricted **R**otation
SA :	**S**imulated **A**nnealing
SB :	**S**mall **B**ins
SBSBPP :	**S**ingle **B**in **S**ize **B**in **P**acking **P**roblem
SPBCH :	**S**equential **P**acking with a **B**in **C**entric **H**euristic
SSSCSP :	**S**ingle **S**tock **S**ize **C**utting **S**tock **P**roblem
TS :	**T**abu **S**earch
UL :	**U**sable **L**eftovers
UR :	**U**nrestricted **R**otation
VNS :	**V**ariable **N**eighbourhood **S**earch

Chapter 1

Introduction

Cutting and packing is a very common scenario in day-to-day life. For example, when packing bags to go on a vacation or wrapping gifts for Christmas, we tend to find the most efficient way of utilising the available space. Also, an efficient cutting and packing is regarded to a significant degree when planning the layout of a magazine, arranging articles on a web page, or tiling walls of an office space. Above all, manufacturing industries such as garment, furniture, component and tools use cutting and packing to make efficient use of material. Examples include; a metal casing of a personal computer requires assembly of several parts to cut from a sheet metal, a decorative sofa requires different cut shapes from a set of cushion foam sheets, a garment requires to cut panels from a roll of fabric and an assembly of a ship requires cut metal parts from sheets. In each of these cases, manufacturers are either aiming at minimising the material waste or maximising the value of cut pieces as much as possible under essential constraints such as overlapping and containment of pieces.

Out of numerous application areas, this research is based on the applications arise in business and manufacturing. The main focus of this study is on applications related to sheet cutting industries where irregular shaped items are cut from sheets of material. Therefore the targeted problems of the study are mainly laid under the category of two-dimensional (2D) irregular shape problems with multiple stock sheets (or bins). In general, the packing problems with irregular pieces is defined as arranging a given set of irregular pieces inside a multiple number of bins so that the overall utilisation of bin spaces is maximised. This problem can also be denoted as a waste minimization problem considering the minimization of trim loss which occurs when irregular items are cut out of the stock sheet material. Alternatively, some practical problems related to irregular shape bin packing consider objectives such as reducing the number of bins, or reducing the cost of bins.

1.1 Background

In general, the Cutting & Packing (C&P) is a category of optimisation problems that is focused on finding a good arrangement of multiple items inside large containers. The problem category of this thesis is irregular bin packing problems (IBPP) and they are widely applicable in a diverse range of application fields. As a result, a number of researchers have attempted to solve the problems related to this category even though there is limited existing research. The literature review on Chapter 2 reviews those attempts in detail. As a related problem, the irregular shape packing in a single strip; also known as the nesting problem, is one of the mature areas of C&P research. While the problem definition differs, there are many common elements in the solution approaches available for irregular strip packing problems and irregular bin packing problems. Similarly, some important features of regular bin packing problems are also applicable when solving IBPP. As in general C&P problems, all the solution methods applicable for IBPP should reach to a feasible solution and finding the best packing layout during a reasonable computation time is the best to be achieved. However, in reality, this is still a challenging task with irregular shapes due to the high computational time it takes to find local optimal solutions. Moreover, finding the global optimal solution with irregular shapes is still far away since the existing research is limited to work out the problem with few pieces.

The irregularity of the pieces, range of orientations of the pieces and different placement orders of the pieces cause IBPPs to have a higher number of solutions. For the problems with such complexities, researchers use heuristic-based techniques combined with meta-heuristics approaches since they provide good solutions at reasonable computational cost.

We have two main motivations for studying IBPPs. First, 2D irregular shape sheet cutting problems have numerous applications. The problem occurs in many industries as metal, wood, paper and plastic, where the application of automatic packing algorithms can yield considerable savings in bin space and number of bins used. Usually, the cost of sheet material is significant in a manufacturing process and a saving in sheet material would lead to significant reduction of cost. According to Guide (2000) reducing waste, on the other hand, is much appreciated in terms of economic and environmental implications. Therefore an efficient algorithm which facilitates a better utilisation of material while reducing trim loss is definitely advantageous for manufacturers. Even though there are good computational approaches to solve a similar packing problem for placing irregular shapes into a single container or to a strip with fixed width and variable length, computational methods developed for packing irregular shapes into the multiple stock sheets are rare in the literature as well.

As an emerging concept in manufacturing, reuse of waste is the second motivation of the study. In most scenarios, sheet cutting processes generate off-cuts which can be reused.

Having the policy to reuse off-cuts than discarding them can lead to a reduction of material waste. However, due to the additional handling and storing it brings, the effort may have a trade-off effect on the overall cost of production. Therefore, we are motivated to investigate the economic impact of production decisions and policies for sheet cutting processes. Such effort will include managerial perspectives when solving the related operations research problems. This will provide useful insights for the managers to take production decision effectively. To our knowledge, so far, no such tool is available related to the irregular shapes sheet cutting process.

1.2 Focused Problems

This thesis investigates and provides solution methods for three problems in the category of irregular shape bin packing problems.

In investigating each problem, it is necessary to define each problem clearly with their objective function and constraints. As a common objective, all three problems cover the objective of input minimization which is common to any irregular shape bin packing problem. As a common set of constraints, all three problems cover the generic constraints of C&P problems, which are non-overlapping constraint among the pieces and containment constraint ensuring that pieces are placed within the boundaries of the bin.

When defining problems, we use the term ”irregular pieces” or ”irregular shapes” to describe a simple polygon which can either be convex or non-convex. The term ”bin” is referred to a rectangular-shaped container with a given width and length. Therefore, in a general sense, in all three problems, a given set of simple polygons is arranged in multiple bins without overlapping each other to minimise an objective. As another two variants, for each problem, we discuss the effect of the restricted rotation of pieces and unrestricted rotation of pieces.

Problem 1: Two-dimensional (2D) irregular shape single size multiple bin packing problem (2D ISBSBPP)

The main objective is to minimise the number of bins used in the solution. It is also assumed that the input number of bins is a very large number.

Problem 2: Two-dimensional irregular shape multiple bin size bin packing problem (2D IMBSBPP)

The main objective is to minimise the sum of bin costs. Out of a finite number of bin types available (i.e. Heterogeneous rectangular bins), the aim is to find a solution with the minimum sum of bin cost where the area of the bin is proportional to its cost. It is also assumed that the input number of bins from each available type is a very large number.

Problem 3: Two-dimensional irregular shape bin packing problem with residual reuse

With reference to this problem, we discuss two aspects. First, we attempt to solve the problem with the aim of minimising the trim loss. A sequence of cut orders with different quantities of pieces are considered as one set of inputs. The other inputs include standard sheets purchased, which can be either one single standard size or multiple standard sizes. The problem is a multi-period irregular shape bin packing problem, in which a set of demand quantities in consecutive periods are considered. The problem considers using leftovers of a certain period which are usable as input stock material for the next periods. Corresponding to each time period, an irregular shape bin packing problem is solved to satisfy the demand quantity of pieces required in that period. The usable leftovers (ULs) gather from one period may be used in future periods and this makes two forms of input bins; *standard bins* and *ULs*, to be considered when solving the irregular bin packing problem at each period. We solve this problem, assuming that the future demand quantity of pieces is unknown when the current order is being processed. Also, the demand quantity of a certain period must be fully met within that period.

1.3 Aim and Scope

The purpose of this study is to provide a contribution to the C&P field by solving a set of practical problems arise in sheet cutting industries. Our objective is to explore the different forms of sheet cutting problems and develop computational methods for each of them with a focus on cutting irregular shaped pieces.

Exploration of the literature reveals two categories of developing computational methods; exact methods which find the globally optimal solution or heuristics and metaheuristic approaches which do not guarantee to find the globally optimal solution but provide good solutions within reasonable computation time. In current studies, the exact methods are limited in solving irregular shapes packing problem only with a small number of pieces (Toledo et al., 2013; Alvarez-Valdes et al., 2013; Cherri et al., 2016). Therefore, we exclude exact approaches and use heuristics and metaheuristics based local search methods to solve the aforementioned three problems. Our focus is on developing fast heuristic methods to generate feasible solutions. Our conjecture is that each problem may contain many local optima and it will require the involvement of search mechanisms to avoid becoming trapped in local optima.

In our investigation, we discuss solutions methods for ISBSBPP, IMBSBPP and IBPP with ULs and expand the investigation of irregular shape bin packing problems. The optimisation routine followed in solving each of the problems relies on the use of geometry as they have to primarily deal with placement of irregular pieces with no overlaps. Accuracy of solutions is one of our primary concerns and we intend to use efficient and accurate computational geometric techniques within the proposed solution generation procedures.

The contribution of this study can be briefly outlined as follows.

The computational methods developed in this study are able to identify the promising rotation of pieces when they are being placed in the bins rather than restricting the piece rotation to a predefined finite set of orientation angles. Only a few papers related to irregular shape packing approaches (e.g. Han et al. (2013), Rocha et al. (2013), Stoyan et al. (2015a), Martinez-Sykora et al. (2017)) have considered this factor, yet their procedures take a substantial amount of computational time which causes the overall computation to be inefficient. This is a realistic requirement for many sheet cutting industries such as sheet metal, foams, artificial leather, plastics which have sheets with homogeneous surfaces. We design a new method to find such promising rotations efficiently. In essence, this allows pieces to rotate within the layout to any angle.

We design a new computational method to solve 2D ISBSBPP and 2D IMBSBPP following the Jostle procedure which was implemented by Dowsland et al. (1998) for an irregular shape strip packing problem with the finite rotation of pieces and hole-filling. Our implementation of Jostle in bin packing problems is new and includes additional features such as unrestricted rotation of pieces which was not available in the previous implementations of the Jostle algorithm. We introduce this method as a fast heuristic for solving irregular shape bin packing problems.

Having algorithms for the 2D ISBSBPP and 2D IMBSBPP allow us to tackle another variant of IBPP by considering usable leftovers. To our knowledge, no literature is available in considering IBPP with ULs. While introducing a new method to solve this problem, we examine the potential opportunities to implement reusing practices in material waste arise in sheet cutting processes by considering ULs in the problem. The existing literature has not discussed well a solution method for this problem where the scope is limited only to 1D cutting stock problems and 2D rectangular item packing problems using ULs. We further evaluate outcomes of such reuse practices and their effect to the material costs, setup costs and handling costs of the sheet cutting process. As one of the major contributions of this study, we present how a set of well-known operational policies is applied within nine different operational scenarios, with regards to saving material and direct costs incur in the sheet cutting processes.

1.4 Research Strategy

This thesis converts three types of real problems in sheet cutting industries to optimisation models and finds solution methods to solve those modelled optimisation problems.

In order to solve each problem, we design different algorithms and then evaluate and validate those to test whether the solution methods can satisfy the objectives associated with each problem. We conduct evaluation and validation through computational experiments based on the quantitative analysis of different features such as improvement of bin utilisation, minimization of trim loss and minimization of cost. We design computational experiments to assess the efficiency of the solution methods and draw conclusions to provide useful insights for the process of sheet cutting, by analysing different geometric features in the derived cut layouts.

The initial phase of this study explores the current practices of C&P research. This includes recognising the existing methods for different C&P problems. In this phase we study and review what has been considered, to identify the research gaps. This fact leads us to solve three different optimisation problems; 2D ISBSBPP, 2D IMBSBPP and 2D IBPP with ULs, by taking into account the degree of rotation of pieces, variety of bin sizes and use of produced leftovers. For each problem, we follow a modelling phase, a solution design & development phase and experiments & validation phase, to achieve the aims of the study. In Figure 1.1, we demonstrate the key steps of each phase to provide a general overview of the flow of this study. In the next chapters we describe each step in detail, justifying the accepted methodology.

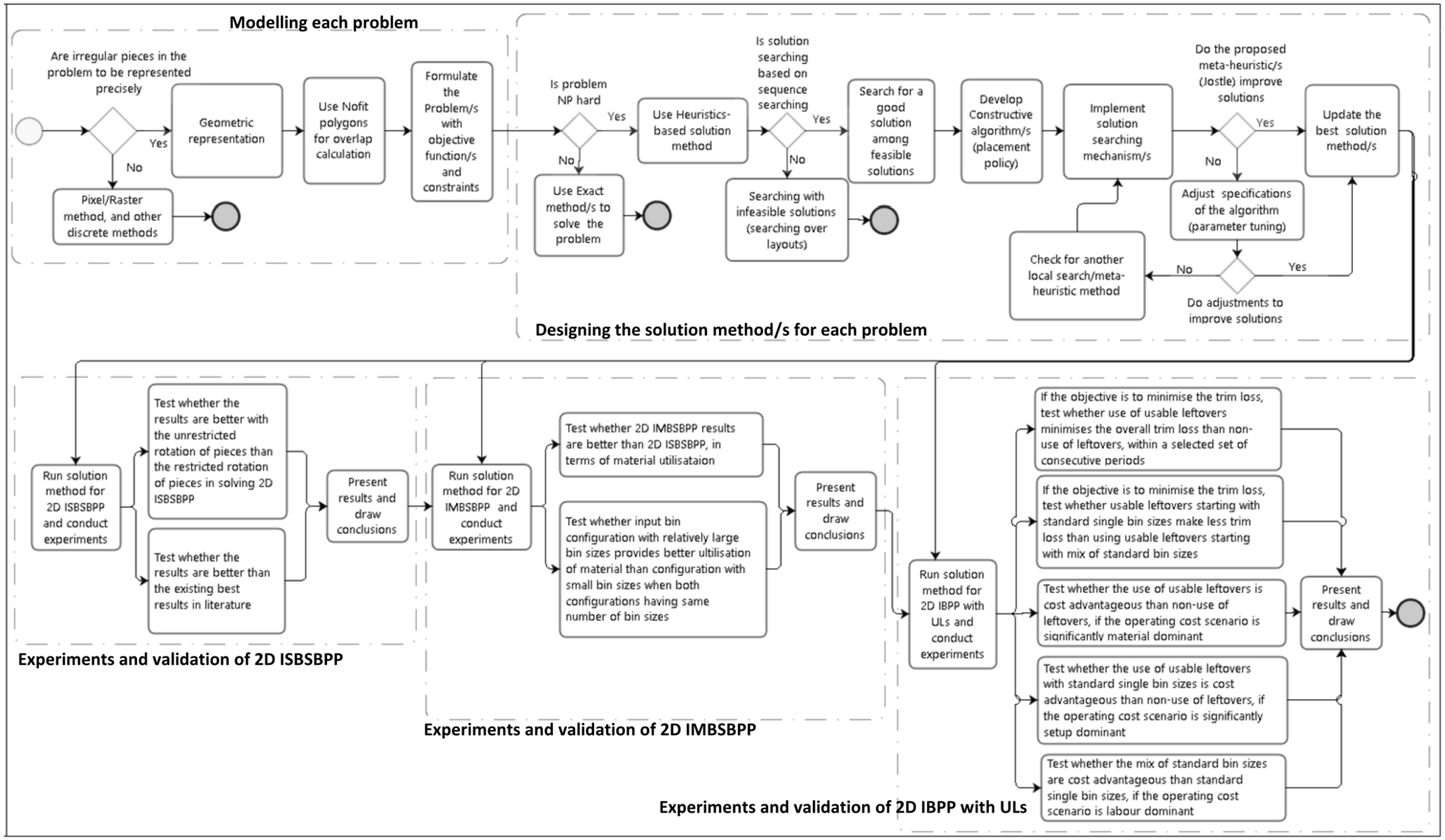

Modelling each problem
Are irregular pieces in the problem to be represented precisely
Yes
No
Geometric representation
Pixel/Raster method, and other discrete methods
Use Nofit polygons for overlap calculation
Formulate the Problem/s with objective function/s and constraints
Is problem NP hard
Yes
No
Use Exact method/s to solve the problem
Use Heuristics-based solution method
Is solution searching based on sequence searching
Yes
No
Searching with infeasible solutions (searching over layouts)
Search for a good solution among feasible solutions
Develop Constructive algorithm/s (placement policy)
Implement solution searching mechanism/s
Do the proposed meta-heuristic/s (Jostle) improve solutions
Yes
No
Update the best solution method/s
Adjust specifications of the algorithm (parameter tuning)
Do adjustments to improve solutions
Yes
No
Check for another local search/meta-heuristic method
Designing the solution method/s for each problem
Run solution method for 2D ISBSBPP and conduct experiments
Test whether the results are better with the unrestricted rotation of pieces than the restricted rotation of pieces in solving 2D ISBSBPP
Test whether the results are better than the existing best results in literature
Present results and draw conclusions
Experiments and validation of 2D ISBSBPP
Run solution method for 2D IMBSBPP and conduct experiments
Test whether 2D IMBSBPP results are better than 2D ISBSBPP, in terms of material utilisataion
Test whether input bin configuration with relatively large bin sizes provides better utilisation of material than configuration with small bin sizes when both configurations having same number of bin sizes
Present results and draw conclusions
Experiments and validation of 2D IMBSBPP
Run solution method for 2D IBPP with ULs and conduct experiments
If the objective is to minimise the trim loss, test whether use of usable leftovers minimises the overall trim loss than non-use of leftovers, within a selected set of consecutive periods
If the objective is to minimise the trim loss, test whether usable leftovers starting with standard single bin sizes make less trim loss than using usable leftovers starting with mix of standard bin sizes
Test whether the use of usable leftovers is cost advantageous than non-use of leftovers, if the operating cost scenario is significantly material dominant
Test whether the use of usable leftovers with standard single bin sizes is cost advantageous than non-use of leftovers, if the operating cost scenario is significantly setup dominant
Test whether the mix of standard bin sizes are cost advantageous than standard single bin sizes, if the operating cost scenario is labour dominant
Present results and draw conclusions
Experiments and validation of 2D IBPP with ULs

Figure 1.1: Research flow

1.5 Outline

The thesis is arranged in seven chapters including this chapter. Chapter 2 reviews literature related to the three problem areas investigated in this thesis. This includes a review of different versions of irregular shape cutting and packing problems and their solution methods.

Chapter 3 presents the methodologies followed that are common to all the problems. The methodology related to formulating problems, developing models, designing algorithms and implementations are discussed separately in the following chapters for each problem addressed.

In Chapter 4, we present our solution method to solve 2D ISBSBPP with free rotation of pieces and discuss the constructive and improvement heuristic approaches in solving the problem.

Chapter 5 presents the details of three solution methods we propose to solve 2D IMBS-BPP in which we deal with heterogeneous bins.

In Chapter 6, we provide an overview of the problem two-dimensional irregular shape cutting in multiple periods, where we discuss how the residuals can be used to minimise the overall usage of input material. Then we extend this discussion towards the economic use of resources for a selected set of production and procurement policies, which applies in several practical scenarios in sheet cutting industries.

Finally, Chapter 7 concludes the overall study with a discussion of the advantages and limitations of the proposed computational methods.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents the literature that relates to this thesis. The thesis focuses mainly on solving three types of two dimensional cutting and packing problems, particularly with irregular shapes packing into rectangular containers. The chapter presents the sources of literature which make a significant contribution to an understanding of the topic while interpreting the major issues, research gaps and the potential areas of future research related to the irregular shape bin packing problems and its variants.

The scope of this review addresses two main areas. First, it provides a general overview of cutting and packing problems and their variants in Section 2.2. Our aim of this is to review and highlight the areas which have been applied in different industries even though they have been considered only a little in existing literature. By introducing these problem types, in Section 2.3 and 2.4, we focus on reviewing the specific problems relevant to the scope of this thesis.

The second interest area is reviewing the existing computational approaches of packing irregular pieces. In this case we start reviewing the methods available for irregular shape strip packing problem which is widely discussed in literature than methods related for IBPP. The main difference of the irregular strip packing problem compared to IBPP is that instead of using multiple bins, the packing uses only one large strip with fixed width and variable length as the stock object to pack the pieces. Although our problems are on IBPP, we review those approaches of strip packing since they provide useful insights and understanding to work with IBPPs.

In Section 2.5 we review the published approaches of irregular shape strip packing problem, covering a wide range of techniques available. The majority of those methods are established on the basis of heuristics and metaheuristics while there is a few noticeable exceptions based on the exact methods. Then we review the computational approaches specific to the irregular shape bin packing problems in Section 2.6.

2.2 Overview of Cutting and Packing Problems

In a manufacturing organization, the quality of cutting and packing procedures greatly affects the utilization of resources; i.e. minimizing input material or maximizing the output. The automation of cutting and packing is highly motivated by the context of mass scale production due to its efficiency in getting quality solutions in quick time. The research in C&P spread over a wide range of problem types occur in various industries. With the improvement of computer technology, research in C&P have been mostly supported by the benefits gained through powerful computer facilities, and become more dynamic area of research. Our intention of this section is to review the different types of problems addressed by the previous authors. The aim of this review is to gain insights into the problem types and identify which problem types been considered rarely by the previous authors. As a guide we follow systematic classifications used to differentiate each problem type separately by their characteristics.

The C&P problems have over 75 years of long history starting from 1940 (Brooks et al., 1940). During this time period, many problems have been investigated and many solution methods have been developed to produce quality layouts. Sweeney and Paternoster (1992) arranged 359 published research articles in chronological order from 1940 to 1990. This is evidence that C&P is not a new stream of research and it has reached to a certain maturity. In such context, it is interesting to investigate which areas of the stream has developed and which areas being addressed less and rarely. The majority of the publications listed by Sweeney and Paternoster (1992) have addressed one dimensional cutting stock problems and two dimensional regular shape packing problems while the publications related to the two dimensional irregular shape packing problems were rarely addressed.

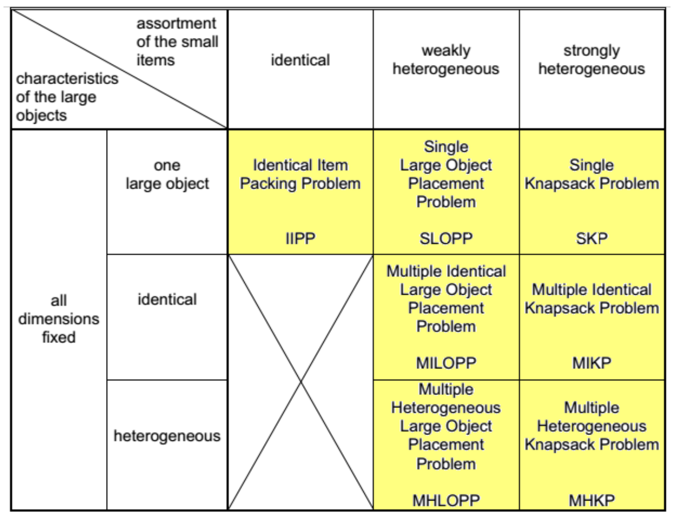
Publications in early years denoted the focused problems in various names (e.g. Cutting stock problem, trim loss problem, bin packing problem, nesting problem, knapsack problem) rather than following a consistent, systematic approach to denote each problem type. This leads to an issue of identifying and concentrating on the problems with similar categories and therefore further studying of problems become harder. Also, a categorization enhances the theoretical background of the research area, which is always supportive of a new person who is interested in C&P research. With the aim of clarifying those problems and standardising their notations, Dyckhoff (1990) introduced a typology based on a logical structure of C&P problems. He described four important characteristics of C&P problems; dimensionality, assignment type of large objects or items, assortment of large objects, and assortment of small items (see in Table 2.1), to differentiate each category of C&P problem. Considering the new problem types after 1990, Wäscher et al. (2007) presented an improved typology for C&P. In one sense this is an extension of the earlier typology, since a part of the Wäscher et al. (2007)'s typology was based on the classification criteria presented in Dyckhoff (1990). Also, the new typology introduces new categories of problems.

According to Wäscher et al. (2007), the new typology addressees *six basic types* of C&P problems based on the two characteristics; *kind of assignment* and *assortment of small items*. The typology extends these six basic problems into a set of *intermediate problems* based on the characteristic of *assortment of large objects*. The characteristic; *kind of assignment*, categories problems as *output maximization problems* and *input minimization problems*. The output maximization problems focus on cutting small items as much as possible within a (given) limited number of large objects, whereas the input minimization emphasizes on minimizing the number of large objects or the space occupied by the large objects required to pack the items. Based on the characteristics of small item, which Wäscher et al. (2007) denoted as *degree of assortment of small items*, classifies items as identical items, weakly heterogeneous items and strongly heterogeneous items. The large objects are also classified based on the assortment, depending on the number of large objects used; one or many. In the category of output maximization problems, the problem is classified as a single object (for one), identical (for many) and heterogeneous (for many). Similarly, for the category of input minimization problem, the classification was made as identical, weakly heterogeneous and strongly heterogeneous. Figures 2.1 and 2.2 illustrate *14 intermediate problems* described in Wäscher et al. (2007)'s paper. The classification further refines the problem types at its tertiary level (named as *Refined Problem Types*) by adding the properties of *dimensionality* and *shape of small items* as adjectives to the intermediate problem types. Accordingly, the problems were denoted as *1-dimensional*, *2-dimensional* or *3-dimensional* and *regular* or *irregular* shape problems using those adjective phrases. Wäscher et al. (2007)'s typology can be used to identify unexplored or rarely explored areas of cutting and packing problem types. As a systematic typology, it provides some guidance to find new problem types which are necessary to evolve the research stream. Wäscher et al. (2007)'s typology has been cited in many papers since several authors adapt this typology to denote problem they consider in their studies. In one sense, this standardizes the current and future researchers' effort to a common framework so that identifying and referring someone's work become easier.

A common complication of these typologies is, they cannot explicitly differentiate the different variants of the problem. As an example, Trkman and Gradisar (2007) proposed an extension to the scope of Dyckhoff (1990)'s typology. They proposed to include the variant *multi-period* when a certain problem's scope extends to multiple time periods, since the typology only denoted the cutting and packing problems discussed for a certain time period. On the other hand, the available typologies are limited in incorporating the

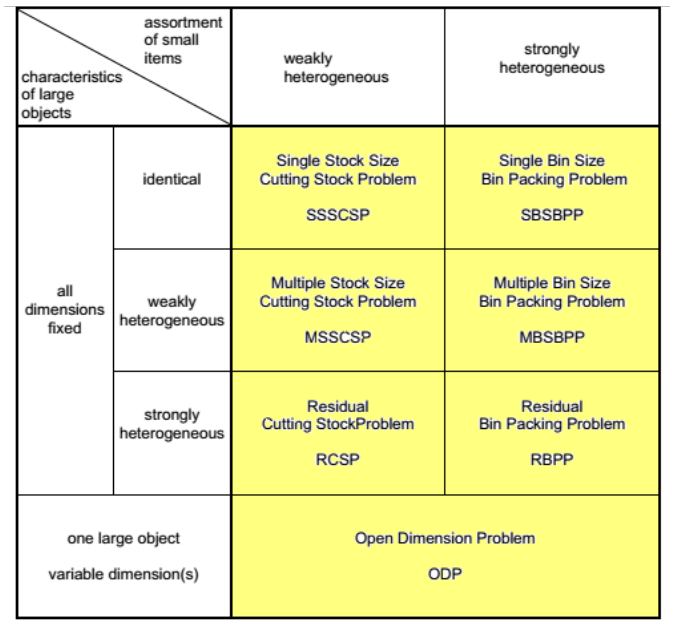
Characteristic	**Description**
Dimensionality	Either one-dimensional, two-dimensional, three-dimensional or N-dimensional where $N > 3$
Assignment type	All objects and a selection of items (needs to fill all the objects available) A selection of objects and all items (need to pack all the items)
Assortment of large objects	One object Identical objects Heterogeneous objects
Assortment of small objects	Few items of different figures Many items of many different figures Many items of relatively few different figures Congruent figures

Table 2.1: Base characteristics of Dyckhoff's typology Dyckhoff (1990)



characteristics of the large objects \ assortment of the small items		identical	weakly heterogeneous	strongly heterogeneous
all dimensions fixed	one large object	Identical Item Packing Problem IIPP	Single Large Object Placement Problem SLOPP	Single Knapsack Problem SKP
	identical		Multiple Identical Large Object Placement Problem MILOPP	Multiple Identical Knapsack Problem MIKP
	heterogeneous		Multiple Heterogeneous Large Object Placement Problem MHLOPP	Multiple Heterogeneous Knapsack Problem MHKP

Figure 2.1: Seven intermediate problem types under output maximization - extracted from Wäscher et al. (2007)



characteristics of large objects \ assortment of small items		weakly heterogeneous	strongly heterogeneous
all dimensions fixed	identical	Single Stock Size Cutting Stock Problem SSSCSP	Single Bin Size Bin Packing Problem SBSBPP
	weakly heterogeneous	Multiple Stock Size Cutting Stock Problem MSSCSP	Multiple Bin Size Bin Packing Problem MBSBPP
	strongly heterogeneous	Residual Cutting StockProblem RCSP	Residual Bin Packing Problem RBPP
one large object variable dimension(s)		Open Dimension Problem ODP	

Figure 2.2: Seven intermediate problem types under input minimization - extracted from Wäscher et al. (2007)

practically relevant constraints arise in production processes applying C&P problems. As an example, Morabito and Belluzzo (2007) presented a hardboard cutting problem which is categorized as an extended two-dimensional multiple stock size cutting stock problem, since it includes the additional aspects such as the cutting pattern sequencing. Considering such extensions and variants, recently, Wäscher (2012) discussed a new set of characteristics to describe C&P problems specifically following the different constraints attached to the problems. This includes *input-related constraints*, *output-related constraints*, *assignment constraints of small items*, *orientation constraints of small items*, and *technological constraints*, as depicted in Table 2.2. The Wäscher (2012)'s presentation also discusses some variants for C&P problems based on some factors directly related to the problem. These variants can be categorized based the demand pattern of small items, regularity (i.e. Regular or Irregular) and quality of large objects and placement of items. Exploring such variants helps to understand the characteristics of a particular problem in depth.

Characteristic	**Description**
Input-related Constraints	with limited availability of large objects of a particular type with an upper limit on the number of large objects types
Output-related Constraints	Output small items of a particular type exactly to the right quantity A lower limit on the quantity of small items of a particular type An upper limit on the quantity of small items of a particular type Lower limit and upper limit on the quantity of small items of a particular type
Assignment constraints (limited assignment of items to large objects)	Upper limit on the number of items in a large object Upper limit on the number of different item types Upper limit on the number of different item classes; e.g. colours in a large object Specific item types must not appear in the same large object
Orientation constraints	Fixed orientation, no rotation is allowed Finite orientation, rotation of small items restricted to a finite number of angles Finite orientation, however a small deviations are allowed Unrestricted orientation, free rotation of items
Technological constraints (based on the type of permitted cuts)	Guillotine unrestricted Guillotine restricted

Table 2.2: New characteristics proposed by Wäscher (2012)

Wäscher et al. (2007) reviewed the 413 C&P papers from 1995 to 2004 and categorized the problems according to their typology. Out of the statistics for considering each problem type, it is interesting to investigate the rarely addressed problem types which have considerable practical value. According to their study, the input minimization problem of packing 2D irregular with multiple stock sheets is one of the areas being rarely discussed in the literature 1995 to 2004.

Within the scope of this chapter, one of our attention is on such rarely discussed input minimization problems related for 2D irregular shapes. Here onward we discuss about those rarely discussed input minimization problems related for 2D irregular shapes, following the terminology proposed in Wäscher et al. (2007)'s typology.

In comparison, the number papers on 2D irregular shapes with open dimension problems are higher than 2D irregular SSSCSP, 2D irregular SBSBPP, 2D irregular MSSCSP, 2D irregular MBSBPP, 2D irregular RCSP or 2D irregular RBPP from 1995 to 2004. The paper reveals only two papers considered 2D irregular SSSCSP and one paper considered the 2D irregular SBSBPP, compared to the 49 papers addressed 2D irregular ODP. Recently Martinez-Sykora et al. (2015) and Xu (2016) comment that number of researches in 2D irregular CSP are lower than 2D irregular strip packing problem. On the other hand, the irregular shape packing with multiple stock sheets has greater applicability in several industries such as ship building, shoe making, garments (Xu, 2016), glass cutting (Han et al., 2013), leather (Baldacci et al., 2014), furniture (Baldacci et al., 2014) and ceramic plate cutting (Martinez-Sykora et al., 2017) industries. Considering this research gap and the practical importance, this thesis addresses on 2D ISBSBPP, 2D IMBSBPP and 2D IMBSBPP with usable leftovers. The three problems are addressed considering the appropriate problem variants gathered from industry related requirements. As an example, unrestricted rotation of pieces is one scheme that can be applied in cutting sheet metal, ceramic plates, foams, and artificial leather, to get a better utilization of material (Rocha et al., 2013; Martinez-Sykora et al., 2017).

In the next section, a review of the related literature focusing particularly on the problems considered in this thesis.

2.3 Overview of 2D Irregular Bin Packing Problems/Cutting Stock problems

Only a limited number of publications considered the bin packing problem with irregular pieces. As a result, the computational methods for irregular bin packing problems are limited in the literature. In this section, we only present the previous work on the irregular bin packing problems and their solution approaches in brief. Details of these solution approaches and their computation methods are reviewed in Section 2.6 of this chapter.

According to Lopez-Camacho et al. (2013a), a solution method for irregular bin packing problems focuses on two types of heuristics; 1) bin selection heuristic to pack the next piece and 2) the placement heuristic which decides exactly where to locate the piece within the selected bin. Lopez-Camacho et al. (2013a) is one of the studies considered on irregular bin packing problems with homogeneous bins. The problem is modelled in a way that the pieces are restricted to rotate only for a pre-defined set of angles. They used several heuristic methods to solve the problem. Lopez-Camacho et al. (2013a), Lopez-Camacho et al. (2013b), Lopez-Camacho et al. (2014), and Terashima-Marín et al. (2010) all described a hyper-heuristic approach based on a genetic algorithm for the 2D BPP for regular and irregular shapes. They describe several selection heuristics to solve the allocation of pieces to bins including variants of First fit, Best fit, Filler and Djang & Finch (DJD). They determined the placement position of the pieces inside the allocated bin using the placement rule; Bottom-Left (BL) algorithm developed by Jakobs (1996) or an improved bottom left version proposed by Liu and Teng (1999) or a variant of the constructive approach by Hifi and M'Hallah (2003). We further review the details of these approaches in Section 2.5.1.

Song and Bennell (2014) proposed a column generation procedure to solve a related problem called irregular shape cutting stock problem. However the cutting stock problem has many copies of each piece leading to cut multiple copies of a particular layout to meet demand. Song and Bennell (2014) investigated three solution approaches; column generation, adapted column generation that permits fewer patterns, and a sequential heuristic procedure to compare the quality of packing solutions. Song and Bennell (2014)'s approach determines required number of copies of each layout by solving the master problem and the sub-problem is solved heuristically by using the beam search approach to generate a bin layout.

While above-mentioned studies were limited to restricted rotation of pieces, Han et al. (2013) and Martinez-Sykora et al. (2015) considered placing pieces with unrestricted rotation into multiple identical stock sheets, particularly targeting the application of glass cutting. The considered pieces were limited to convex and adapted the requirement of guillotine cuts while placing the pieces. The mirror images of the pieces were also considered in these cases. These two studies are good examples of different variants attached to irregular bin packing problems as they covered several variants discussed in Wäscher (2012).

Martinez-Sykora et al. (2017) presented a study of 2D ISBSBPP and considered both convex and non-convex polygons with unrestricted rotation of pieces. They presented a constructive procedure to pack pieces, allowing both free orientation and finite set of rotations. This paper discussed 5 different strategies of assigning pieces to the bins and compared the performance of them. In order to satisfy a feasible placement; while considering the orientation aspects, they proposed a Mixed Integer Programming (MIP) model. Two local search methods were introduced to search over the different assignments of pieces among the bins.

All the above-mentioned approaches are focused on homogeneous bins where the sizes of the rectangular bins are identical. As an exception, Babu and Babu (2001) presented an idea to pack irregular pieces into the irregular shape surfaces. This is irregular shapes packing in multiple irregular stock sheets. The solution method considers a sequence of surfaces; i.e. irregular stock sheets, placement order of pieces and rotation angle of pieces, where the assignment of pieces to the bins is decided according to this sequence. The pieces are placed into the bins using the bottom-left heuristic following the piece order and they propose a genetic algorithm to search over these sequences. Babu and Babu (2001) packed irregular pieces with restricted rotation angles given by an integer degree from 0 to 89. The mirror images of pieces are also considered in this case. Unlike the previous approaches, Babu and Babu (2001) use discrete placement positions when placing the pieces. Baldacci et al. (2014) introduced a heuristic approach to solve the irregular multiple stock-size cutting stock problem with the objective of minimizing the total cost of the used stock sheets, focusing on the leather and garment industry. Baldacci et al. (2014) omit the defect area on stock surfaces and packed pieces into the irregular shaped bins. Similar to Babu and Babu (2001), Baldacci et al. (2014) employs the Discrete/raster representation of pieces and irregular shaped bin. One major issue of this Discrete/raster representation is the accuracy of representing the exact shape of the irregular pieces. The technique approximates the shape profile of pieces and makes mislead the overlapping tests in some cases either by showing pieces are overlapped when they are actually not or by showing pieces are touched when they are actually not touched. In Chapter 3 we discuss characteristics of this representation technique in detail.

In summary, it is clear that there are different forms of irregular bin packing problems and only a few of them were considered in the literature. Out of all the approaches, the irregular MBSBPP is one area that hasn't been considered with unrestricted rotation of pieces. In terms of practical applications, industries such as sheet metal, foams, furniture requires the cutting of irregular shapes from sheets having different standard sheet sizes. The placement-orientation are not restricted in most cases since the surfaces of sheets are homogeneous. On the other hand, we believe that the existing literature available for solving 2D ISBSBPP can be further improved by applying efficient heuristic techniques which can provide reasonably good solutions in quick time so that they can be implemented easily in production and procurement environments.

2.4 Overview of Problems considering Usable Leftovers

Another important trend in manufacturing scenarios is checking for the reusing possibilities of material during the manufacturing process. This is one of the well known philosophies of manufacturing more biased toward the material saving and environmental friendliness. In this section, we review the literature related to C&P problems with off-cuts. In such problems, the off-cuts which smaller than a standard bin size used in the C&P problem are considered to be reused in future orders. To our knowledge, no literature is available for two-dimensional irregular shapes with usable leftovers. Therefore, our discussion includes other studies related to one-dimensional cutting problems and two-dimensional regular shape cutting problems, using leftovers. Reviewing the solution approaches of those problems are important as they contain useful concepts that can be applicable to the solution methods for irregular shape cutting problems. Leftovers are reusable when their sizes are adequate to reuse as input material for the next orders. Cherri et al. (2009) categorized leftovers either as small" which considered as scrap or sufficiently large" which is considered as retail (usable leftovers) to return to the warehouse.

There are very few investigations that consider usable leftovers in cutting and packing problems and most of them are found with one-dimensional cutting stock problems (1DCSP) (Kantorovich, 1960; Brown, 1971; Roodman, 1986). Scheithauer (1991) investigated this problem by satisfying customer demand so that cost is minimized. They calculated the cost of material as the cost of stock length minus the cost of residual lengths. In their methodology, the residual lengths are considered using a linear programming model and they claim that the same approach can be used to solve general two-dimensional guillotine cutting problem with rectangular pieces. In their study, Sinuany-Stern and Weiner (1994) considered two objectives when formulating 1DCSP with usable leftovers. Out of those two, the first priority is assigned for trim-loss minimization and then on maximizing the quantity of leftovers accumulated for one bar (Sinuany-Stern and Weiner, 1994). In order to solve this problem Sinuany-Stern and Weiner (1994) used a heuristic approach.

Since 1999, the research work for one-dimensional cutting stock problem with usable leftovers had increased significantly. The major studies include Gradišar et al. (1999b), Trkman and Gradisar (2007), Cherri et al. (2009), Cui and Yang (2010) and Cherri et al. (2013), with the objective of minimizing the trim loss to satisfy the exact demand. Gradisar and Trkman (2005) discussed two cases, one of which is focused on minimizing trim loss related to utilized stock objects where an order can be fulfilled with enough material availability. If trim loss of a utilized stock object is higher than a defined upper bound, such stocks are returned to the inventory for reuse. In addition to trim loss, Arbib and Marinelli (2005) considered scheduling and cutting preparation costs when developing the cut plans with the possibility of using residuals in subsequent optimizations. Trkman and Gradisar (2007) emphasized that allowing leftovers can cause accumulation of partly used stock lengths in the warehouse. This is a tough situation for a manufacturer since handling and storing become problematic with large number of partly used quantities which are heterogeneous in size. As a solution, Trkman and Gradisar (2007) suggested to allocate handling and storage cost for leftovers so that the objective function is based on the cost of trim loss and the stock return cost of leftovers. As one of the application focused studies, Koch et al. (2009) proposed a decision support application based on an integer programming model for a company in the wood industry, in which the same form of the problem applied.

Reviewing all these investigations, we summarize that leftovers are mainly used to satisfy the primary objective of minimizing the cost of material. The same objective is denoted in different ways; i.e. minimizing the trim loss, in some of the studies. However, recently the problem is attached with several secondary objectives and constraints due to the practical issues that arise when implementing the residual reuse task in a manufacturing environment. Authors are concerned more with demand and supply constraints, inventory constraints and handling of material when determining the number of left-overs and their allowable sizes. Since C&P problems in practice are not independently progressed, they are always linked to the different kind of costs; i.e. trim loss, machine setup, inventory and handling. As examples, Henn and Wäscher (2013) included setup cost in the objective function aiming to minimize the total cost. Combining several objectives, Erjavec et al. (2012) and Wang and Liu (2014) studied 1DCSP and propose a new decision model to reduce trim loss and inventory in the paper industry. Cui et al. (2013) solved two-dimensional guillotine cutting stock problem having the main objective as input-minimization and the auxiliary objective as pattern reduction so that the material and setup cost can be minimized. Usually, the number of machine setups occur during a cutting process has a significant effect on the total cost. Whenever manufacturer has to process a cut layout, there is a need to adjust the knives and some other settings in the cutting machine, which occupies time. Since excessive times of such setups are unproductive, minimization of the number of cutting layouts (i.e. the stock sheets with placed pieces) is usually considered as the second goal when generating cutting plans (Foerster and Wäscher, 2000).

While some studies apply exact methods to solve the problem with usable leftovers, particularly in one dimensional cutting problem, some researchers use heuristic methods to find solutions. Cherri et al. (2014a) presented an interesting discussion on challenges and issues arising when dealing with usable leftovers (ULs). Even though their study is solely focused on 1DCSP, there are some important findings to consider when addressing usable leftovers with two-dimensional cutting and packing problems as well. Therefore, we summarize some interesting facts of their study by providing some related examples from the literature. When dealing with this problem, a clear definition of the context of the application scenario is necessary. Cherri et al. (2014a) identified three situations where the context can be different; 1) cases where the demands of the pieces in subsequent periods are completely unknown, 2) cases where the demands of the pieces in subsequent periods are unknown, but there is a set of positive demands and 3) demands of the subsequent periods are unknown, however their probability distributions are known.

In the first option, manufacture has no other option other than keeping usable leftovers in the warehouse and waiting for a future demand of the items to use them. According to the second case, some items will be in future demand, hence manufacturers can either overproduce these items and store them in the warehouse to fulfill the future demand or generate usable leftovers. In the third approach, only the probability distribution is known so that the manufacturer has to estimate the demand. Similarly, Trkman and Gradisar (2007) discussed about considering consecutive orders operating within a selected time-span to deal with usable leftovers (ULs) by presenting the following three categories; 1) optimization solution of the C&P problem for a given data at one time-period. They refer to this as instantaneous, 2) optimization for two or more consecutive time-periods when the demand and supply of material for all time-periods are known in advance. Therefore, the demands are deterministic for consecutive time periods, and 3) optimization for two or more consecutive time-periods when only the demand for the first period is known.

Another challenge is how to determine the size of the usable leftovers. According to Cherri et al. (2014b), the following categories are available in the literature. 1) Determine the size of the leftover as the length of the largest or the smallest item, 2) Determine the size as a pre-established value, 3) define a 'good' size which is cost effective, by considering possible future demands.

It is also important to find solutions for C&P problems with leftovers within a reasonable computation time. As stated in Cherri et al. (2014b), most of the approaches use heuristic methods which solves the problem in a reasonable time, even though the optimal solution is not guaranteed. On the other hand, unlike a single cut order, the context of this problem has to solve a sequence of cut orders so that it is important to use a solution method which provide solutions for all those cut orders within a reasonable computation time. Controlling the number of different UL types also has a significant effect on the computational complexity (Trkman and Gradisar, 2007; Cui et al., 2016). A higher number of heterogeneous UL types makes the problem complex, hence a mechanism to control UL generation is recommended in Trkman and Gradisar (2007) and Cui et al. (2016).

As an additional issue, Cherri et al. (2014a) considered about the number of UL types generated during the cutting process and its significance in managing the ULs effectively. Usually researchers focus on the ULs generated from standard large objects, so that chances of generating larger UL become higher. However, when dealing with these problems, Cherri et al. (2014a) emphasize the importance of evaluating other factors such as handling off-cuts.

In most cases, the existing models available, either focusing on simple variations of the traditional C&P problems, or broader problem focused on a specific industry (e.g. Textile industry (Gradišar et al., 1997), agricultural light aircraft manufacturing (Abuabara and Morabito, 2009), and wood-processing industry (Koch et al., 2009)). Any of these approaches have not been able to guarantee the investigation of all the possible scenarios of the problem considering varying costs of material, setups and handling off-cuts. Cherri et al. (2014a) stated that a company should have an economic analysis mechanism to compare the cost of handling leftovers. They believed in keeping usable leftovers in the warehouse is economical for 1DCSP when the cost of the material is high, since the savings can compensate for the costs of handling, transportation and warehouse (Trkman and Gradisar, 2007; Cherri et al., 2014a).

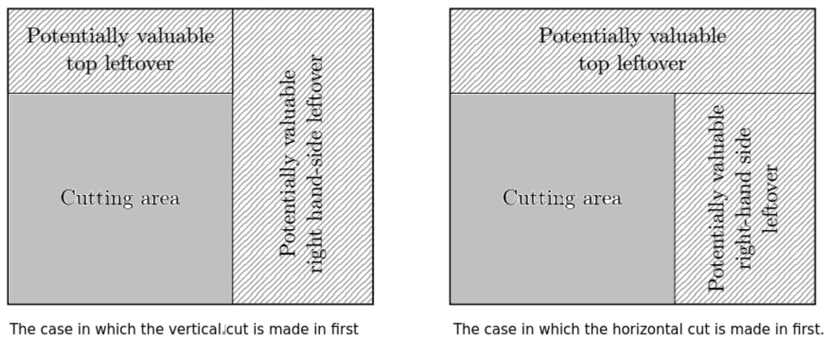

Potentially valuable top leftover
Cutting area
Potentially valuable right-hand-side leftover
The case in which the vertical cut is made in first
Potentially valuable top leftover
Cutting area
Potentially valuable right-hand side leftover
The case in which the horizontal cut is made in first.

Figure 2.3: Ways of generating usable leftovers by Andrade et al. (2014)

Only a handful of research has published in two-dimensional rectangular shape cutting problems with usable leftovers. Andrade et al. (2014) and Andrade et al. (2016) presented two-dimensional residual bin-packing problems with heterogeneous rectangular items considering non-guillotine and guillotine cutting cases respectively. Andrade et al. (2014)'s method arbitrarily assume that two guillotine leftovers (vertical and horizontal) can be produced from each used object that each leftover is generated by a single guillotine cut. The proposed models consist of cutting the item quantity using a set of stock-sheets at the minimum cost. If there are any ties in the solutions of minimum cost, the solution which maximizes the value of the usable leftovers are selected. If there are any ties in the solution with minimum cost and maximum value of leftovers, the solution with the minimum number of usable leftovers is selected. This study was continued by Andrade et al. (2016) and presented two mixed integer programming models for the non-exact two-stage two-dimensional guillotine cutting/packing problem with usable stock-sheet remainders.

In Table 2.3 we summarize a set of solution methods used to solve the C&P problems with usable leftovers. Due to the complexity of the problem, the majority of the existing solution approaches is limited to using heuristic approaches for developing models and solving them.

Table 2.3: Solution methods attempted to solve C&P problems with usable leftovers

Problem type	Solution method	Source
1DCSP	Residual lengths are considered using a linear programming model	Scheithauer (1991)
1DCSP	Algorithm which provides an optimal solution for small problems. A heuristic approach is suggested for the larger problem	Sinuany-Stern and Weiner (1994)
1DCSP	A sequential heuristic procedure to solve the problem on a large scale aiming to minimize trim loss in clothing industry	Gradišar et al. (1997)
1DCSP	Sequential Heuristic Procedure (SHP) applied to heterogeneous stock lengths. Based on this algorithm the computer program called CUT is developed.	Gradišar et al. (1999a)
1DCSP	A hybrid approach for optimizing 1DCSP combining two methods: the pattern-oriented LP-based method, and the item-oriented sequential heuristic procedure.	Gradišar et al. (1999b)
1DCSP	A solution combining a linear programming based method and a sequential A heuristic procedure so that the method leads to almost optimal solutions	Gradisar and Trkman (2005)
1DCSP	A review two frequency used heuristic method; COLA (Gradišar et al., 1997) and CUT Gradišar et al. (1999a), and modifies existing heuristics to solve the problem	Cherri et al. (2009)
2D Regular	Introduced Mixed Integer Programming models for 2D non-guillotine problems with ULs	Andrade et al. (2014)
2D Regular	Two bi-level mathematical programming models to represent two-stage two-dimensional residual bin-packing problem. Based on the of special characteristics of these bi-level models, the models are reformulated as one-level mixed integer programming models.	Andrade et al. (2016)

2.5 Computational Method for Irregular shape Strip Packing Problems

In this section we review key computational methods available for irregular shape strip packing problems. Out of different approaches, the majority are based on the category of heuristics and metaheuristics while there is few methods based on the exact methods. We review both these categories referring the up-to-date research articles published related to this problem.

The section divides into three subsections. First, Section 2.5.1 reviews the methods used to build up a final packing layout. In C&P literature, this is generally denoted as constructive heuristics (Bennell and Oliveira, 2009). As the name implies, these methods determine the construction of a complete solution (i.e. packing layout) piece by piece. Next, in Section 2.5.2 we review those methods that work with complete solutions where changes are made to find an improved solution. In this case, we review the metaheuristic approaches methods used when it is required to find a good solution by searching over feasible set of solutions or required to find a good solution by searching over the physical layout with the allowance of infeasible solutions. Third, our focus is on the exact methods which find global optimal solutions. A review of those methods are presented in Section 2.5.3

2.5.1 Constructive heuristics for irregular shape packing problems

The main task of constructive heuristics is to grow the partial packing layouts. Bennell and Oliveira (2009) referred to these growing partial layouts as *Partial solutions* at each insertion of a piece, till it reaches to a *Complete* packing solution. In this section, the review is specific to the methods described as *single pass constructive methods* where pieces and placed in sequence following a certain placement strategy. Constructive methods follow legal placement of pieces and do not violate the piece overlapping constraint. The method can be used with several repeated efforts using different orderings or different placement strategies to choose the best solution.

In this section, the review is based on different constructive methods, ordering pieces and placement rules discussed for irregular shapes packing problems.

2.5.1.1 Placement rules

The Placement rule finds an acceptable placement position for the next candidate piece to be placed on the packing area according to a certain rule. By applying a placement rule, the authors should ensure that placing position of the piece is contained within the stock sheet without overlapping with other pieces. According to Bennell and Oliveira

(2009), *left most placement* is one of the most popular placement strategies used in irregular packing problems. The rule follows placing pieces toward the left of the stock sheet as much as possible. The rule can be applied with being biased toward the top or bottom of the stock sheet. Art (1966) selected *bottom* side in his study and justified it as *bottom left placement policy*. Qu and Sanders (1987) described the bottom left placement in an alternative way so that the pieces are placed along the bottom of the stock area. Dowsland et al. (2002) attempted bottom left placement (BL) to locate the next piece horizontally farthest to left and vertically to the lowest within the packing space. The objective is to minimize the total length required when placing each piece into a strip. This placement rule has been used later in several research studies (e.g. Jakobs (1996), Dowsland et al. (2002), Gomes and Oliveira (2002)).

Dagli and Tatoglu (1987) proposed pairwise matching of the sides of the pieces. In order to support this, the placement position yields *the minimum rectangular enclosure* is ensured when placing the pieces. Oliveira et al. (2000) described three placement rules. The first one is *Minimum area placement* in which the placement point is selected based on the area of the enclosing rectangle. Pieces are placed in a position where the area of enclosing rectangle is minimum (Figure 2.4a). Accordingly the heuristic controls the build of the partial solution to be as rectangular as possible. Second is *Minimum length placement* where the placement point is selected based on the *length of enclosing rectangle* by placing pieces in a position where the length of enclosing rectangle is minimum (Figure 2.4b). The third rule: *maximum overlap of rectangular enclosures* selects the placement point based on the overlap between rectangular enclosure of the placing piece. Pieces are placed in a position where the overlap between rectangular enclosure of the placing piece is maximum, while keeping no overlap between the pieces(Figure 2.4c).

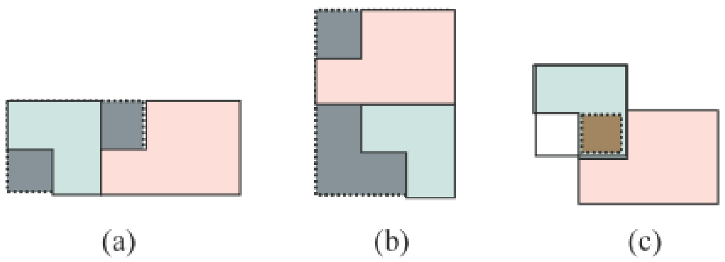


Figure 2.4: Placement rules (extracted from Oliveira et al. (2000))

Błażewicz et al. (1993) paid attention on the gaps created by the packed pieces (i.e Holes) and highlighted the importance of filling holes to obtain a better solution. Dowsland et al. (2002) and Gomes and Oliveira (2002) described holes as the spaces either surrounded by the packed pieces or spaces generated between the edges of stock-sheet and the packed pieces.

While Segenreich and Braga (1986) allowed hole-filling relying completely on grid representation, Burke et al. (2006) used a strategy in which the horizontal movement of the

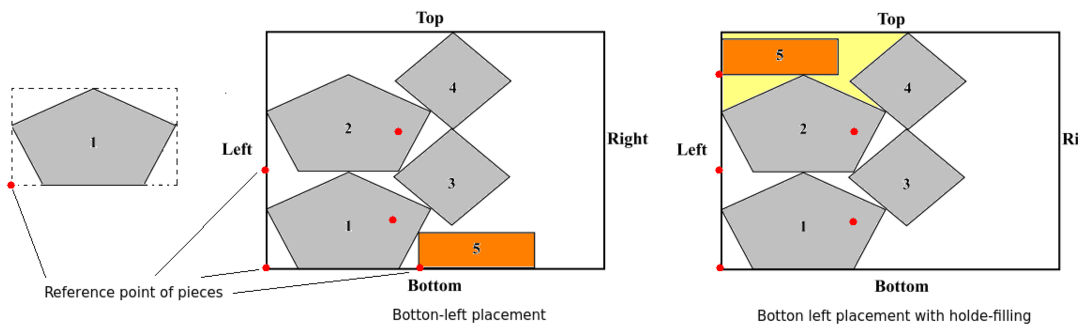

Top
Left
Right
Bottom
1
2
3
4
5
Reference point of pieces
Botton-left placement
Botton left placement with holde-filling

Figure 2.5: Bottom left and Bottom left with Hole-filling (Dowsland et al., 2002)

pieces is discrete and the vertical movement of the piece is continuous when filling holes. Since grid/raster methods are approximation methods, it is interesting to review a hole filling mechanism without involving grid/raster method to ensure high accuracy.

Similar to previous studies, the C&P researchers who believed in constructive heuristic approaches tested their constructive methods using different placement rules with the objective of generating reasonably better solution in a single attempt with less time. One issue of constructive methods is finding the feasible placement positions accurately for each piece in an efficient manner. In order to find feasible placement positions accurately, considering a partial solution with previously placed pieces in fixed locations, one can compute the set of NFPs between the candidate piece and each of those placed pieces. Any part of edge/edges of a certain NFP which are not surrounded by any of other NFPs results the placement region for the next piece to place without overlapping with other pieces. In other words, the edges or parts of the edges of a NFP inside another NFP, do not denote placement positions. When this region is bounded by the inner fit polygon between the candidate piece and the stock sheet object, the corresponding intersecting edges denote the feasible region for the next piece. Gomes and Oliveira (2002) used this method to find the feasible placement region for the next piece. If this feasible region is found by means of set of vertices, then these vertices are the vertices of NFPs and the points created at the intersection of NFP edges which are bounded by the IFP_k as illustrated in Figure 2.6. Accordingly, the the final placement position of the next piece can be found following a placement rule such as bottom-left. This method can be also used to find the feasible placement positions within the holes in the partial solution.

Oliveira et al. (2000) proposed an alternative approach called TOPOS to construct layout. They generate the No-fit polygon between the candidate piece and the whole partial solution which is represented in one polygon. At the insertion of each piece, the candidate piece is placed and merged into the existing packing layout and updates the partial solution. Therefore the whole partial solution is considered as one merged polygon of packed pieces. The reference point of the partial solution is not fixed in the packing area and consists a floating origin, so that the packed pieces have fixed positions among themselves but float within the strip together. However the width

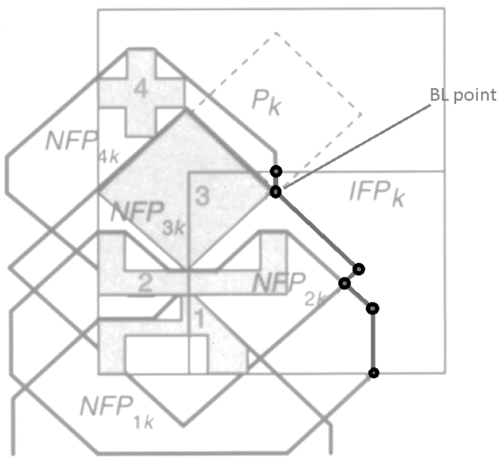


Figure 2.6: Feasible placement region for the next inserting piece (image is from Bennell and Oliveira (2009))

of partial solution is always maintained to be less than or equal to the stock sheet width. The study tested several placement strategies such as *minimum area rectangular enclosure of the new partial solution*, *minimum length rectangular enclosure of the new partial solution* and *maximum overlap between the rectangular enclosure of the pieces placed without overlapping pieces.* Since the existing partial solution is not fixed, the partial solution can adjust their position within the boundary of packing space (Oliveira et al., 2000). This allows a new piece to fit into the partial solution along any side of it.

The original TOPOS cannot detect the holes surrounded by the pieces and this is considered to be a major limitation of TOPOS.

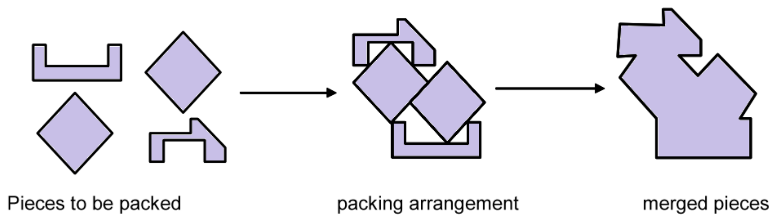


Figure 2.7: Pieces merge in TOPOS and missing the enclosed gaps (extracted from Bennell and Song (2010))

Bennell and Song (2010) suggested a revised approach for TOPOS. Instead of merging pieces, at every insertion, Bennell and Song (2010) proposed to merge the NFPs between candidate piece and the pieces already in the partial layout. These NFPs are used to find the feasible positions for the new piece. The method allowed automatic generation of inner-fit polygons from the enclosed gaps and accordingly established hole-filling.

2.5.1.2 Placement sequences:

The placement sequence of pieces describes the sequence of positioning each piece in the packing area. Similar to the placement rule, the placement sequence of irregular pieces also makes a significant impact to the final complete layout. Bohme and Graham (1979) used a constructive method by throwing pieces on to the stock sheet randomly. They used at least 2000 random orders of pieces to obtain a reasonably acceptable solution. In order to find the best solution, they believed on improvement given by small random perturbations. These *random sequences of pieces* produced highly variable solutions and the issue of this approach was to consider a higher number of multiple runs to find a better solution.

Qu and Sanders (1987) proposed two methods to construct solutions. The first method sorts pieces by *decreasing length* and then places them along two adjacent edges of the stock sheet, whereas the second method sorts pieces by *decreasing heights*. The second method was later improved by the authors to have an advantage of possible interlocks between the pieces. In this case, the tallest piece is selected first as the starting piece. The other pieces are then *dynamically sorted* so that the best fitting piece can be found in subsequence placements. In order to calculate the fit between the partial solution and each of the remaining pieces, the area ratio between the piece and the free space remaining from along the partial solution to the right border of the piece was calculated. Dagli and Tatoglu (1987) conducted an experimental analysis of different initial ordering of pieces such as *sorted by area*, *irregularity* and *number of of sides*. Oliveira et al. (2000) described ordering pieces using the following criteria; *decreasing length*, *decreasing area*, *decreasing concavity*, *increasing rectangularity and total area (quantity$\times$area)*. As similar approaches, the authors also test the piece orders; *decreasing width*, *decreasing perimeter*, *decreasing aspect ratio*, and *decreasing area utilization*, in their study. Dowsland et al. (2002) used eight static orderings; *Decreasing area of enclosing rectangle*, *Decreasing length of enclosing rectangle*,*Decreasing width of enclosing rectangle*, *Decreasing perimeter of enclosing rectangle*, *Decreasing aspect ratio of enclosing rectangle*, *Decreasing area of polygon*, *Decreasing perimeter of enclosing polygon* and *Decreasing utilisation ratio of enclosing rectangle*. The strategy of each placement sequence was designed to place the *difficult-to-place pieces first*. All these methods discussed in this paragraph represent the placement sequences based on a *fixed sequence*. In comparison to the *random selection*, the heuristics are slightly expensive in terms computation to generate a single solution. However, there are some rules, making reasonably promising results (e.g. sorted by area) for some sets of instances (Oliveira et al., 2000).

Albano and Sapuppo (1980) used a construction method using a placement rule and found that focusing only a fixed ordering leads to an inefficient result. As a solution, they proposed a search tree where each node represents the partial solution at each stage of solution construction. The approach permits all piece types to be selected as the next piece to be placed. Bennell and Oliveira (2009) denote this as *dynamic selection approach*, where a greedy selection mechanism decides which piece to select for placement at the current state (i.e search node), based on the characteristics of the previous state. Using this principle in, Albano and Sapuppo (1980), Bennell and Song (2010) have modelled the progression of partial solution generation as a search tree. The lowest level of the tree represented the final solution. In some cases, authors pruned some branches of the search tree depending on the evaluation results of each partial solution. In comparison with other rules of ordering, the dynamic selection investigates several combinations of order sequence which causes higher computation efforts. However, exploring several combinations in a search tree leads to finding an improved solution if the criterion for evaluating partial solutions is effective and directs the search towards finding a better complete solution when it reaches to the end of the search tree.

According to Bennell and Oliveira (2009); Gomes (2013), working with partial solutions is one of the two main computational approaches for solving irregular shaped C&P problem. Generally, the constructive procedures are developed to find a feasible packing layout by placing pieces one after another. The other approach works with all the pieces as a whole where all pieces are placed in the layout and tries to derive the final solution by moving, swapping, or rotating the pieces with a focus of improving the complete solution. According to Bennell and Oliveira (2009), the constructive procedures are capable of generating feasible better solutions if it is exposed to an efficient strategy for finding a better order of pieces and placement strategy. In order to find high quality solution, it is necessary to work through multiple solutions. According to Bennell and Oliveira (2009), researchers achieved good solutions using constructive methods through the approaches like beam search based on a dynamic selection of pieces (Bennell and Song, 2010) or by iteratively improving the complete solutions based on sequence of pieces. The constructive heuristics typically support to construct feasible initial solutions and requires an improvement stage which implement a search mechanism to search over the solution space to find a good solution.

2.5.2 Work with complete solutions

This section reviews how a search is conducted to find a better solution by making changes iteratively to the complete solution. Usually these approaches use metaheuristics to search for better solutions. E.g. Simulated Annealing (SA) by Heckmann and Lengauer (1995) and Oliveira and Ferreira (1993), Tabu Search (TS) by Bennell and Dowsland (1999) and Burke et al. (2006), Genetic Algorithm (GA) by Jakobs (1996) and Babu and Babu (2001). The articles Dowsland and Dowsland (1995) and Bennell and Oliveira (2009) present useful surveys and categorization of using search methods have been applied to irregular shape packing problems. Bennell and Oliveira (2009) reviewed two categories of search algorithms for irregular shape packing problems. While a certain set of approaches is working directly with the complete solution by moving the pieces around in the physical layout, another set of approaches is searching over a sequence of pieces and explore different solutions relying on a certain placement rule.

2.5.2.1 Searching over with feasible solution

These search methods are working similar to the search methods applied for sequencing problems in operational research. In sequence searching methods, researchers apply neighbourhood search strategies such as *swap moves* (e.g. Gomes and Oliveira (2002)), *insert moves* (e.g. Takahara et al. (2003)) and *change orientation move* (e.g. Jakobs (1996)) to the insertion sequence of pieces. We briefly describe the methodology of each of these metaheuristics and their features in Chapter 3.

The methods work with solutions which are decoded into sequences of pieces. The solutions are always feasible and the requirement is to search for a sequence of pieces to have a better layout using a given placement rule. Figure 2.8 illustrates how each of these move performs to change the permutation of the pieces.

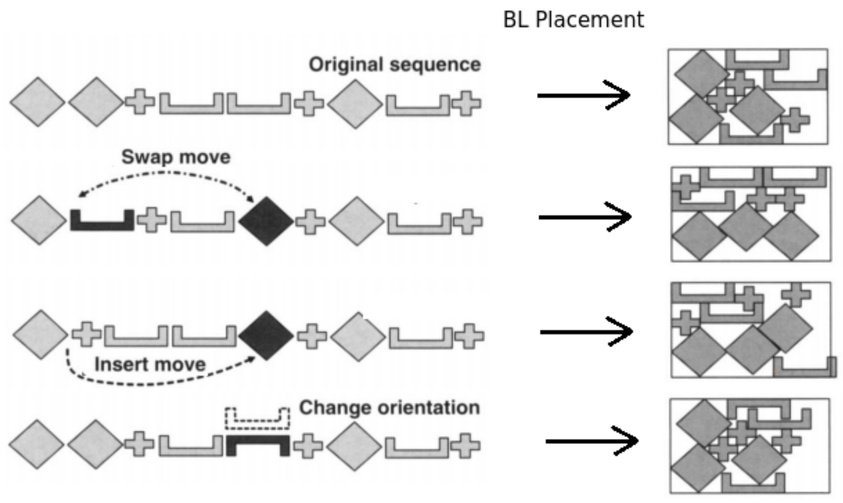


Figure 2.8: Piece moves (extracted from Bennell and Oliveira (2009))

Jakobs (1996) used GA to search over the piece orders. In this approach the initial offsprings are generated through a randomly ordered sequence of pieces and progressed the search through crossover and mutation operators. For mutation, they introduced an orientation-move by rotating the pieces in 90 degrees. Babu and Babu (2001) dealt with packing irregular pieces in multiple heterogeneous stock sheets using GA. Their chromosome coding included three parts; sequence of stock sheet types, sequence of pieces and sequence of orientation of the pieces. The crossover operator of this algorithm randomly selects two crossover points, one from the sequence of stock sheets and the other from the sequence of pieces and changes the sequences by swapping the genes of two chromosomes. They used mutation operators to implement the swap moves and orientation moves.

Dowsland et al. (1998) proposed the Jostle algorithm, which packs the pieces to the left end of the stock-sheet and then to the right end and vice versa. The placement order of the pieces is determined by the x-coordinate of each piece in the previous iteration. The heuristic starts with an arbitrary or defined ordering of the pieces and packs pieces from left to right following the leftmost placement policy. Next, the pieces are re-ordered in the decreasing order of the x-coordinates of their rightmost points and packed following the rightmost placement policy (see Figure 2.9). Then the pieces are re-ordered in the increasing order of their leftmost points and pack again using the leftmost policy. This will continue for a fixed number of iterations, allowing pieces to oscillate from left to right and right to left.

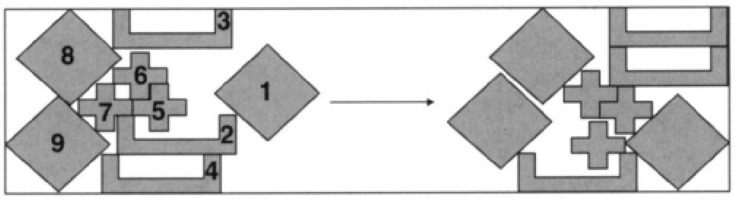


Figure 2.9: One step of Jostle Algorithm (extracted from Bennell and Oliveira (2009))

Takahara et al. (2003) proposed a neighbourhood search based only on insert moves. The pieces are moved according to a probability which is dependent on their relative weight. The probability of move is increased by increasing the weight by one unit, if it makes any improvement to the solution.

Burke et al. (2006) applied the hill climbing local search method and TS to find reasonably better solutions. In this study, the neighbourhood size was limited to randomly selected five solutions by applying a random moves called; insert, pairwise swap, three-way swap and n-way swap.

2.5.2.2 Search over the layouts allowing infeasible solutions

According to Bennell and Oliveira (2009), allowing infeasible solutions during the search space is one of the main characteristics of this category. Accepting solutions with overlaps and penalizing the degree of overlap in the objective function open ups the search space. However, this form of search, sometimes struggles to reach a local optimum with feasible placement of pieces. Bennell and Oliveira (2009) also highlights that the issues arising in designing a mechanism is mainly due to the size of the neighbourhood, which is infinite since placement and orientation of the pieces can be infinite in a stock sheet which is defined on a continuous scale.

Many local search approaches have been used to improve the physical layout. SA and TS are two initial local search techniques used in solving irregular shape packing problems. In these search methods, the search process starts with a starting solution and explores one or more neighbour solution as a replacement for the current solution. As described in Dréo (2006), the entire search method focuses on the solution space, neighbourhood structure and the cost function.

Oliveira and Ferreira (1993) used SA, which places pieces randomly on the packing area and performs neighbourhood movement one piece at a time. They move a piece in the grid layout by one grid unit at each move. As states in (Bennell and Dowsland, 2001), SA was most favoured in the studies of the 90s than TS since the fundamental approach of TS does not favour the infinite neighbourhoods in a continuous stock sheet. However, Błażewicz et al. (1993) used TS in a discretized packing area which allows the conversion infinite piece placement positions into a finite and discrete set.

Egeblad et al. (2007) developed an efficient search method which translates pieces in a specified direction and finds a promising position for a piece that minimizes its overlap area. They used Guided Local Search to escape local minima.

Bennell and Oliveira (2009) review two major frameworks to change the solution and define the neighbourhood of a solution. They are; 1) moving a single piece or a set of pieces either using inset-piece move, swap move, changing orientations or horizontal translation, vertical translation, or flipping of pieces and 2) Use compaction and separation procedure, to change positions of pieces simultaneously. Bennell and Dowsland (2001) first introduced compaction using linear programming. Gomes and Oliveira (2006) used a hybrid methodology of Simulated Annealing (SA) and linear programming. Once an initial layout is constructed using the bottom-left placement rule, SA was used to guide the search over the solution space. The linear programming was used to apply the compaction algorithm at each neighbourhood structure. The objective of compaction was to slide the pieces in a given layout to achieve better layout. A separation algorithm was used to remove the overlapping while moving the pieces.

Imamichi et al. (2009) proposed a new separation algorithm based on unconstrained Non-linear programming. They also designed an algorithm which swaps two pieces placed in the layout and this was combined into an iterated local search algorithm to minimize the overlap by those swap moves. Leung et al. (2012) proposed an extended local search algorithm based on Non-linear programming. In this case, two neighbourhoods were used to change piece positions during the local search where as the feasibility of the solution is determined by a set of non-linear programming separation and compaction models. Elkeran (2013) proposed a methodology which uses a heuristic to cluster the pieces in pairs and then use guided cuckoo search to pack the pieces into the packing area. This study achieved the best results in the literature for the irregular strip packing problem.

According to Bennell and Oliveira (2009), searching with feasible solutions is computationally more expensive than searching for a piece move within the layout since the sequence change requires generating the whole solution to evaluate each change. The quality of the solution depends on the placement rule used in the constructive algorithm. The approach guarantees evaluation of feasible solutions at every time it generates a complete solution. In contrast, the moving pieces within layout can create overlaps among the pieces while searching for better positions. In some situations, it takes longer to find a feasible solution.

2.5.3 Exact methods for irregular shape packing problems

Though heuristic and metaheuristic algorithms are popular among the majority of the studies and recorded the best performance (e.g Imamichi et al. (2009), Bennell and Song (2010), Elkeran (2013), Lopez-Camacho et al. (2013a)), they are not able find optimal solution or determine how far the resulting solution is in the distance to the optimal. This research gap is trying to be filled by a few researchers using exact solution approaches which are based mathematical programming models.

As an initial approach, Ribeiro et al. (1999) developed an exact method to solve the strip packing problem with convex pieces based on constraint programming. The constraints of the mathematical programming model ensured the overlapping and containment constraints. They defined the allowable placement points for each piece by imposing non-overlap constraints for each pair of pieces. As another initial work, Carravilla et al. (2003) used the exact computational approach for solving nesting problem with non-convex pieces and was able to solve the problem to optimality with maximum of 7 pieces. In both studies, the pieces reference point were associated with a discretized placement within the stock sheet.

Fischetti and Luzzi (2008) developed a mixed integer programming model and a branching strategy which is used with CPLEX. The non-overlapping constraints of this model uses tighter coefficients instead of big-M constants (Martinez-Sykora, 2013). Kallrath (2008) introduced an exact nesting algorithm for convex polygons using the idea of separating hyperplane. The convex polygons are considered pairwise and a hyperplane that separates the two polygons is searched to satisfy the non-overlap constraint. The approach can also handle non-convex polygons by decomposing them into convex polygons. The main issue with this approach is the difficulty of finding optimal solutions for the problems with more than two polygons, especially with non-convex polygons.

For strip packing problems, Toledo et al. (2013) proposed the Dotted-Board Model where the placement points on the stock sheet (they referred this as a board) are allowed in the integer positions. This approach was efficient and able to solve to the optimality for 21 pieces in 7 different instances. In some cases it solved to the optimality for 56 pieces only with two different instances. In a bin packing problem, Baldacci et al. (2014) discretized both bins and the pieces into a bit-matrix, and imposed constraints to cover each position of the matrix. Due to this reason, the exact model does not allow placement position in a continuous scale. The issue of this model is the accuracy which is highly determined by the resolution of discretization. However, the discretized methods simplify the formulation and solution procedure.

The studies related to continuous positioning models using mixed-integer programming have used no-fit polygons (NFPs) so that the outside area of the NFPs is used to set up the non overlapping constraints of polygons relative to each other (Gomes and Oliveira, 2006; Alvarez-Valdes et al., 2013). When compared to the discretized methods, the exacts models based on continuous placement are able to solve to optimality for a smaller number of piece types (Cherri et al., 2016). As a better performance, Alvarez-Valdes et al. (2013)'s continuous exact approach, the model solves to optimality instances up to 16 pieces within 5 hours of computation time. Alvarez-Valdes et al. (2013) developed an exact Branch & Bound Algorithm and explored different branching strategies. Cherri et al. (2016) produced mathematical programming models based on two directions. The first direction they used direct trigonometry to derive non-overlapping constraints without using NFPs. The second direction decomposes the irregular pieces into convex shapes and generated NFPs for the convex shapes. The intension of these approaches was to reduce the complexity of the exact model by reducing the complexity of of geometry components attached to it. The models addressed irregular strip packing problems with polygonal shape geometries without any approximation. The major drawback of this approach was the size of the instances possible to solve to the optimality. The approach NFP-CM discussed in Cherri et al. (2016) finds an optimal solution for most of the instances with up to 12 pieces whereas the method cannot find a feasible solution with more than 30 pieces. In summary, the exact method approaches developed so far haven't been able to deal with the large instances (>30 pieces).

2.6 Computational Method for Irregular shape Bin Packing Problems

2.6.1 Constructive methods

In this section, the review scope is limited only for irregular shape bin packing problems and discusses the constructive methods applied for irregular bin packing problems. Only a limited number of publications consider the bin packing problem with irregular pieces where most of them are only limited to pack irregular shapes in single size bins.

Therefore, the computational methods developed for irregular bin packing problems are not plenty in the literature.

In general, the existing studies focus on two decisions; 1) selecting both the next piece and the bin, and 2) finding the exact placement locating of the piece within the selected bin, specific to the 2D IBPP (Lopez-Camacho et al., 2013a, 2014). Lopez-Camacho et al. (2014) claimed that for non-trivial irregular bin packing problems the exhaustive search is impractical, and highlighted the necessity of heuristic methods.

Terashima-Marín et al. (2010) and Lopez-Camacho et al. (2013a) discussed following bin selection heuristics; First Fit Decreasing (FFD), Best fit Decreasing (BFD), Filler (FL) and Djang and Finch (DJD) for the first decision. According to Lopez-Camacho et al. (2013a), the above heuristics generated reasonably better solutions compared to the other heuristics methods investigated. In this study, Lopez-Camacho et al. (2013a) investigated 11 selection heuristics, including a few of worst performing heuristics such as Worst fit (WF), First fit Increasing (FFI). In general, the first three heuristics; FFD, BFD and FL are used for the cases of 1D and 2D bin packing problems in the literature. In all three heuristics, the pieces to be packed are arranged in an a certain sequence and one piece is packed at a time.

The FF heuristic opens a bin if the piece cannot fit in any of the opened bins. As a subversion, the *First-Fit Decreasing* (FFD) heuristic considers the pieces sorted in decreasing order of the area and places each piece inside the first bin where it fits (Lopez-Camacho et al., 2013b, 2014). The NF heuristic always selects the current bin to place the next piece. If the new piece fits in the current bin, it is placed there, otherwise open a new bin and place the piece in the new bin (Lopez-Camacho et al., 2013b, 2014). The BF heuristic tries to place the piece in the opened bins where it best fits. A new bin is opened only when there is no space to allocate the next piece in the opened bins. A subversion of this called Best Fit Decreasing (BFD) sorts the unplaced pieces by decreasing area and places each piece in the opened bins where it fits best (Lopez-Camacho et al., 2013b, 2014).

The key idea of Filler (FL) is to pack pieces as much as possible for an opened bin without considering the order of the bins. The heuristic starts with sorting the pieces in order of decreasing area and then packs as many pieces as possible into the opened bin. In a situation where no single piece can be placed in the open bin, a new bin is opened to pack the remaining unpacked pieces from largest to smallest (Lopez-Camacho et al., 2013b, 2014).

Lopez-Camacho et al. (2013a) denoted Djang and Finch(DJD) as a complete single-pass constructive heuristic for 1D problems. However, for 2D problems it is only a bin selection heuristic (Lopez-Camacho et al., 2013a). The heuristic works in one open bin at a time and does not consider the other open bins. Pieces should have an order which is usually decreasing area. The heuristic works with an assigned value of fullness, usually taken as 1/3, 1/4, 1/2, for example, if 1/3 of fullness is allowed (i.e. then the algorithm is called as DJD 1/3), then the heuristic fills the bin with the sorted pieces until one-third of it becomes full. As the next step, it initializes a variable $waste = 0$. The heuristic then attempts to find either one, two, or three pieces that completely fill the bin leaving a free area up to *waste*. If there is no combination matches with the requirement and *waste* is less than the bin free area, then the value of *waste* is increased a certain amount (usually a fraction of the bin capacity), otherwise a new bin is opened. Lopez-Camacho et al. (2013a) empirically found that, increasing this waste as one-twentieth of the total bin area, provides a good balance to generate efficient solutions. Lopez-Camacho et al. (2013a)'s experiments demonstrate that the DJD 1/3 and DJD 1/4 produce better results than the other selection heuristics. However, DJD heuristics are time consuming since it tries several combinations before placing the 3 pieces.

Lopez-Camacho et al. (2013a) considered four placement heuristics to work with the selection heuristics. They considered *bottom left* and three versions of *constructive approach* placement rule which is originally presented by Hifi and M'Hallah (2003). They used BL heuristic in a way that piece starts moving the right hand corner of the bin and slides down and left with a set of movements until no further movement is possible. The piece does not overlap with the bin edges and it is not possible to pass over the already placed pieces while sliding the candidate piece. The *constructive approach* heuristic starts by placing the first piece at the bottom-left of bin. Each candidate piece starts its placement from each of the nine placement positions. These nine points include the four corners of the bin and coordinate positions; $(0, \bar{y})$, $(\underline{x}, \bar{y})$, $(\bar{x}, \bar{y})$, $(\bar{x}, \underline{y})$, $(\bar{x}, 0)$ where $\bar{x}$, $\underline{x}$, $\bar{y}$, and $\underline{y}$ are the maximum and minimum coordinates in x and y of the previously placed piece as illustrated in Figure 2.10.

From each of these starting points, the feasible movement positions are recorded when the next piece slides vertically and horizontally following down and left movements as illustrated in Figure 2.10. Out of the feasible points reordered, the most bottom-left position is selected as the placement point for the next piece.

Lopez-Camacho et al. (2013a) modified the CA rule to select the best placement position based on the position that yields the minimum bounding rectangle (the minimum area) which contains all placed pieces and the new piece. This was called as CA with Minimum Area (CAA). Another CA version was tested in the same study where the placement position is evaluated based on the largest adjacency. The adjacency describes the common boundary between piece perimeter and the placed pieces and the bin edges (Terashima-Marín et al., 2010). As described in CA rule, the candidate piece is sliding down and left, to place in the position where the largest adjacency is found. Lopez-Camacho et al. (2013a) denoted this heuristic as CA with maximum Adjacency.(CAD).

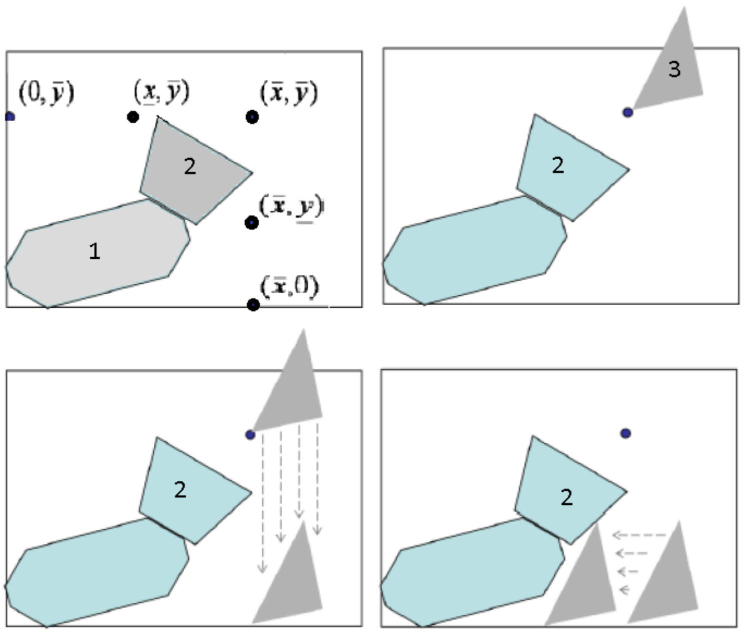


Figure 2.10: Constructive approach (Terashima-Marín et al., 2010)

When placing the pieces, orientation of the piece (angle of rotation) impacts significantly for the final placement. Terashima-Marín et al. (2010) tested two schemes of rotating pieces; the first one rotates each piece by multiples of 90 degrees (0, 90, 180 and 270 degrees) and the second rotates each piece in multiples of 5 degrees. They found that there is no significant different in both approaches. We believe this is mainly due to the jigsaw type pieces in which the optimal solution can only be found with the multiples of 90 degrees (i.e. 0, 90, 180, 270) rotation angles.

The placement rules suggested in Lopez-Camacho et al. (2013a) is time consuming due to their sliding operations which performs several times till the piece reaches its final placement position. In addition, CA, CAA and CAD heuristics are more time consuming than BL due to the exploration of the movements start from several starting points as described in the CA placement rule. The other issue of Lopez-Camacho et al. (2013a)'s constructive procedures is they are only functioning with a predefined discrete set of angles. The study mostly used jigsaw type pieces to test each of the constructive methods. Usually such pieces are aligned to a better solution with a selected set angle. However, in practical situations, most pieces are not jigsaw type and requires to consider a higher number of orientation angles efficiently by the constructive method use.

Han et al. (2013) proposed two constructive heuristics for bin packing problems with irregular pieces. The problem only considered convex pieces arises from glass industry where the cutting process forces use of guillotine cuts. Han et al. (2013) discussed a one-stage solution approach as a constructive heuristic which begins with generating efficient clusters of polygons. These clusters were used to construct a complete solution.

A part of this clustering process possesses the matching of two polygons. This includes placing each polygon relative to the other polygon based on the objective of minimizing the waste. When two polygons are placed (touching with each other) relative to each other, Han et al. (2013) calculated the utilization gained by the pieces over the area of the convex hull of these two polygons, and the utilization gained by the pieces over the area of the rectangle enclosure of those (see Figure 2.11). In order to evaluate the best matching position, these two measurements were used following a weighted sum approach. Always pieces are touched by its edges and along the edges it finds the best relative placement position.

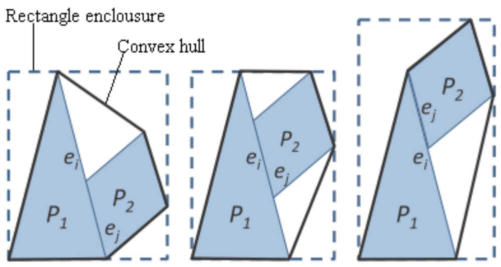


Figure 2.11: Slide along the edges (extracted from Han et al. (2013))

Han et al. (2013) defined a search forest to represent the constructive stages of the solution. A node denotes the matching of a set of polygons if the utilization ratio of those polygons achieves or exceeds an acceptance threshold. The forest is continued till there are no further matches for the blocks. Han et al. (2013) defined blocks as a set of polygons grouped together by a feasible transformation. Figure 2.12 illustrates a forest made with composition of blocks through the levels of the search. The top level contains all pieces individually. The second level contains pairs of pieces match to a higher utilisation. The last level (level six in this case) contains all six pieces transformed to a block.

The next step of this approach is to pack the blocks into the bins. In this step Han et al. (2013) placed those blocks in bins efficiently. They defined a utilization threshold to check whether a bin packing pattern is acceptable. Han et al. (2013) also proposed a two-stage algorithm that groups pairs of pieces into rectangles and then packs those rectangles using the guillotine bin packing algorithm proposed by Charalambous and Fleszar (2011). The approach works well for convex pieces and presents quality results within a reasonable time. However, this approach includes guillotine constraints and edge matching which addresses significantly a different problem than the problems considered in this thesis. Also, the proposed constructive algorithms are only capable of dealing with convex pieces.

As a similar problem, Martinez-Sykora et al. (2015) presented a constructive algorithm to build irregular shape bin packing solutions with guillotine constrains by adding pieces sequentially to the current bin. They use a mixed integer programming (MIP) model

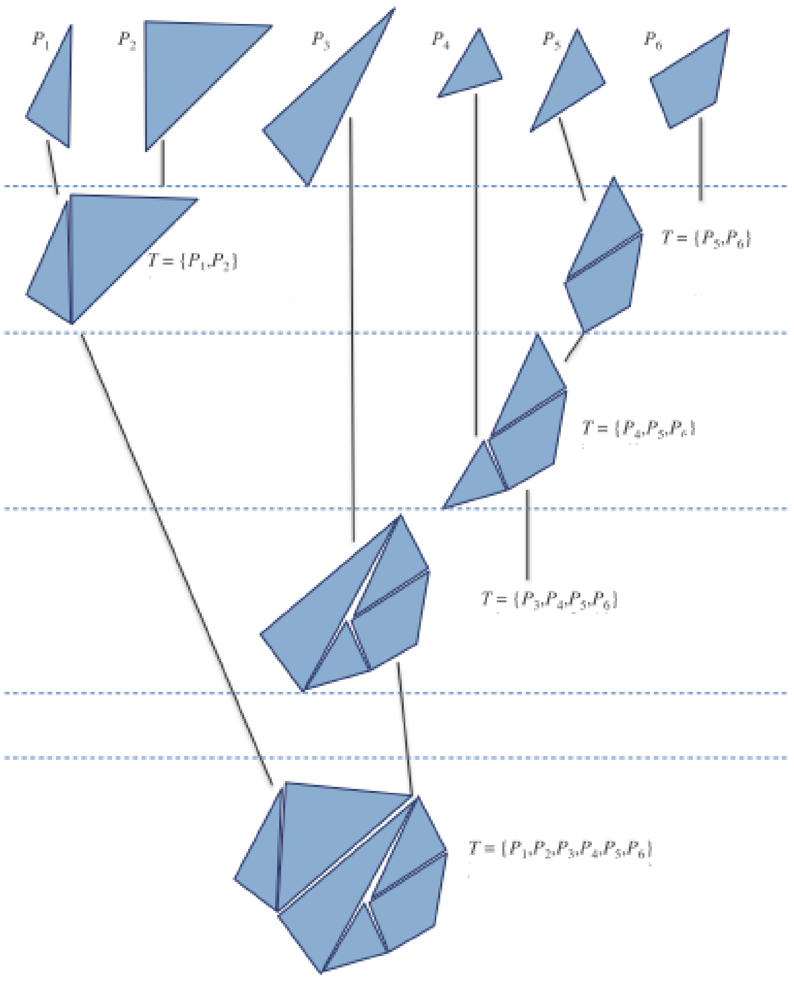


Figure 2.12: Search forest (extracted from Han et al. (2013))

to determine the position of the pieces in the bin and the corresponding guillotine constraints. In their constructive method, the relative position of the piece with respect to the other pieces is fixed, however the absolute position of each piece can be changed at each insertion of pieces when the constructive algorithm is being run. The constructive algorithm uses the set of angles of rotation and reflection for each piece sequentially place to the bin. For a given initial permutation of the pieces, within an open bin, the candidate piece generates the rotation angles using an algorithm called GR which produces the rotation angles of the piece by matching the edges of the piece with the edges of the other pieces. For each rotation the authors solve the MIP model and try to find the best feasible placement position. If such a feasible solution is found, the algorithm adds the candidate piece into the bin and updates the orientation angle; otherwise they leave the piece unpacked, and move on to the next and try to place it in the bin. When no other piece can feasibly fit into the bin, the algorithm closes that bin, opens a new bin and start packing from the first unpacked piece in the list. After packing all pieces, the authors evaluate the occupied bin with the lowest utilization using two evaluation functions and rebuilt the bin with the aim of packing the pieces as tightly as possible. Martinez-Sykora et al. (2015) denotes these two objective functions as FO1 and FO2, where FO1 is used to minimize the width when there is tie in length used and the FO2 used is to minimize the length, when there is tie in width used. At the last step of the constructive algorithm, the rebuilding of the lowest utilized bin involves applying a horizontal or vertical cut respectively to differentiate the occupied area of the bin to find the fractional number of bins. At each insertion of a piece, Martinez-Sykora et al. (2015)'s method calls the MIP model several times. This requires a significant computational effort. In Martinez-Sykora et al. (2015)'s approach, computational time to run the constructive algorithm highly depends on the number of pieces in a problem instance.

The same study proposed another constructive algorithm having a change to the previously discussed constructive algorithm which identifies the guillotine cuts after each piece insertion and then sets as constraints to the MIP. In the previous constructive method, the next piece is forced to satisfy all the guillotine cuts previously defined before finding the best insertion. Martinez-Sykora et al. (2015) proposed that this can over constrain the solutions space. Therefore, they propose a change to their first method so that the next piece is inserted ignoring the guillotine cut constraints. After finding the best insertion, they set the algorithm to perform a procedure to identify a new guillotine cut structure, if one exists. They denoted this procedure as two phase constructive algorithm. According to this new modification, they also changed the constraints in their MIP model to work with this second constructive procedure as well. Martinez-Sykora et al. (2015) experimented three alternatives sorting criteria for the pieces; random, Non-increasing area and by shape.

Martinez-Sykora et al. (2017) described a constructive algorithm considering the two aspects of the irregular bin packing problem including free rotation of pieces. As the first aspect, the constructive method considers the assignment of pieces to bins where as the second aspect considers the arrangement of assigned pieces in the bin. Martinez-Sykora et al. (2017) implemented five different strategies for assigning pieces to the bins; 1) Greedy Decisions, 2) First Fit Algorithm, 3) Partial Bin Packing, 4) Two Phases Strategy and 5) a simple construction heuristic. In the constructive algorithm, one of these was implemented as the first phase.

The first method greedy decisions (GD) was based on the idea of retaining the bin that has a feasible arrangement with some of the assigned pieces. Then the focus is to add pieces from the other bins without violating the feasibility and improve the bin utilization. This strategy assigns pieces to bins by solving an IP model used in a 1D bin packing problem. Since it solves an IP formulation, the first strategy requires a considerable time to generate a complete solution. The second strategy uses the popular first fit heuristic (FF). According to Martinez-Sykora et al. (2017), this algorithm is very fast and the quality of the solutions depends on the initial ordering of the pieces. The third strategy Partial Bin Packing (PBP) uses an objective function that favours assigning larger pieces. This assignment strategy follows a knapsack formulation as in 1D BPP, so that the objective is to maximize the value of bin by packing some of the available pieces. If *packing procedure* (This is separately explained in the next paragraph briefly, more details of it are available in Martinez-Sykora et al. (2017)) fails to place a given piece j feasibly, then they solve the knapsack formulation again for the remaining capacity by removing j from the consideration. Therefore, with the new assignment, the *packing procedure* attempts to place the newly assigned piece feasibly from the available pieces. These tasks; *feasible packing* (performed by the packing procedure) and *reassignment*, continues until it finds a bin with feasible placements of pieces. Any piece(s) j which are removed during the process are returned back to the available set of pieces to pack. As the next step, the algorithm then solves the knapsack formulation to generate a new bin. The fourth strategy is a two phase one, which executes the GD first. In this case, after solving the assignment model of the GD, it provides the minimum number of bins. Martinez-Sykora et al. (2017) solve another IP model to reassign pieces across bins. This model forces the assignment of large pieces into the earlier bins so that it can minimize the total number of bins. The fifth strategy was called simple construction heuristic which packs pieces into a bin following a given sorted order of piece. The method uses Next-Fit Decreasing (NFD) strategy, in which the next piece i is packed feasibly into the opened bin by solving the IP model described in Martinez-Sykora et al. (2017). In this case, the algorithm considered packing the next piece $(i+1)$ to the current bin. If piece i cannot be placed feasibly, close that bin and open a new bin.

In the second phase, the subset of pieces assigned to each bin is sequentially placed inside the bin using the *packing procedure*. This includes determining a set of promising orientations for the piece and for each orientation angle determining whether the piece fits into the bin according to the MIP model. The *packing procedure* allowed all pieces to move within the bin when placing the next piece. When assigned pieces are failing to pack feasibly, *reassignment* takes place as explained in the previous paragraph. Comparing each version of the constructive algorithm, out of these five options, Martinez-Sykora et al. (2017) found the best results for the PBP strategy. The *packing procedure* described in this article allows placement of pieces with free rotation. This is also the first study to deal with free rotation of pieces for both convex and non convex pieces in irregular bin packing. Since the problem definitions are the same in this article and the first problem addressed in this thesis, we use the results of this study to compare the performance of our new computational method. As the key improvement areas we focus on improving the computational time, since the *packing procedure* has taken considerable computational effort to solve the MIP due to the higher number of constraints attached to it.

2.6.2 Search methods for irregular bin packing problems

Terashima-Marín et al. (2010) presented a GA-based method to produce general hyper-heuristics for the irregular shape bin-packing problem. In the hyper-heuristics approach, they attempted to explore the efficient combination of heuristics to apply for a given situation of the problem. Hyper-heuristics design generic methods to produce solutions with acceptable quality(Burke et al., 2013). The performances are based on the set of simple low-level heuristics where as the hyper-heuristic acts as a high-level guidance methodology (Burke et al., 2013). After evaluating the individual performance of each heuristic, Terashima-Marín et al. (2010) used GA to evolve combinations of rules to produce the suitable hyper-heuristic. Terashima-Marín et al. (2010) observed a slightly better performance when the hyper-heuristics are tested with more irregular instances. They also found that, when compared to the 12 hours in solving 541 instances with 40 single heuristics, the hyper-heuristic approach takes 11 hours to run the GA to generate a hyper-heuristic. Once a suitable hyper-heuristic is found, solving each instance in the set with the produced hyper-heuristic taken only a few minutes. Accordingly, Terashima-Marín et al. (2010) believed that hyper-heuristic approach is faster than solving the instances with each of the 40 single heuristics. However, one issue with this approach is the performance of low level heuristics when it investigates a higher number of rotation angles. All 40 instances investigated by Terashima-Marín et al. (2010) worked with a finite set of angles (< 4 in most cases). In Terashima-Marín et al. (2010)'s study the performance of the hyper heuristic approach is based on the range of low level heuristics and the selection mechanism of heuristics. However, as reviewed above, the low level heuristics of this particular study shows its efficiency only with a finite set of rotation angles. In addition, no constructive method is described to consider unrestricted rotation of pieces and only provision there is to consider the higher number of finite rotation angles which cause higher computational time ultimately.

Martinez-Sykora et al. (2015) discussed the solution method for irregular bin packing problem considering guillotine placement of convex pieces. In their solution method, they developed an improved procedure embedded into the construction algorithm so that utilization of a bin can be improved before that bin is closed. Then the improvement phase is executed within the constructive algorithm before opening a new bin. The authors applied this to the bins only with utilization below a given threshold. According to this method, the improvement procedure removes the piece with the worst utilization and rebuilds the bin without changing the placement and orientation of the other pieces in the bin. In this case, the utilization is calculated as the ratio between the area of the piece to the area of the containing polygon. As illustrated in Figure 2.13, the guillotine cuts define the containment polygon around each piece. As the next step the algorithm tries to insert pieces in the unpacked list. In this phase, the removed piece is also considered at least to insert to the same bin if it is possible to place considering ten

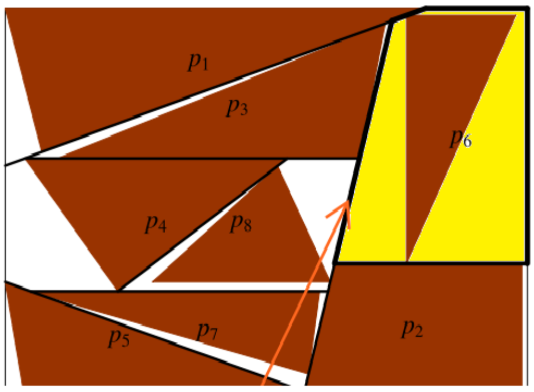


Figure 2.13: Containment Polygon (extracted from Martinez-Sykora et al. (2015))

alternative rotations at each reflection option. As the acceptance criteria, the bin solution is accepted and identify the next piece with the lowest utilization if the utilization of the repacked bin is higher, otherwise the procedure continues until no improvement can be found.

Martinez-Sykora et al. (2017) presented two local search strategies based on hill climbing first found improvement. The search method was simple, did not include any diversification scheme and terminated when a local optimum solution is found. They considered the coefficient F proposed by Lopez-Camacho et al. (2013a) to evaluate solutions. If a move makes any reduction in the number of occupied bins, as a result F increased and such move was accepted during the search process. Also, if a move makes any improvement in utilization in one bin maintaining the same number of bins, F was increased, and accepted the move. During the search process, both properties are important to find good solutions as it differentiates solutions with the same number of bins.

Out of two search methods, the first method starts with a solution of occupied bins sorted by non-decreasing utilization. Then the local search procedure considers all pairs of bins, say i, j and tries to move pieces one after the other, from one bin (i) into another (j). The move accepted if all pieces from both bins fit into only one bin or if utilization of j is increased and utilisation of i is decreased. The objective of this search method is to reduce the usage of the least utilised bins with the aim of emptying those bins. The second method proposed by Martinez-Sykora et al. (2017) was an extended version of the first search method, In this case, in order to empty bin i, a set occupied bins were proposed instead using a single bin j. This increases the chances of packing all pieces of bin i into a set of bins. As the set of j bins, the bins with higher utilisation than bin i and lower than 0.99, are selected for new placement. In the first local search, Martinez-Sykora et al. (2017) rebuild bin i when a new bin j is built. However, in the second search method, they consider more than one bin of js before rebuilding the bin i. A fail to pack all the pieces from bin i, or in other words, a fail to reduce at least one bin from the previous solution, fails the whole movement and replaces to the previous solution. According to Martinez-Sykora et al. (2017) both local search procedures obtained significant improvement and the second LS2 performed better than first, even though it consumed almost similar computational time.

Both Martinez-Sykora et al. (2015) and Martinez-Sykora et al. (2017) approaches present the best result in each of their contexts. The approaches established high accuracy due to the use of the geometric representation and they allow free rotation of the pieces which is rare in the literature. One main issue arises is the high computational time they take to find a reasonably good solution. Reviewing all the solution approaches for irregular shape bin packing problems, we find that it is strongly significant to reduce the computational time of the constructive algorithm especially when pieces orientation is unrestricted. Meanwhile, improvements in search method are also critical and require experimenting different search mechanisms to improve the solutions.

2.7 Summary

Wäscher et al. (2007) and Wäscher (2012) provide substantial information about the entire scope of cutting and packing problems and their variants in existing literature. Within this context, our review of literature mainly concentrated on irregular shape strip packing and irregular shape bin packing problems. Even though the objective of the thesis is on irregular bin packing problems, we reviewed the solution methods used for irregular strip packing problem since some of the concepts are useful in developing solution methods for the irregular bin packing problems.

Out of the different solution approaches for irregular shape packing problems, the literature contains two main categories of solution approaches; exact methods and heuristic based approaches. Within the scope of this thesis, we intend to develop algorithms for the industries running mass scale production, which is dealing with a higher number of pieces and a higher number of stock sheets. Out of two main computational approaches; heuristic based approaches are favoured more as they can provide good solutions within reasonable computation time even for large problems. On the other hand, the use of exact method demonstrated its power in finding the global optimum solution only for a limited number of pieces.

Reviewing heuristic based approaches, we noted that there are two main approaches for dealing with the problem. One approach always works with the feasible solutions and attempts to find better solution using a search method. The other approach accepts overlaps (i.e. infeasible solutions) and continues searching over the layout until it reaches to feasible, good solution. The chapter reviews the advantages and disadvantages of both approaches based on the outcomes of different studies.

We also review the literature specific to the IBPP and note that the computational time for solution construction is a significant issue when the packing orientations of shapes are not restricted. Even-through the number of studies related to this area is limited, so far, a few researchers attempted to solve this problem using metaheuristics and math-heuristics approaches.

Chapter 3

Methodology

3.1 Introduction

This chapter presents a discussion on the key methodologies adopted in solving the three problems mentioned in Chapter 1. According to Bertrand and Fransoo (2002), the research discussed in this thesis belongs to the category of *axiomatic research* where the primary concern is to produce solutions within a defined model while ensuring that the solutions generated provide insights to the real problem. The problems are described based on the objective models we developed during the study, which explain the behaviour of real-life operational processes with 2D cutting and packing functions. With respect to each problem, the models investigated in this study are determined by various methods and techniques in mathematics, computer science and management science.

As explained in most of the axiomatic researches, the study discussed in this thesis is interested in developing strategies, tools and techniques to find good solutions for a newly defined problem or to improve the results for a problem already considered in the literature. In general, each problem of the study has been passed through three phases. In the first phase, the model of the problem is designed, which includes conceptualization, reviewing similar types of problems, making decisions about variables, defining the scope of the problem and formulating the model. During the second phase, the formulated problem is solved using the principles of mathematics, geometry, computer science and operational research. In the third phase, an analysis is conducted using the implementation of results, so that useful insights can be obtained about the problem and its relatives.

The problem modelling is started with the intention of identifying the characteristics of the problems within the operational process we aim at. A comprehensive literature review is conducted using peer reviewed articles specific to the problem type and its variants. By focusing on the research gaps of existing literature the problem types that have been rarely considered are identified. Chapter 2 reviews those gaps and presents the respective problem types and their features. We then focus on developing the models for each problem in a way that these can be read through a computer program and solvable with a designed algorithm. The concept of geometry used in this study is a key research topic within irregular shape C&P problems and many techniques have been published describing how to handle the irregularities of pieces, especially in the overlapping and containment calculations.

Literature review in Chapter 2 reveals two main categories of solution methods; *exact methods* and *heuristic methods*. Exact methods have been applied in order to find the global optimal solution to the given problem. However, the existing studies reveal that exact methods are not able to prove optimality for large-size" instances which causes a high level of complexity to the C&P problems. As reviewed in Chapter 2, so far, the exact methods find an optimal solution for most of the instances with up to 12 pieces and the methods even cannot find a feasible solution with more than 30 pieces for most instances (Cherri et al., 2016). Heuristic techniques, on the other hand, have been developed to provide a good feasible solution in such cases. Though the heuristic approaches are generally fast, they do not guarantee to solve the problem towards optimality. For the improvement of the solutions from the heuristic approaches, the local search techniques have been used. The purpose of local search techniques is to reach a local optimum of the specific area it explored over the solution space. Usually, the local search methods cannot perform an intelligent search as they do not access other areas of the solution space. To avoid this limitation of being trapped in a local optimum, researchers use metaheuristic algorithms. Metaheuristics are a set of high-level algorithms which apply a set of guiding strategies in designing problem-specific heuristics and local search techniques. Meta-heuristics conduct an intense search over the solution space starting with one or more input solutions and find a good solution. Unlike exact methods, the heuristic methods are more popular in solving irregular shape packing problems. Even though this approach is not able to prove optimality or to provide any measure of how far it is away from the optimal, still they are capable in finding a reasonably good solution when they are applied in the cases that arise in practical scenarios where large-size instances are involved. Therefore, in this study, we use heuristic methods when solving the three versions of IBPP.

The remainder of this chapter is organised as follows. Section 3.2 describes the mostly used computational techniques involve in calculating overlaps and containments of irregular shaped pieces when they are placing in the containers. In this section, we also describe the specific techniques we use in this study by reviewing the pros and cons of each technique. Section 3.3 presents a variety of basic heuristics and some examples of popular metaheuristic frameworks used in irregular shape packing problems. The purpose of this presentation is to identify the features of each approach to involve them in solving the problems wherever they are necessary. In Section 3.4 we provide an overview of implementation details of our algorithm in general. This includes the specification of the instances and specification of computer tools and hardware used to conduct our experiment.

3.2 Computational Geometric Techniques

The main purpose of involving geometric techniques is the calculation of overlaps and containments of pieces within the objects. The overlap calculation evaluates whether two pieces are overlapped, touched or separated. The containment evaluation checks that the piece is placed within the object (i.e. bin). Several methods have been adopted by researchers to perform this task, where the performance and accuracy of them are different.

3.2.1 Representation of pieces

Some representation techniques approximate the geometric shape of the irregular piece while some retain the shape as a simple polygon. The accuracy of piece placements is not 100% accurate in approximation methods since it cannot represent the exact shape of the piece. However, the approximation can reduce the complexity of shapes and makes the execution of operations simple and efficient. Therefore, there is always a trade-off between speed and accuracy that both affect the quality of the solutions. Usually the overlap and containment calculation is based on the technique adapted for representing the pieces and objects.

As one of the approximation techniques; Segenreich and Braga (1986) and Oliveira and Ferreira (1993) discretized the whole packing area into several discrete areas (i.e. a grid). In this case, identifying non-overlapping placement positions is easy since it can be easily determined by the grid intervals. Dagli and Hajakbari (1990) placed a grid inside the rectangular enclosure of the piece. This technique has been widely used in different papers (Oliveira and Ferreira, 1993; Babu and Babu, 2001; Xu et al., 2013; Guo et al., 2015). The major aim of this technique in common is identifying non-overlapping placement positions as it can be easily determined by the grid intervals. The same concept is also used as a pixel / raster method which divides the packing space into the pixels and each pixel has been assigned a value to indicate if it is occupied by a piece (Babu and Babu, 2001). In this case, the geometric information of pieces and stock sheet are transformed to a data matrix following a certain codification, to represent the irregular pieces, holes and defective areas within the stock sheet. The intuition of this technique is to convert the geometric information to a coded data matrix. Bennell and Oliveira (2008) and Bennell and Oliveira (2009) described several features of those discretized methods. According to their review, discrete/raster methods are simple to code and they are efficient in checking the geometric feasibility. However, the representation of the shape is not accurate and that sometimes misleads the overlapping tests by showing pieces as overlapped when they are not actually overlapping. On the other hand, the technique is memory intensive which may require computers with higher memory capacity to realise computational efficiency. In order to increase the accuracy, raster approaches increase the resolution of pixels which leads more towards expensive computation. Figure 3.1 illustrates a few uses of discretized methods with different codification schemes as examples.

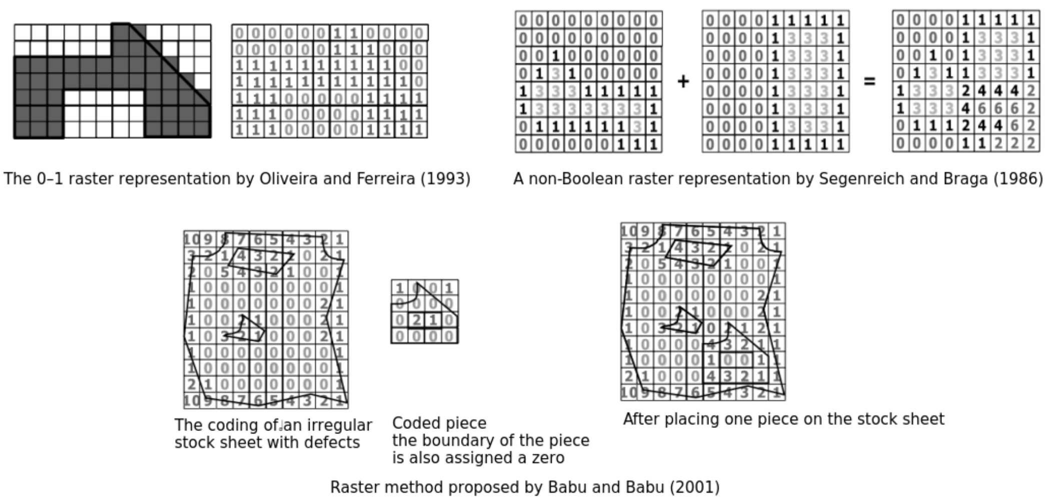

The 0–1 raster representation by Oliveira and Ferreira (1993)
A non-Boolean raster representation by Segenreich and Braga (1986)
The coding of an irregular stock sheet with defects
Coded piece the boundary of the piece is also assigned a zero
After placing one piece on the stock sheet
Raster method proposed by Babu and Babu (2001)

Figure 3.1: Discrete representation methods (extracted from Bennell and Oliveira (2008))

Some researchers approximate the irregular shapes into rectangles or convex polygons (Hopper and Turton, 2001). Heckmann and Lengauer (1995) represented irregular pieces by their polygonal enclosure. The method has only provided an approximation rather than representing the exact shape of the piece. When dealing with C&P problems, accurate representation of pieces directly affects the accuracy of overlap calculation. Since a given piece is constructed by means of segments, for a complex shape, it requires a higher number of segments to represent the shape accurately. Encoding an arbitrary closed curve to a minimal enclosed rectangle (Freeman and Shapira, 1975), packing several arbitrary polygons and curved pieces into rectangles (Martin and Stephenson, 1988), algorithms to find the convex enclosure with the smallest area for two concave polygons (Grinde and Cavalier, 1995) are a few attempts for approximating irregular shapes to a finite number of geometric segments. However, these techniques seem to be outdated as they have not been used in recent methods for solving irregular shape C&P problems.

Rocha et al. (2013) introduced circle covering (CC) for overlap computation in nesting problem allowing free rotations of pieces. The technique represents a piece as a set of overlapping circles whose sum of areas approximates the piece area. This approximation technique aims to minimise the necessary number of circles so that the complexity of related operations can be reduced. As the paper states, the main difficulty of CC approximation is to achieve a lower approximation error with the minimum of a number of circles. Rocha et al. (2014) discussed the trade-off between piece representation and the corresponding number of circles required. They demonstrated that the low-resolution approximation achieves better computational efficiency if the accuracy of the approximation is compromised. In this case, the low-resolution circle coverings contain a less number of circles than the high resolution covering which leads to a lower quality of representation. Details of this technique are available in Rocha (2014).

To take advantage of accurate representation of pieces, several researchers use the exact shape of an irregular piece as a simple polygon (e.g. Gomes and Oliveira (2006); Bennell and Song (2008)). However, in order to take the benefit of accuracy, it is necessary to use an accurate overlap calculation technique (Bennell, 1998).

3.2.2 Overlap calculation and finding feasible placement positions

D-function is used as an efficient joint approach to giving the relative position of a certain point with respect to an oriented edge (Konopasek, 1981). Mahadevan (1984) used the D-functions to find the relative position of two oriented edges. When two pieces are represented accurately by a closed series of vertices and edges, the relative positions of the two polygons can be identified using this approach (Bennell and Oliveira, 2008). For example, Mahadevan (1984) and Ghosh (1991b) used trigonometry and D-function to detect the intersection of edges in the overlapping pieces. According to Bennell and Oliveira (2008), the approach can further identify whether the points of a piece are located within another piece. This method maintains a better accuracy in overlap calculation which is governed by a set of hierarchical tests to evaluate if two polygons are overlapping or not. A detail description of these hierarchical tests is available in Bennell and Oliveira (2008). An advantage of using trigonometry with D-function is the accuracy they provide in overlaps calculations since the polygons are represented in their exact shape. Bennell and Oliveira (2008) reports that the techniques are not efficient in two terms; First, they review that the computational effort put on these calculations is higher with the number of floating points used. Second, they believe it is not efficient to apply such calculations in an iterative search process since these calculations have to be repeated from scratch at every step. Due to these reasons, it is rare to find using D-functions in recently published papers of C&P steam.

Adamowicz and Albano (1976) exploited the use of No-fit polygons to find the minimum enclosure of the number of nested irregular pieces. Later, the concept was widely developed and used by many researchers in C&P stream (e.g. Mahadevan (1984), Milenkovic et al. (1991), Li and Milenkovic (1995), Bennell et al. (2001), Gomes and Oliveira (2006), Huyao et al. (2007), Burke et al. (2007), Bennell and Song (2008), Imamichi et al. (2009) ,Bennell and Song (2010), Burke et al. (2010), Leung et al. (2012), Junior et al. (2013), Hu et al. (2015) Martinez-Sykora et al. (2015), Leao et al. (2016), Martinez-Sykora et al. (2017)). These examples reveal that no-fit polygon (NFP) is the most popular technique used in literature recently. Bennell et al. (2001) and Bennell and Oliveira (2008) denote NFP of two polygons A and B as NFP_{AB} the resulting polygon from a sliding operation between A and B. Polygon orientations are fixed. When polygon A is in a fixed position, then polygon B is considered to be the tracing polygon that moves along the perimeter of A so that B slides around A. Then NFP_{AB} is described as the path of touching points of B with A, marking the locus of a reference point on B as it traces around the boundary of A (Bennell et al., 2001; Bennell and Oliveira, 2008). Figure 3.2 illustrates how the locus of the reference point on polygon B creates the NFP_{AB} as polygon B traces around polygon A.

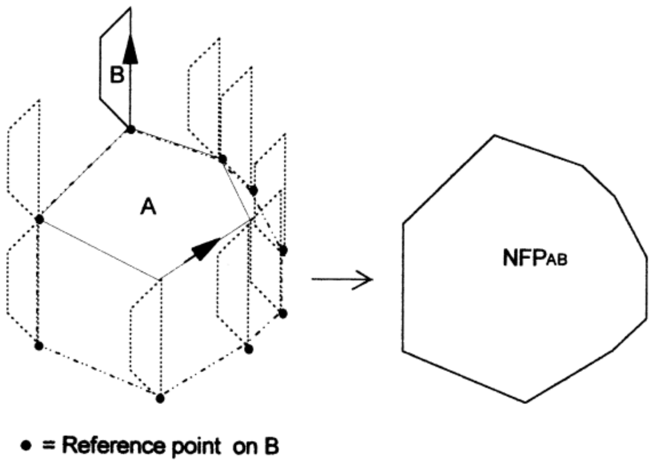


Figure 3.2: NFP (A,B): The reference point on B traces around A (Bennell et al., 2001))

When the position of A is known and B is positioned so that its reference point is inside NFP_{AB}, then it overlaps with polygon A. If the reference point of B is on the boundary of NFP, then B touches A. When the reference point is outside of NFP, it shows that A and B neither overlap nor touch (Junior et al., 2013). As the locus of points where the reference point of polygon B should be placed in contact with polygon A, its exterior, represents the set of feasible coordinates for the reference point of polygon B so that it doesn't overlap A (Leao et al., 2016). Since this is a promising technique for overlap calculation and placement decision of irregular pieces, it is vital to find an efficient technique to derive NFPs. Usually, the calculation of NFP for purely convex polygons is simple (Cuninghame-Green, 1989). This is the most simple case in terms of complexity of polygonal shapes. However, Bennell and Oliveira (2008) illustrates some cases (illustrated in Figure 3.3) which require a considerable effort to calculate the correct NFP. Bennell and Song (2008) referred those as the degenerate cases.

A related concept called *inner-fit-rectangle* was discussed in Gomes and Oliveira (2001). They denote the rectangle it creates when an irregular piece slides along the internal contour of the rectangle object as the *inner-fit-rectangle*. Extending this idea, Liu and

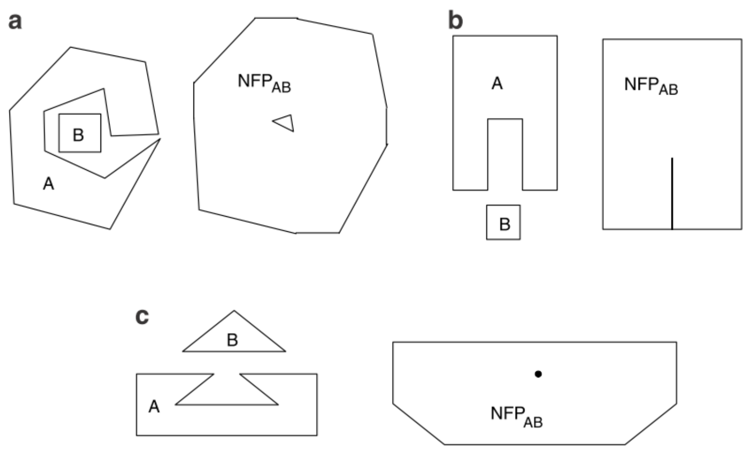


Figure 3.3: Different cases of generating NFPs (Bennell and Oliveira, 2008): a) B fits inside the concavity space of A, b) B can slide into the concavity of A in one direction, c) B fits only in one point inside the concavity of B

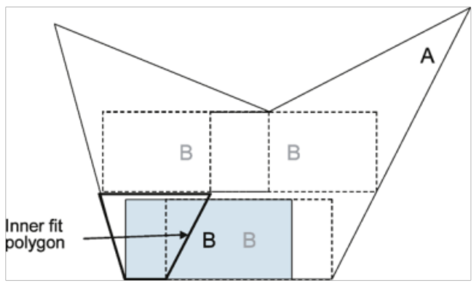


Figure 3.4: Inner-fit Polygon (extract from Bennell and Song (2010))

He (2006) described *Inner-fit NFP* as the locus of the reference point of the piece when the piece slides around the inside boundary of an irregular container (as illustrated in 3.4). In Bennell and Song (2010)'s paper uses the same concept and called it the *Inner-fit polygon* (IFP). Since IFP finds the feasible placement path of a polygon, Bennell and Song (2010) states that IFP can be useful for packing pieces inside an irregular shape (e.g. cutting irregular shapes natural leather hides).

In order to generate NFPs, Mahadevan (1984) used a sliding algorithm to find the movement path of the tracing polygon around the fixed polygon. However, this only finds the outer boundary of the NFP. The approach does not work with holes and does not work for the cases illustrated in Figure 3.3.

Milenkovic et al. (1991) and Bennell (1998) use the Minkowski Sum concept to generate NFPs. They describe the approach as *Minkowski difference*. Using Ghosh (1991a)'s Boundary Addition Theorems, Bennell (1998) described how NFP can be generated for the case where one polygon has concavity and the other polygon is convex. In order to deal with the case where neither polygon is convex, Bennell et al. (2001) proposed an approach that first replaces B by its convex hull and solves the simple case of deriving the NFP of a convex polygon and a polygon with the concavity, as mentioned in Ghosh

(1991a). Then the resulted NFP (this is considered as an intermediary NFP) is repaired to obtain the real NFP. Details of this can be found in Bennell et al. (2001) and Bennell and Oliveira (2008).

Burke et al. (2007) introduced a different method called *robust orbital method* to generate NFPs using the Mahadevan (1984)'s sliding algorithm. This resolved the limitations of Mahadevan (1984)'s approach related with the degenerate cases. Later, Bennell and Song (2008) presented a better approach based on the Minkowski Sum to generate true boundary of the NFP, including holes, slits and exact fits, without generating the convex hull of one polygon and then repairing the resulting intermediate NFP as in Bennell et al. (2001). The authors presented an empirical analysis to demonstrate that this method is reasonably efficient. This method is also capable of generating NFP for the degenerate cases and the polygons containing spiralling concavities.

Based on the quick method to find the NFP of two convex polygons proposed by Cuninghame-Green (1989), Watson and Tobias (1999) and Agarwal et al. (2002) decomposed simple polygons into convex sub-polygons. The first step of the approach is to decompose each given polygon into convex sub-polygons and generate the NFP of each pair of sub-polygons. Next, they combine these NFPs of the sub-polygons and generate the final NFP of the given two polygons. In comparison with other methods of generating NFP, the method possesses higher complexity in the decomposition and combining operations.

Stoyan et al. (2001, 2004) developed and applied Phi-functions to formulate the mutual position of simple geometric objects. A specific research group in C&P stream involves using Phi-functions to describe the placement and interactions of geometric objects. The phi-function measures the relative position of objects. The phi-function normalizes its value as the Euclidean distance between the two phi-objects (i.e shapes) (Bennell et al., 2010; Chernov et al., 2012). At early stages, Stoyan et al. (2004) derived the phi-function for primary objects such as rectangles, circles, regular polygons, convex polygons as well as the compliment of these shapes, based on trigonometric theories. Bennell et al. (2010) pointed out the effectiveness of Phi-functions in terms of measuring the degree of overlap or distance between shapes. Even though Phi-functions can be used directly in the mathematical modelling part of the problem, it is necessary to derive the corresponding function for the objects beforehand. Chernov et al. (2012) have derived ready-to-use basic phi-functions to construct radical free phi-functions for arbitrary shaped 2D objects. The major disadvantage of this method is the limitation of constructing phi-functions for a selected set of pieces; e.g. ellipses. Extending the concept of phi-functions, a new concept; quasi-phi-functions is introduced by Stoyan et al. (2015b). Quasi-phi-functions are simpler than phi-functions. For example, it is possible to develop Quasi-phi-functions even for the objects that construction of phi functions are limited. (Stoyan et al., 2015b). Stoyan et al. (2015b) highlight that the new method would lead to a larger set of parameters and extra variables due to the derivation of new functions. The Stoyan et al. (2015b)'s quasi-phi-functions have also been successfully implemented with various types of objects (including ellipses) that can be continuous rotated. Stoyan et al. (2015a) used phi-functions and quasi-phi-functions to solve placement problem of irregular shapes using a non-linear programming model. Both Phi-functions and Quasi-phi-functions approaches deal with an accurate representation of objects with no approximation. However, the approach becomes inefficient when highly irregular objects where the compositions of those objects are complex, those objects are compiled from basic objects (Stoyan et al., 2015b,a).

In summary, our methodology of addressing the geometric aspect involved mainly the steps in *representing pieces*, *identifying the data structures* that the algorithm is implemented and *selecting a proper overlap calculation technique* to find feasible placement positions for pieces.

As described in Section 3.2.1, there are three ways of modelling the positioning of pieces in a stock sheet and accordingly the representing method is decided. The first approach uses the continuous positioning of the pieces along the area of the stock sheet, while the second approach considers discretized positioning. The third approach is based on positioning with semi-discrete geometric representation where one axis of the placement coordinate system along the stock sheet is discrete and the other axis is continuous. Out of the three, we used the continuous positioning which is based on geometric representation when modelling the problems. This approach addresses one of the objectives of our study by guaranteeing accurate placement of pieces without any approximation. Also, it was evident that existing research that used *geometric representation* for irregular packing problems have generated better results than the other two approaches (Egeblad et al., 2007; Imamichi et al., 2009; Umetani et al., 2009; Bennell and Song, 2010; Burke et al., 2010; Elkeran, 2013).

Irregular pieces and rectangular bins were the two main elements of the model of our problem. An irregular piece was represented as a closed series of vertices and edges oriented in anti-clockwise direction. Hence, the model allowed irregular shapes to be defined as simple polygons. Any other geometric cases such as partial solutions and holes were also defined as simple polygons. Accordingly, the data structure of a piece is described by a *list of vertices in the polygon*, an orientation angle which defines the rotation of the piece with reference to the initial orientation and a reference point which defines as the bottom left point of the enclosing rectangle of the piece at its current orientation.

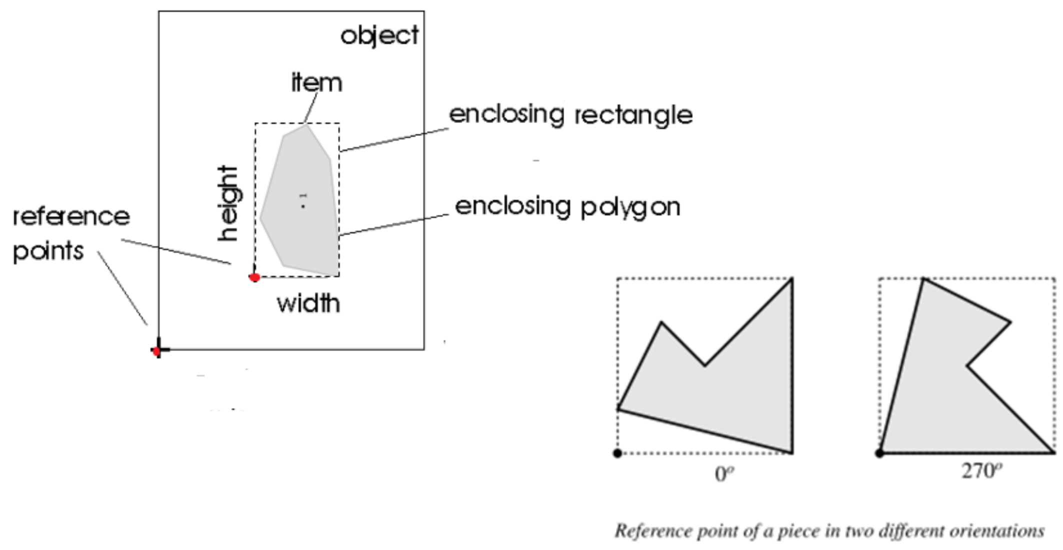


Figure 3.5: Shape representation of a piece

The geometric representation enabled close nesting of pieces, hence brings the higher accuracy of the solution. However, the higher number of vertices consume longer computational time; since the calculation of no-fit polygon need more time when there is a higher number of edges in the corresponding polygons. Due to this reason, we eliminated the very narrow concavities of some polygons (i.e. especially in the merged polygon of irregular shapes inside the container) without affecting to the accurate placement of pieces.

Similar to pieces, a bin was described by a list of four vertices (i.e. rectangular bin) and a reference point (bottom left point of the bin).

The third step; the *overlap calculation* was performed involving the concept, no-fit polygons (NFPs). As we reviewed in Section 3.2.2, the boundary of NFP represents the path of feasible placement positions of a polygon with respect to the other neighbour polygon. Therefore by finding this path, we can find a feasible placement positions of a polygon with respect to the other so that those two polygons are neither overlapped nor separated. As the most famous tool in the recent literature, the recently developed methods for generating NFPs work efficiently with various types of polygons, without compromising the benefit of accuracy (Burke et al., 2007; Bennell and Song, 2008; Bennell and Oliveira, 2008; Burke et al., 2010). We used the NFP generation method based on the Minkowski Sum approach by Bennell and Song (2008), to calculate NFPs in our constructive algorithms, since it was able to facilitate the expected requirements of our algorithms such as dealing with jigsaw-type shapes and interlocking polygons.

3.3 Heuristics-based Approaches

In this section, we describe some design principles of heuristics and metaheuristics approaches and briefly introduce some of the key facts to be understood when implementing those approaches.

Heuristics based approaches find good solutions to large-size problems at acceptable computational costs. They are simple in understanding, easy in implementation and less costly in computational time than the exact algorithms. Generally, the heuristics based methods are distinguished between constructive heuristics and local search methods.

3.3.1 Constructive heuristics

Constructive heuristics always generate a solution from scratch by adding the components which require to build up a solution. Initially, it starts with an empty partial solution and then it is gradually extended until the solution is complete. As reviewed in Chapter 2, Sections 2.5.1 and 2.6.1, C&P researchers have used constructive methods to generate solutions by adding pieces to the partial solution, one after the other. In this case, the solution of a constructive heuristic contains a sequence of partial solutions. The heuristic criteria of the algorithm decide the extension required at each insertion of components. The extension made at one insertion affects the next extension of the partial solution.

The design structure of constructive heuristics is straight forward. Some methods insert each piece following a given order of pieces. Also, there is another method which chooses the next insert piece using dynamic selection (see. Chapter 2, Section 2.5.1.2). In this case, the constructive methods follow a selection procedure to select the piece to be positioned next, by evaluating the best partial solution among other partial solutions correspond to the pieces to be placed next. For C&P problems, the decision related to constructive heuristics are attached with the insertion order of pieces and the placement rule used in the algorithm to insert the new piece. Chapter 2 reviews a set of well-known ordering and placement strategies used in irregular shape packing problems.

3.3.2 Local search methods

As described in Talbi (2009), the local search is operated on the appropriately defined neighbourhood of the selected feasible solution s. The neighbourhood is a set of solutions that are close to s. They are derived from s so that they share some amount of structure of solution s. The constructive heuristic can be used to find neighbour solutions, by making small modifications; called *moves*.

As a basic local search algorithm, *iterative improvement* tries to explore the neighbourhood as follows. Starting with an initial solution the algorithm moves from one neighbour to another to find a better solution until a termination criterion is met. If a neighbour has an improved objective value than the current solution, then the neighbour becomes the new current solution. The search process is iterated until no such accessible improved neighbour is found (see Figure 3.6). The solution found by the local search method may not be the best solution to the problem, but it finds a good solution.

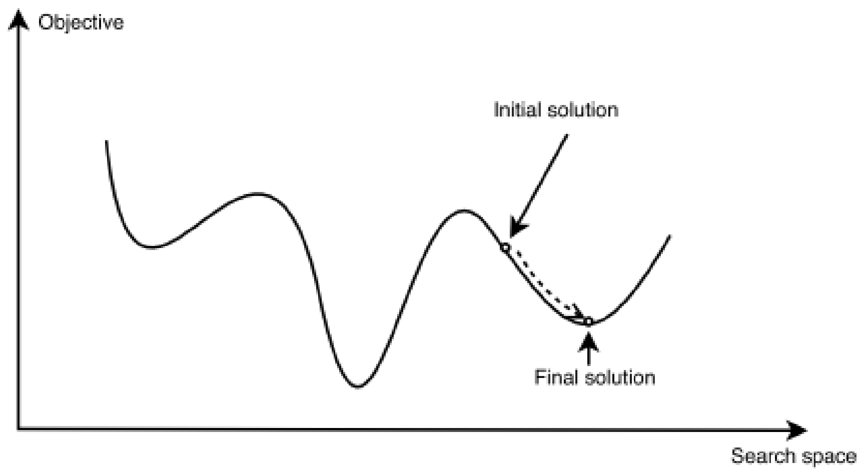


Figure 3.6: Iterative Improvement Local search (Decent) behavior (extracted from Talbi (2009))

Even for a basic local search method, it requires deciding the way of generating the initial solution and the procedure for producing neighbours. Usually, the initial solution is produced by the constructive heuristic following a random approach (e.g. a random order of pieces) or a predefined strategy (e.g. a predefined rule of ordering pieces). The selection of an initial solution may cause a significant effect in a simple local search heuristic. However, more powerful improved local search techniques with broad search space will provide a solution which is almost independent from the initial solution.

In addition to the iterative improvement, Talbi (2009) discusses some other variations of basic local search algorithms. In the above case, it accepts the solution corresponds to the first improving move. This is referred as the first-improvement way of choosing the improved neighbour. Instead of selecting the first improving solution, there is another way called best-improvement, in which the whole neighbourhood is searched before a solution is accepted as a better one. As Gonzalez (2007) states, the local search methods based on the first-improvement is comparatively faster in finding individual improvement steps since the full neighbourhood will only be used when the current solution is the local optimal.

The local search algorithms are intuitively understandable and easier to implement than exact algorithms. Most importantly, they are very useful when solving problems with large instances. However, a local search algorithm only finds a local optimal solution.

The local optimal solution is the best solution that can be found considering a certain neighbouring configuration that the local search being applied. This situation in existing literature has been referred as being stuck in local optima which researchers consider as a limitation of local search. In order to overcome this, researchers use repeated local search; known as *Restarts*, or more complex algorithm schemes based on iterations such as Iterated Local Search, Tabu Search and Simulated Annealing.

3.3.3 Metaheuristic Algorithms

Metaheuristics are problem-independent heuristic frameworks which guide the heuristics which are problem-specific. Meta-heuristics find good solutions in an acceptable computational time. They provide reasonable solutions to the problem when the search being stuck in local optima. Meta-heuristics resolve this by allowing worsening moves or generating new starting solutions for the local search in an intelligent manner than the random initial solutions in multi-start local approaches. A key feature of metaheuristic frameworks to improve the solution search process is the implementation of randomization while controlling intensification and diversification of the search process. The *intensification* is used when there is a solution which needs to be further explored by its neighbours. This is referred as the exploitation of the best solutions found. This is usually implemented with the Local Search techniques so that the selected promising regions are explored thoroughly to find better solutions.

In the diversification phase, instead of conducting a search in a particular area which is not further appealing, the search is guided towards a new area. Here the non-explored regions are explored to ensure that most regions of the search space are explored. This is usually implemented with random search in the search space.

Different metaheuristic algorithms have different techniques to implement these intensification and diversification phases. Before applying a metaheuristic, it is also important to know the size of input instances the algorithm is supposed to solve and the level of complexity of the problem. There are some hard problems which are solvable using the exact approach if they are with small instances. Also, the time it takes to search and obtain a reasonably acceptable solution is an important concern in the selection of metaheuristic.

Next, we briefly describe the main features of metaheuristic approaches focusing particularly on those which are related to the irregular shape C&P problems and the solution methods present in this thesis.

3.3.3.1 Simulated Annealing (SA)

According to Gendreau and Potvin (2010), the principle behind SA is derived from physics. The idea is to simulate the annealing process of solid, where the solid changes its structure as the temperature cools, which was proposed by Metropolis et al. (1953). Kirkpatrick et al. (1983) introduced Simulated Annealing in solving optimisation problems. In combinatorial optimisation scenario, the elements of problems can be mapped to the elements of the physical cooling process.

The objective of SA is to escape from local optima thereby exploring the opportunity of searching a broader search space. The general procedure of SA is described in Talbi (2009) as follows. Starting from an initial solution which becomes the current best solution (x) at the first step, a tentative neighbour solution ($x^{'}$) is generated at each iteration with respect to x. If the neighbour solution improves the value of the objective function (i.e. $f(x^{'}) < f(x)$), then it is accepted. Otherwise, the non-improving solution is accepted based on a given probability which depends on the amount of improvement of the objective function $f(x) - f(x^{'})$ and the control parameter (T). At higher values of T, the probability of acceptance of non-improving solutions are high, hence almost all the solutions are accepted. When the algorithm finds a local optimum at a certain T value, T is decreased according to the cooling schedule, and continue the local search. When T is gradually decreased, the probability of acceptance decreases, hence less non-improving solutions is accepted. Accordingly, at the end of the procedure, the best solution is found.

In comparison to the basic local search which is getting stuck at local optima. SA solves this issue by allowing more time for less quality moves during the search process. In other words, it allows some high-cost solutions so that the search can escape from local optima. Also, SA selects a random move from the neighbourhood while basic local search selects the best move from all those available at higher temperatures the search is analogous to random search. Then the temperature is decreasing though out the search. When it is zero, then only the better moves will be accepted similar to the basic local search.

3.3.3.2 Tabu Search (TS)

According to Gendreau and Potvin (2010), Tabu Search was introduced as an optimisation approach in Glover (1986). The general idea of TS is to search a space by choosing a point, and then going to its best neighbour which is not in the tabu list. The tabu list records forbidden moves, which are referred to as Tabu moves. As key features, the method uses memory to conduct a local search. As described in Dréo (2006) and Gendreau and Potvin (2010), the Tabu search prevents revisiting and maintains increased diversity in exploration which helps to escape from local optima.

Tabu search uses a search approach which is similar to the best-improvement local search. However, it escapes from local optima by accepting non-improving solutions when there is no improvement in neighbours. Unlike in SA, TS usually explores the whole neighbourhood in a deterministic way.

A significant difference between simulated annealing and tabu search is their level of use of memory. Simulated annealing has no memory, whereas Tabu Search has the ability to refer information stored about previous moves. As described in Talbi (2009), there are three purposes of using memory in TS. First, it prevents the search from revisiting previously visited solutions. Second, it helps diversification by searching the unexplored areas of the search space. Third, TS uses it memory structure to store best-found solutions and explore them further (i.e. application of intensification).

3.3.3.3 Iterated Local Search (ILS)

As described in Gendreau and Potvin (2010), the general idea of Iterated Local Search is to apply iteratively a local search until a local optimum is reached and then apply a perturbation (also known as a kick), in order to escape from the local optimum and explore further solution space. In this case local search only moves towards better solutions. The main role of the kicks is to change the current solution enough while keeping the good properties of the current local optimum. According to Lourenço et al. (2010), the idea of ILS has been proposed in many forms in its early research. One early implementation of ILS was stated as iterated decent in Baum (1986) and Baum (1987).

Basic procedure:

Algorithm 1: ILS

```
1 Initialize;
2 Generate initial solution, $x \in S$;
3 $x^* \leftarrow LocalSeach(x)$;
4 Repeat
5   $x^{'} \leftarrow Perturbation(x^*, history)$;
6   $x^{*'} \leftarrow LocalSearch(x^{'})$;
7   $x^* \leftarrow AcceptanceCriterion(x^*, x^{*'}, history)$;
8 Until termination condition met
```

The entire procedure is repeated until a stopping criterion; generally, a limit on computation time or the total number of iterations is satisfied.

As described in Talbi (2009), the ILS procedure allows using single solution-based metaheuristics (e.g. simple iterative improvement local search, TS, SA). However, it is vital to control the strength of perturbation when implementing ILS as described in Luke (2013). If perturbation is too strong, then it is close to random restart behaviour. If

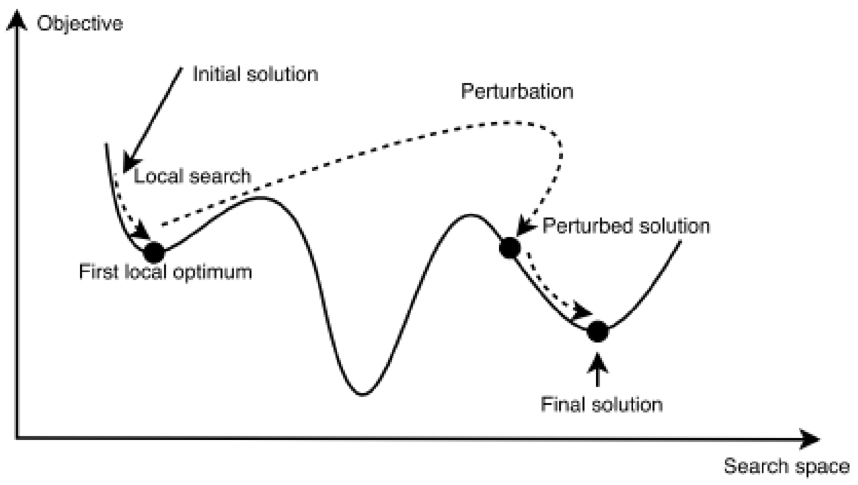


Figure 3.7: Working principle of iterated local search (extracted from Talbi (2009))

perturbation is too weak, then the algorithm will not perform as expected since the Local Search may undo the effect of the perturbation.

3.3.3.4 Variable Neighbourhood Search (VNS)

Variable Neighbourhood Search is proposed in Mladenović and Hansen (1997) as a search method which explores the neighbourhood to find a better solution. As described in Figure 3.8, VNS explores a set of neighbourhoods and visits different local optima. If a search method can explore all possible neighbourhood structures, then finding global optimum is possible. However, finding all possible neighbourhood structures is hard for a complex problem. In this context, VNS deals with different neighbourhood structures by changing the neighbourhood during the search process. When implementing VNS, the algorithm generates different neighbourhood structures such as Cross, Swap, Exchange, and Flip. The movement between neighbourhoods is either deterministic, random or both.

3.3.3.5 Genetic Algorithm (GA)

GA is a population-based search metaheuristic which is inspired by the evolution theory of the natural world which follows the principle; survival of the fittest. GA was formally recognised by Holland in 1975 (Holland, 1992) (the book was originally published in 1975). A solution in search space is referred as individuals (or chromosomes) that genes of a chromosome represent parts of the solution or solution construction. The numerical evaluation of an individual is known as its fitness which is an evaluation value of the quality of each individual.

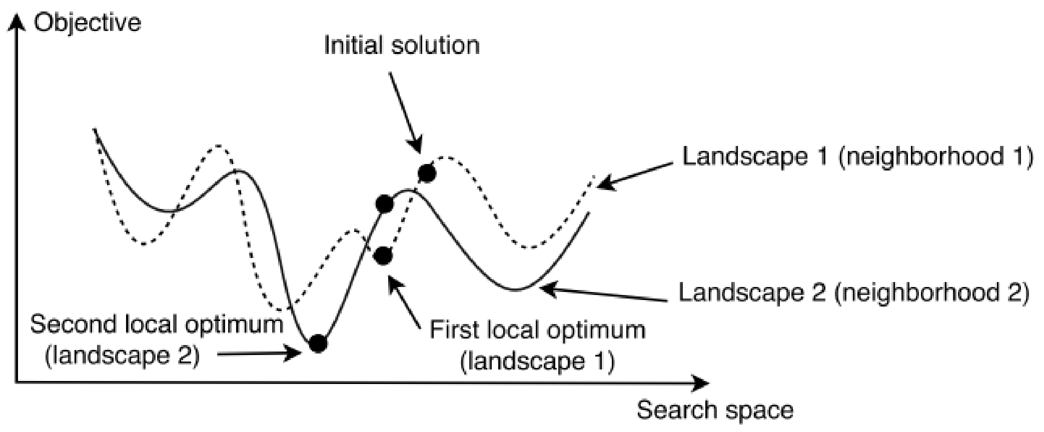


Figure 3.8: Variable neighbourhood search in two neighbourhood (extracted from Talbi (2009))

With respect to neighbourhood 1, the first local optimum is derived. Then the second local optimum in neighbourhood 2 is derived from the first local optimum.

When searching occurs, the population is being replaced by new individuals (chromosomes). The parents are selected for mating based on the fitness. The aim is to propagate the good properties of a chromosome into new individuals so that the population can be improved when generations progress. The new chromosomes are made by applying the below two types of genetic operators that mimic the evolution of natural systems;

1) Crossover operator: This considers two chromosomes (i.e. parents), and combine their features by changing their genes to produce new chromosomes (i.e. children). Usually, two parents make two children. The crossover probability (P_c) parameter describes the rate of crossover happens. If $P_c = 0\%$ then all offsprings become exact copies of parents. If $P_c = 100\%$ then all offsprings are generated by crossover.

2) Mutation operator. This introduces diversity and allows enter new genetic features to the population. After crossover, the chromosomes are subject to mutation. Mutation supports exploration during a search while crossover supports exploitation. There are different forms of mutation operators; such as mutation operators for binary or integer representation (e.g. bit flipping) and mutation operators for permutation representation (e.g. swap mutation, inversion mutation, scramble mutation) as described in Eiben and Smith (2013). The main difference between those two methods is that the mutation in binary/integer representation mutates each gene individually while the other consider mutation of the whole chromosome rather focusing each gene individually. Similar to (P_c), the mutation probability (P_m) describes the how often of the part in the chromosome is mutated.

In order to handle a GA search, we have to consider following specification of the operators and parameters.

- Population Size: Number of chromosomes in the population

- Fitness function: The function which evaluates the quality of chromosomes
- Crossover Operators: The mechanism of crossover to produce children
- Crossover Probability: Chance of crossover; generally 0.6-1.0
- Mutation Operators: The mechanism of mutating the features within chromosomes
- Mutation Probability: Chance that a part of a child is changed randomly
- Elitism: The proportion of best chromosomes be ensured a position in the next generation

The general structure of a simple GA is represented as follows

Algorithm 2: Genetic algorithm

```
1 Produce an initial population of individuals;
2 Evaluate the fitness of all individuals;
3 while termination condition not met do
4     select fitter individuals for reproduction;
5     recombine between individuals;
6     mutate individuals;
7     evaluate the fitness of the modified individuals;
8     generate a new population;
9 end
```

According to Talbi (2009), GA requires handling two main search strategies. First, the selection strategy determines which parents are selected for reproduction. Usually, the chromosomes having better fitness has more chance of being parents. However, GA usually allows some worst-fit individuals to be selected as well. Second, the reproduction strategy determines which type of mating operators are used. It is possible to use one or more recombination operators in GA, following the rules below. The combination operators must inherit the same features from each parent when making children and the outcome of it must produce valid chromosomes according to the solution representation technique used. Both strategies support GA search to keep a balance between the diversity and convergence speed.

3.3.3.6 Greedy Randomised Adaptive Search Procedure (GRASP)

GRASP is a multi-start metaheuristic that combines constructive heuristics and local search, which was first proposed in Feo and Resende (1989). As explained in Feo and Resende (1989), Resende and Ribeiro (2010) and Resende and Ribeiro (2016), GRASP consists two phases; solution construction and solution improvement. These two phases are performed iteratively and the best solution is always updated and stored in the memory till the algorithm terminates.

The solution construction method is characterised by a greedy constructive heuristic and randomization. The constructive heuristic constructs the solution by adding each element of the solution to the current partial solution, in which the choice of the next element to be added to the partial solution is picked by randomly from a candidate list of elements. This list of candidates is denoted as the Restricted Candidate List (RCL). At each construction iteration, the RCL is formed with all the elements which can be added to the current partial solution in the order of their short-term benefit which is measured by a greedy function. At each iteration of the construction phase, each candidate element is updated with a score (i.e. myopic benefit of adding that element to the partial solution) measured by the greedy function.

3.3.3.7 Parameter tuning

When applying metaheuristics, it requires tuning many parameters used within the algorithm. The performance of the algorithm heavily depends on its parameters hence it shows a significant difference in performance improvement by a well-tuned algorithm. According to Peng-Yeng (2012), the optimal tuning values of the parameters are mainly based on the problem, the instance and the search time assigned by the user.

As mentioned in Peng-Yeng (2012), the parameter tuning can be done in two different ways; on-line parameter tuning and off-line parameter tuning. In the on-line tuning, the parameter values are dynamically manipulated and updated within the execution of the metaheuristic. The on-line tuning techniques are typically used as a machine learning technique, which deals with a large number of instances. In the off-line parameter tuning, the parameter values assigned to different parameters are pre-defined before the running of the metaheuristic.

As mentioned in Peng-Yeng (2012), the earlier use of off-line parameter tuning considered only the one-by-one parameter tuning (each parameter in tuned independently) which doesn't guarantee the optimal combination of parameter values to maximise the overall performance of the algorithm. To resolve this, researchers use the design of experiments, which uses different levels of potential values for each parameter in the algorithm. With a full design of each level in each parameter (including the combinations) best levels are determined. This is reasonably a manageable method if the designer has some understanding about possible levels of each parameter even though it takes a significant time to conduct the parameter tuning.

As other methods, Talbi (2009) formulates the tuning parameters as the optimisation problem and try to find the parameter values. Hutter et al. (2009) propose automatic algorithm configuration framework called *PramILS*, as a procedure of automatic tuning of parameters. Recently, many papers have been published on parameter tuning and Peng-Yeng (2012) presents most of those approaches in his publications.

Another promising approach for parameter tuning is Irace: Iterated Race for Automatic Algorithm Configuration. It is used as a package to support the configuration of algorithm parameters automatically. It finds the most suitable settings for a given set of instances when solving an optimisation problem. The package was developed on the F-race package proposed by Balaprakash et al. (2007) and later improved by Birattari et al. (2010). In fact, Irace is a generalization of the iterated F-race procedure which runs in R. As a useful tool for parameter tuning, López-Ibáñez et al. (2016) states Irace as a user-friendly, easy and simple tool to tune parameters even for users who are not familiar with R. In terms of limitations, López-Ibáñez et al. (2016) highlight the importance of enough assignment of computational effort for the configuration tasks in order to result in a better configuration than a random configuration. In general, such automatic configuration methods are hard to use for parameter tuning if problem instances are highly computationally expensive (López-Ibáñez et al., 2016).

According to Birattari et al. (2002) and Birattari (2009), in the majority of the cases, the metaheuristics are tuned using the trial-and-error procedure guided by some rules of thumb. Despite most of the research papers adopted this method, trial-and-error approach presents following drawbacks. It may need the involvement of a skilled designer, usually the programmer who implemented the algorithm to make necessary changes when tuning is required. Also, this method is error-prone and tedious.

Out of these approaches, in this study, we use constructive heuristics to build the solution by placing one piece at a time, investigating the applicability of different placement strategies for pieces and sequencing strategies for bins and pieces. Both the local search and metaheuristic approaches are used to find an improved solution by working over the search. As described in this chapter, this involves the key tasks of identifying the solution space, setting up a mechanism to identify neighbourhoods, generating techniques to move from solution to solution and maintaining the dynamic balance between diversification and intensification of the search process.

3.4 Implementation

All the constructive and improvement algorithms were implemented by coding in Visual C++ 2012 and boost C++ 1.63 environments. For each problem, we experimented with different types of constructive algorithm by testing the different placement and sequence strategies. As an example; we implemented six different construction algorithms for solving the 2D ISBSBBP and compared the performance of them with each other. Similarly, the search algorithms were also implemented and tested to find better results.

Each algorithm was run on one 2.6 GHz CPU (Intel Sandy Bridge) with 4GB memory in the Southampton University Iridis 4 computer environment. No algorithm contained any part of parallel programming.

When conducting the experiments, we ran the experiments for multiple trials and average values were reported if any algorithm contained random components. Once algorithms were implemented, we conducted parameter tuning for each problem. Details of these parameters for each case are included in proceeding chapters.

3.4.1 Instances

The different variants of the algorithm were tested on the instances with irregular shapes available in the literature. In this case, we used the instances published by EURO Special Interest Group on Cutting and Packing (ESICUP) web site (http://paginas.fe.up.pt/ esicup/datasets).

In Table 3.1 we summarise the characteristics of each nesting instance used in solving 2D ISBSBPP in Chapter 4. In order to compare with other published results, we also use two more sets of instances with Jigsaw type pieces from ESICUP website. The first set; denoted as JP1, includes a collection of 540 instances where all the pieces of them are convex. The specifications of those instances are presented in Table 3.2. The 540 instances are divided into 18 classes. The second set; denoted as JP2, includes 480 instances and they are divided into 16 in which some pieces have concavities. The specifications of those 480 instances are presented in Table 3.3.

For the experiments in solving 2D IMBSBPP in Chapter 5, we also involve the nesting instances. Since some of the instances contained less number of pieces to be applied in a bin packing problem (i.e. with the combination of large and small size bins), we make sure that we consider enough number of pieces to have more than one bin in a solution. In order to achieve this, we multiply the number the pieces in those instances so that it can have a sufficient number of pieces to be considered in a bin packing problem with multiple bin sizes. Table 5.1 presents the details of these instances. A similar approach is used when deciding the instances used for the experiments discussed in Chapter 6. The details of these instances and their uses are described in Section 6.4.1.

Table 3.1: Nesting Instances (extracted from Martinez-Sykora et al. (2017))

Instance	Types of Pieces	Number of Pieces	Average of Vertices	Allowed Rotation Angles	Problem Type	Authors
dighe2	10	10	4.7	0	Jigsaw Puzzle	Dighe and Jakiela (1995)
fu	12	12	3.58	0-90-180-270	Artificial, convex	Fujita et al. (1993)
poly1a	15	15	4.6	0-90-180-270	Artificial	Hopper (2000)
dighe1	16	16	3.87	0	Jigsaw Puzzle	Dighe and Jakiela (1995)
mao	9	20	9.22	0-90-180-270	Garment	Bounsaythip and Maouche (1997)
albano	8	24	7.25	0-180	Garment	Albano and Sapuppo (1980)
jakobs1	25	25	5.6	0-90-180-270	Artificial	Jakobs (1996)
jakobs2	25	25	5.36	0-90-180-270	Artificial	Jakobs (1996)
shapes2	7	28	6.29	0-180	Artificial	Oliveira and Ferreira (1993)
poly2a	15	30	4.6	0-90-180-270	Artificial	Hopper (2000)
poly2b	30	30	4.53	0-90-180-270	Artificial	Hopper (2000)
shapes1	4	43	8.75	0-180	Artificial	Oliveira and Ferreira (1993)
shapes0	4	43	8.75	0	Artificial	Oliveira and Ferreira (1993)
poly3a	15	45	4.6	0-90-180-270	Artificial	Hopper (2000)
poly3b	45	45	4.6	0-90-180-270	Artificial	Hopper (2000)
swim	10	48	21.9	0-180	Garment	Oliveira and Ferreira (1993)
poly4a	15	60	4.6	0-90-180-270	Artificial	Hopper (2000)
poly4b	60	60	4.6	0-90-180-270	Artificial	Hopper (2000)
trousers	17	64	5.06	0-180	Garment	Oliveira and Ferreira (1993)
poly5a	15	75	4.6	0-90-180-270	Artificial	Hopper (2000)
poly5b	75	75	4.57	0-90-180-270	Artificial	Hopper (2000)
shirts	8	99	6.63	0-180	Garment	Dowsland et al. (1998)

Table 3.2: Characteristics of the JP1 instances (extracted from Lopez-Camacho et al. (2013b))

Problem Type	Number of Instances	Number of Pieces	Average Piece Size	Piece Size Standard Deviation	Average Rectangularity	Percentage of Right Angles	Percentage of Orthogonal Sides	Average of Concavity Degree	Average of Ratio area/convex hull	Optimal (number of objects)
Convex 2D problem instances										
Minimum			0.033	0.014	0.35	11	34	1	1	2
Average			0.154	0.100	0.68	42	65	1	1	5.94
Maximum			0.354	0.280	1	100	100	1	1	15
			Average of instances per type							
Conv A	30	30	0.100	0.069	0.70	42	68	1	1	3
Conv B	30	30	0.333	0.162	0.87	67	84	1	1	10
Conv C	30	36	0.167	0.124	0.68	36	63	1	1	6
Conv D	30	60	0.050	0.036	0.57	23	51	1	1	3
Conv E	30	60	0.050	0.035	0.41	12	38	1	1	3
Conv F	30	30	0.067	0.050	0.59	29	57	1	1	2
Conv G	30	36	0.332	0.156	0.87	67	83	1	1	Unknown
Conv H	30	36	0.333	0.158	0.86	67	84	1	1	12
Conv I	30	60	0.053	0.017	1	100	100	1	1	3
Conv J	30	60	0.067	0.034	0.83	68	83	1	1	4
Conv K	30	54	0.154	0.150	0.63	34	60	1	1	6
Conv L	30	30	0.100	0.075	0.51	23	50	1	1	3
Conv M	30	40	0.125	0.102	0.55	28	55	1	1	5
Conv N	30	60	0.033	0.024	0.62	32	60	1	1	2
Conv O	30	28	0.250	0.223	0.57	27	58	1	1	7
Conv P	30	56	0.143	0.173	0.49	17	43	1	1	8
Conv Q	30	60	0.250	0.053	0.89	51	76	1	1	15
Conv R	30	54	0.167	0.153	0.63	36	62	1	1	9

Table 3.3: Characteristics of the JP2 instances (extracted from Lopez-Camacho et al. (2013b))

Problem Type	Number of Instances	Number of Pieces	Average Piece Size	Piece Size Standard Deviation	Average Rectangularity	Percentage of Right Angles	Percentage of Orthogonal Sides	Average of Concavity Degree	Average of Ratio area/convex hull	Optimal (number of objects)
Non-Convex 2D problem instances										
Minimum			0.044	0.036	0.38	6	27	1.004	0.834	2
Average			0.160	0.135	0.59	26	50	1.130	0.930	5.9
Maximum			0.333	0.314	0.84	60	74	1.560	0.987	12
			Average of instances per type							
Nconv A	30	35-50	0.074	0.062	0.60	28	52	1.12	0.935	3
Nconv B	30	40-52	0.214	0.158	0.69	38	58	1.22	0.923	10
Nconv C	30	42-60	0.123	0.111	0.59	25	49	1.11	0.939	6
Nconv F	30	35-45	0.051	0.045	0.53	20	46	1.10	0.940	2
Nconv H	30	42-60	0.245	0.163	0.73	46	64	1.15	0.944	12
Nconv L	30	35-45	0.076	0.065	0.47	16	41	1.10	0.941	3
Nconv M	30	45-58	0.099	0.092	0.50	20	46	1.07	0.956	5
Nconv O	30	33-43	0.186	0.190	0.51	19	46	1.10	0.940	7
Nconv S	30	17-20	0.106	0.097	0.45	10	33	1.16	0.918	2
Nconv T	30	30-40	0.293	0.239	0.60	26	51	1.24	0.916	10
Nconv U	30	20-33	0.197	0.161	0.55	17	44	1.19	0.888	5
Nconv V	30	15-18	0.306	0.236	0.62	27	54	1.09	0.936	5
Nconv W	30	24-28	0.155	0.097	0.78	53	69	1.12	0.931	4
Nconv X	30	25-39	0.097	0.072	0.66	32	53	1.17	0.895	3
Nconv Y	30	40-50	0.135	0.129	0.61	25	51	1.09	0.943	6
Nconv Z	30	60	0.200	0.234	0.54	19	45	1.09	0.940	12

Chapter 4

Homogeneous Bin Packing Problems with Irregular Shapes

4.1 Introduction

This chapter is devoted to addressing the Two-Dimensional Irregular Bin Packing Problem with homogeneous bins. Following Wäscher et al. (2007)'s typology, the problem can be also denoted as 2D Irregular Single Bin Size Bin packing Problem (2D ISBSBPP). The problem arises in many industries where there is a requirement for irregular shape pieces to be cut from multiple fixed dimension stock sheets such as metal, wood, paper and plastic. Despite the many real-life applications, the problem has received relatively little attention in the literature when compared to the rectangle bin packing problem or the irregular shape strip packing problem. In this chapter, we describe a heuristic solution method for the 2D ISBSBPP where the stock sheets are homogeneous and a feasible solution is found when all the pieces are packed in the bins with no overlap between pieces. The objective is to minimise the total number of bins needed to cut all of the pieces.

Our approach draws on the literature for irregular strip packing and applies these concepts to bin packing. We describe new features that significantly enhance the performance of the algorithm and make it more generally applicable.

The main components of our approach include:

- A single-pass constructive algorithm that constructs the bins

  The constructive algorithm (CA) is based on TOPOS originally proposed by Oliveira et al. (2000) and later improved by Bennell and Song (2010). Both these implementations were for strip packing problems. In our approach, we propose a method to convert the strip packing problem into the bin packing problem.

- Placement decisions that allow either finite or free rotation of the pieces

  A key component of the placement decision is the orientation of the pieces. Most data instances restrict the orientation of the pieces to one, two or four angles of orientation. However, many types of material are homogeneous and pieces can be cut in any orientation. In this chapter, we develop placement rules that permit pieces to rotate either by a pre-defined finite set of orientation (restricted rotation of pieces) or any angle in the range of 0 to 360 degrees.

- An improvement heuristic to search over the solutions

  The improvement heuristic uses the principles of the Jostle approach proposed by Dowsland et al. (1998) for strip packing. The combination of Jostle and the CA means that the approach solves both the placement and allocation problems together rather than considering them as two independent problems. Finally, further improvements are made to the Jostle heuristic to enhance its ability to search the solution space.

Out of the papers reviewed in Section 2.3, the majority of the existing literature on irregular packing, has achieved efficient solutions with restricted rotation of the irregular pieces e.g. Burke et al. (2007), Bennell and Song (2010), Lopez-Camacho et al. (2013a), while only a few papers (Martinez-Sykora et al., 2015; Rocha et al., 2014; Han et al., 2013; Stoyan et al., 2015a; Martinez-Sykora et al., 2017) consider unrestricted rotation of irregular pieces. To our knowledge, Han et al. (2013), Martinez-Sykora et al. (2015) and Martinez-Sykora et al. (2017) are the only papers that consider unrestricted rotation with multiple identical stock sheets. Out of those the first two studies consider an application of cutting glass where pieces are convex and require guillotine cuts. The guillotine cuts may be at any angle relative to the edge of the stock sheet. This constraint has a dominating influence over the algorithm design and strongly influence the rotation angle of the pieces.

4.2 Contribution

In this section, we briefly outline the contribution of this chapter.

- A new method for identifying promising rotations for the pieces, in the case when any angle is permitted.

- Development of an efficient heuristic adapted from the Jostle approach that is competitive across all nesting benchmark data sets. In addition, we introduce a diversification mechanism from local search to improve the performance of the Jostle approach.

- This study is one of the very early studies to tackle the irregular bin packing problem allowing free rotation of the pieces, which is more general and more realistic for many industries.

4.3 Problem Description

Let $P = \{p_1, \ldots, p_n\}$ be a set of n polygons to be packed into identical rectangular bins. We denote the bins as rectangular objects with fixed width W and fixed length L. A feasible solution of the 2D ISBSBPP is described by a set of bins $B = \{b_1, \ldots, b_N\}$ where N is the number of bins required to pack the given n irregular polygons. We denote $P_j \subseteq P$ as the set of packed pieces in each bin b_j and n_j denotes the number of placed pieces in bin b_j. Each packed piece p_i, has an orientation angle $o_i \in [0, 360^o)$ and a position in its allocated bin relative to the piece origin of (x_i, y_i). We consider the reference point of each piece as the bottom-left corner point of the enclosing rectangle for a given angle of rotation and the bottom left corner of the first bin is the origin.

The objective is to minimise N; the number of bins utilised. However, this measure is not very sensitive and many competing solutions will have the same value of N for a given problem instance. As a result, we use the following two measurements to evaluate the bin packing solutions.

1. Fractional number of bins $K = N - 1 + R^*$ where R^* is the proportion of the least utilised bin used to pack pieces after applying either a vertical or horizontal cut. This assumes that offcuts can be kept and used later. See Han et al. (2013).
2. $F = \frac{\sum_{j=1}^{N} U_j^2}{N}$ where U_j is the utilization of each bin b_j, proposed by Lopez-Camacho et al. (2013a). The utilization of each bin U_j is defined as the total area of pieces inside the bin b_j divided by the are of the bin $(L \times W)$.

Both measures reward solutions that highly utilise bins while trying to empty one weakest bin, which supports the search in reducing the number of bins.

The measurement F is helpful in terms of differentiating solutions during the search process while the algorithm runs its iterations and evaluates its solutions. When there are solutions with an equal number of used bins, the F value favours solutions that have tightly packed bins and a tail of low utilisation bins over those solutions which evenly distributed with pieces among the bins. In this case, encouraging some bins to be poorly utilised makes it easier to reduce the total number of bins in the solution, since only a few items need to be relocated into other bins.

The measurement K measures the fractional number of bins considering the partial bin R^* generated after applying either a vertical or a horizontal cut to the least utilised bin.

This measurement is useful to differentiate solutions with an equal number of used bins and it also takes account of the reuse of residuals of the least utilised bin. Usually, it is advantageous to select the solution with lower K value if number of bins used are equal in the solutions since the lower R^* represents less fractional use of bins and higher saving of residuals which can be reusable.

In terms of using each measurement, K reflects the preference of the manufacturer in minimising material use, while F is better at guiding the search.

4.4 Packing Procedure

A solution method of 2D IBPP addresses two main decision problems: assigning pieces to bins and placing those pieces in the bin. The majority of cutting and packing literature that considers 2D irregular pieces focuses on strip packing where there is a single stock sheet. In this case, the main decision is where to place the piece on the strip with the aim of minimising length. Our approach follows the strip packing approach while imposing the division between bins, therefore making the bin allocation implicit within the placement decisions. The proposed method always leads to a feasible placement of the next piece by addressing the allocation and placement decisions together.

Our approach seeks to simulate the jostling of pieces. Dowsland et al. (1998) used this principle to reduce the packing length of the layout for the irregular shape strip packing problem by shaking the pieces from one end to another. They called the algorithm Jostle and suggested it was inspired by the natural human behaviour to shake a container to encourage items to pack together as tightly as possible. The basic idea is to begin with an initial random ordering of the pieces and apply a constructive heuristic following a placement policy of left most position packing from the outside edges of the strip towards the centre. Once all the pieces are packed, Jostle would proceed to take the pieces in order of largest to smallest x-coordinate and pack at the opposite end the stock sheet i.e. a right most policy, filling holes where possible. The next iteration would take the packing order from smallest to largest x-coordinate and again pack with a left most policy, and so on simulating the jostling of pieces from one end of the stock sheet to the other. Note that packing from left to right and then right to left is called a Jostle cycle. At the time of publishing, the approach generated the best results on the benchmark data sets. Since this work, researchers have developed more effective methods for handling the geometry and utilise more powerful computers. In this study, we apply the principles of Jostle to the homogeneous multiple bin packing problem.

In order to adapt Jostle to the bin packing problem, we first define a very long packing strip of width W and length $n \times L$. This strip represents n bins joined together. We then define bin spaces by assigning a set of barrier lines at fixed intervals of length L along the strip. The bottom left corner of bin $i \in \{1, \ldots, n\}$ is placed at $(L \times (i-1), 0)$

(see Figure 4.1). We can directly apply Jostle to the strip with the additional constraint that pieces may not be placed across a barrier. Along with the usual containment and overlap constraints, this ensures pieces are always placed entirely within the bins.

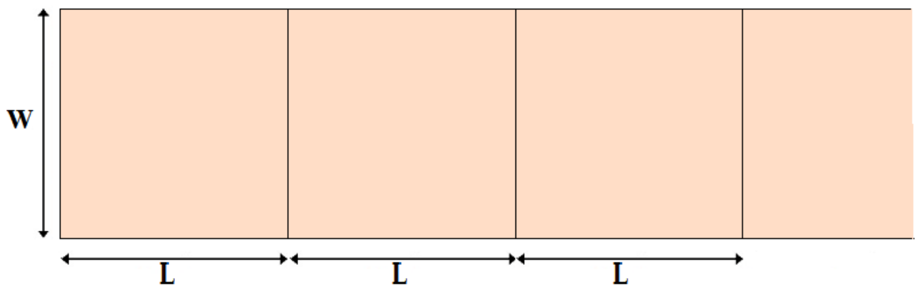


Figure 4.1: Strip to Bins

Dowsland et al. (1998) imposed a grid of feasible placement positions over the strip, which formed a search grid for finding the position of the next piece to be packed. Restricting the placement positions by a grid over constrains the solution space and good solutions can be lost. More recent work in irregular packing has developed effective placement policies based on the no-fit polygon (e.g. Bennell and Song (2010)). As a result, we design several new placement policies that are incorporated within Jostle, these are described in Section 4.4.1. In Section 4.5 we describe the Jostle approach in more detail including some modifications to improve the effectiveness of the search.

4.4.1 Construction algorithm

In this section, we introduce the single-pass construction algorithm, which is made up of the packing order of pieces; and the placement policy. The packing order is predefined for the first layout construction and subsequently decided by Jostle. The placement policy sets the criteria for comparing the candidate placement positions and orientations of the next piece to be placed, where any candidate position must be feasible with respect to overlap and containment in a bin.

Any solution approach to a packing problem involving irregular shapes requires computational tools for handling the geometry. Specifically, search heuristics need an efficient test for identifying feasible placement positions of pieces. Our tool set is based on the no fit polygon (NFP) and inner fit polygon (IFP) obtained using the approach of Bennell and Song (2008). Since pieces are permitted to be placed in any orientation, there are infinitely many possible NFPs, hence it is necessary to generate these on-line as they are needed. In order to place a piece, the algorithm must generate multiple NFPs, one for each of the placed piece with the next piece to be placed. Our investigation shows that it is significantly more efficient to sequentially merge all the placed pieces into a single polygon and then generate a single, albeit large and complex, NFP. This is the approach of Oliveira et al. (2000), called TOPOS, which was later improved by Bennell and Song (2010).

The proposed constructive algorithm represents the existing partial solution as an array of polygons m_j where each m_j is the polygon generated by merging the set of pieces packed within the region b_j (equivalent to a bin). Unlike Oliveira et al. (2000), we fix the position of the partial solution within the bin. As in Bennell and Song (2010), the algorithm retains any enclosed gaps between the pieces when a piece is merged with existing partial solution. These gaps form holes in the merged polygon, which are made available for placement of subsequent pieces, provided the area of the hole is greater than the area of the smallest unpacked piece. Let $H_k : h_1, .., h_k$ be the set of holes generated after placing the kth piece. Figure 4.2, provides an example of three regions, which represent bins and contain a number of packed pieces. The groups of packed pieces are merged into three polygons m_j, which contain usable holes. h_k. The spaces outside of m_j are denoted by S_j.

4.4.1.1 Generating candidate placement positions

In this section, we describe our approach for identifying the feasible candidate positions for the next piece, p_i, including its orientation. The placement policy criteria, described in Section 4.4.2, determines which candidate is chosen. p_i may be placed within a hole, in a space or in a new bin region at the end of the layout. Below we describe the procedure for determining whether p_i will fit in a particular hole or space in a partially packed bin and the range of orientations.

Each hole h_l is represented as a simple polygon. If the area of h_l is less than the area of p_i then p_i will not fit into h_l. Otherwise, the procedure generates the IFP between h_l and p_i ($\text{IFP}_{h_l p_i}$). The interior and boundary of the IFP represent all the relative positions of the two component polygons so that the second subscript polygon, p_i, fits inside the first subscript polygon, h_l. See Figure 4.3. The boundary of the IFP contains all positions where the boundaries of the polygons touch and this forms the set of feasible placement positions within the hole. The exact placement position is determined according to the placement rule described in Section 4.4.2. If the IFP does not exist then p_i will not fit into h_l.

In order to find the feasible placement positions within a bin region that contains polygon m_j, we generate the NFP between the external boundary of m_j and p_i, $\text{NFP}_{m_j p_i}$ and the IFP between the bin region b_j and p_i, $\text{IFP}_{b_j p_i}$. The exterior and boundary of the NFP represents all relative positions of the two component where they do not overlap. The intersection of the $\text{IFP}_{b_j p_i}$ and the complement of $\text{NFP}_{m_j p_i}$ provides all the positions of p_i such that it is contained within b_j and not overlapping m_j, assuming b_j and m_j have a fixed position. We only consider the boundary segment of the $\text{NFP}_{m_j p_i}$ that intersects

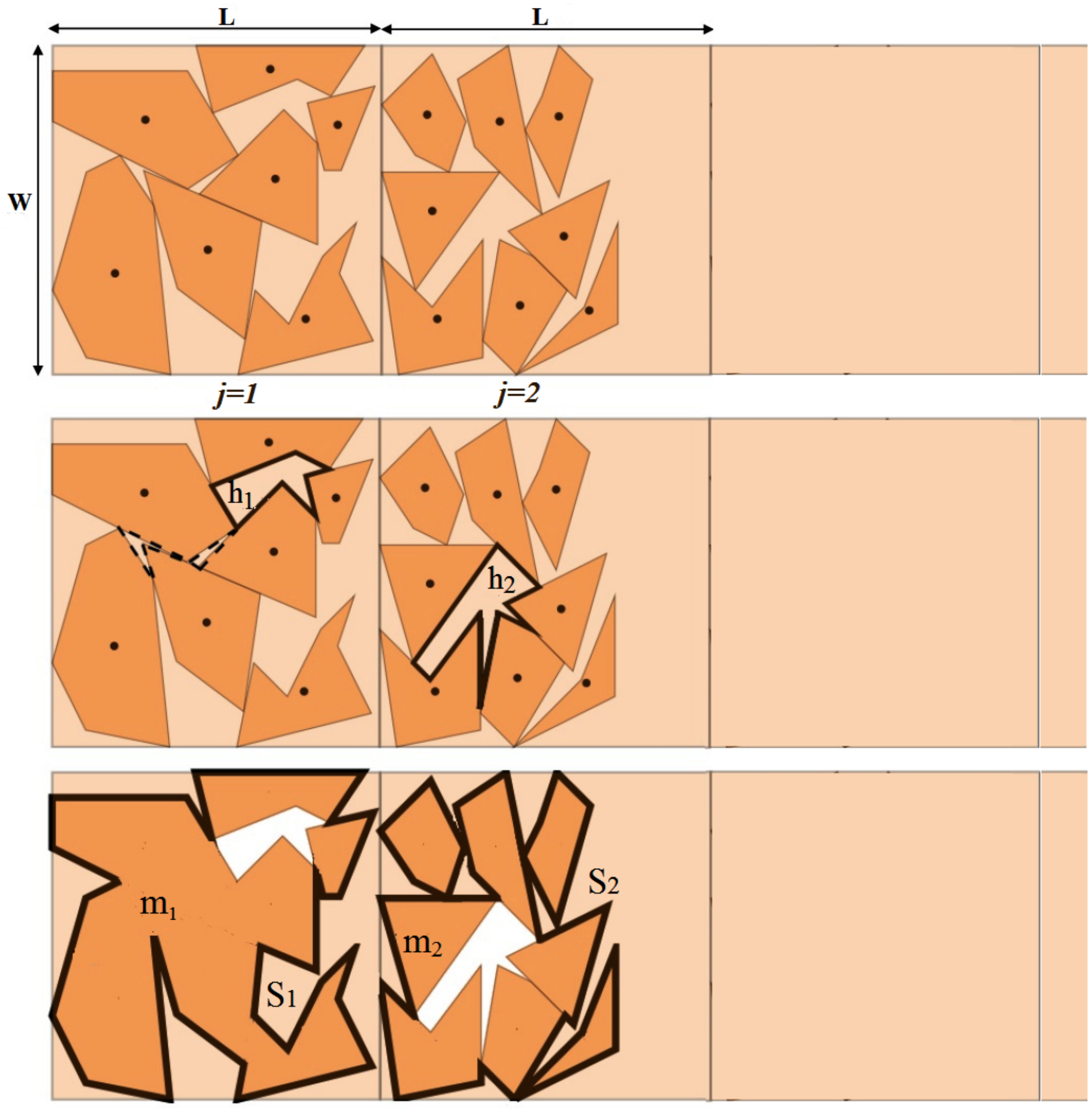


Figure 4.2: Partial solution

with $\text{IFP}_{b_j p_i}$ as the feasible set, which is all the feasible touching positions between m_j and p_i. See Figure 4.4 for an example of this evaluation.

Finally, if we place a piece in a new empty bin, then the position and the orientation is determined by the placement policy (see Section 4.4.2).

4.4.1.2 Placement orientation of a piece

The above discussion assumes p_i is in a given orientation. In this chapter, we consider two scenarios with respect to rotating pieces. First, we allow pieces to rotate by a pre-assigned set of angles, which we define as the finite rotation approach. This is the most common case in the literature where the rotation angles of pieces are restricted to $o_i = 0^o, 180^o$ or $\alpha = 0^o, 90^o, 180^o, 270^o$. Second, we consider rotating pieces freely.

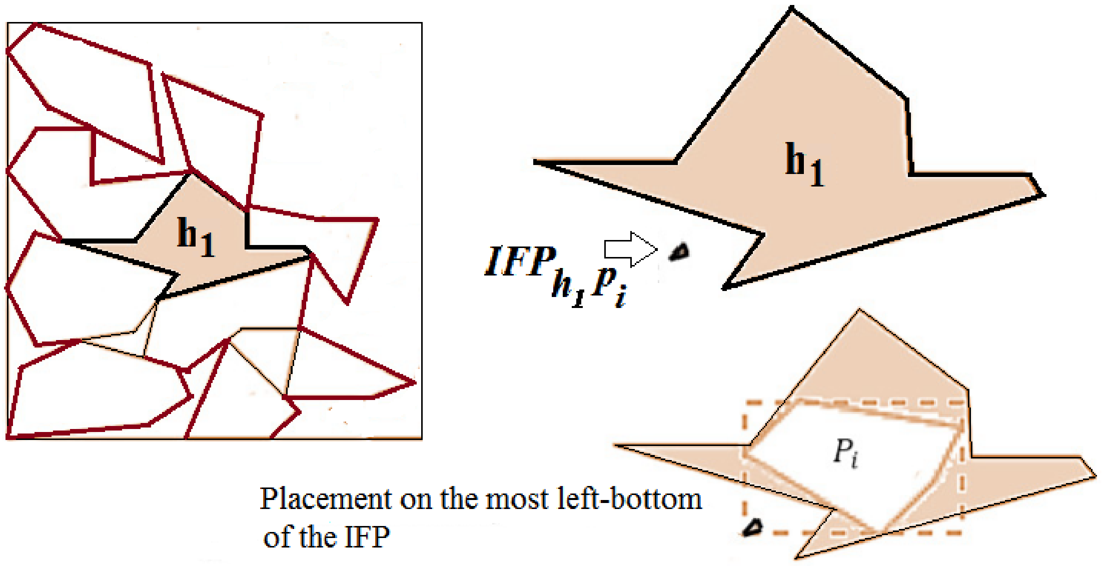


Figure 4.3: Hole-filling

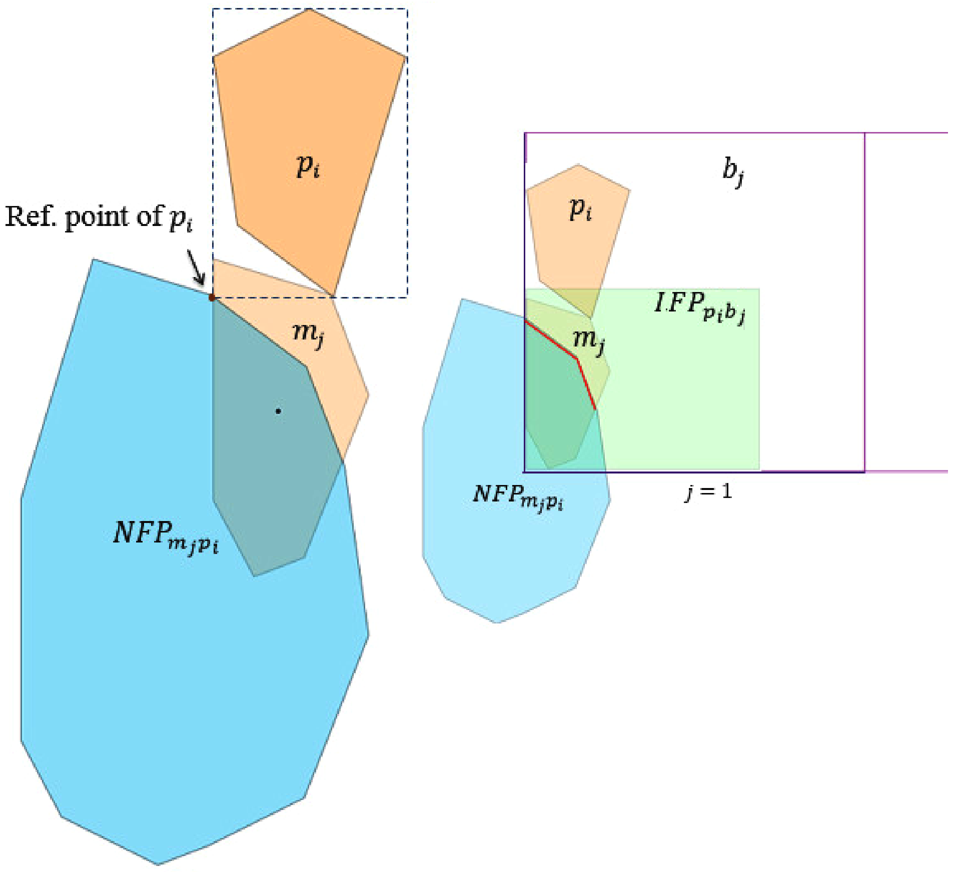


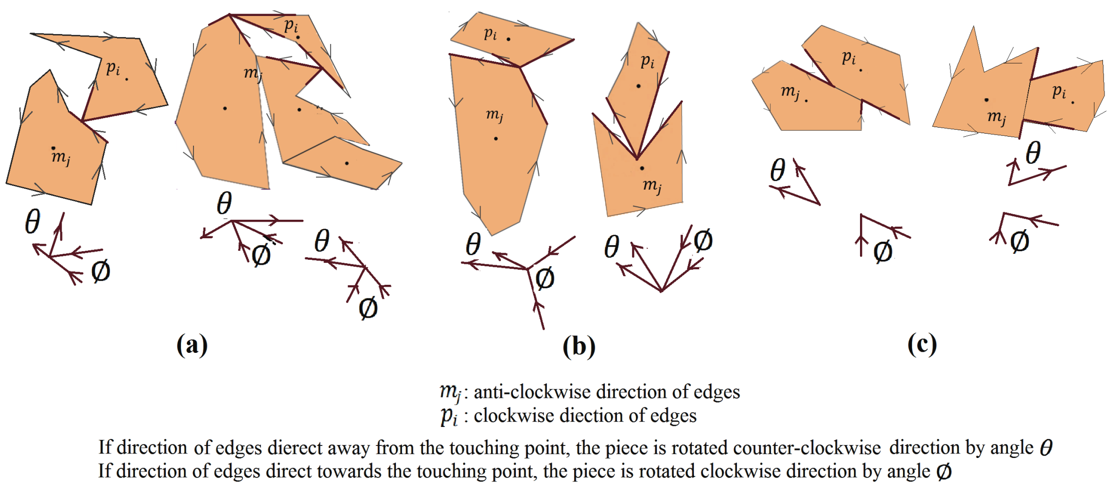
Figure 4.4: Feasible placement for a new piece

In the case of finite rotation angles, we simply consider p_i in each of the four orientations. For each orientation we generate the set of feasible placement positions, as described in 4.4.1.1, and select the best by applying the placement policy, described in Section 4.4.2.

We then compare the best placement for each orientation, using the same placement policy criteria, selecting the best as the final position and orientation of p_i.

For free rotation, there are an infinite set of orientations, which must be reduced to a finite set of candidates. Hence we need a mechanism to identify a subset of promising angles. We first describe the approach assuming p_i can be placed in a region that contains a merged polygon m_j.

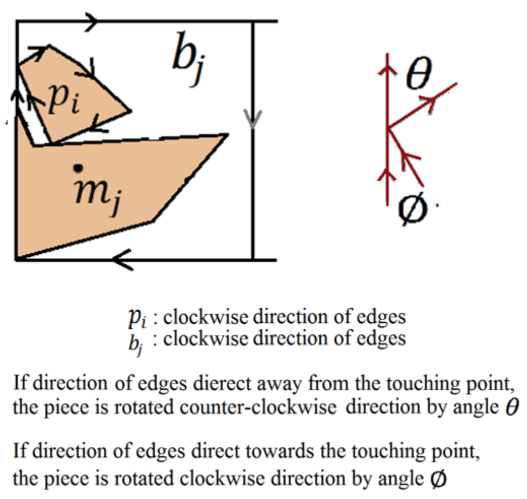
Our approach begins by placing p_i on the packing layout in a temporary position and orientation using the finite rotation placement approach, where the best orientation is selected. As discussed in the previous section, the position will be touching the boundary of m_j. Each touching point or edge defines two new angles of orientation. These are illustrated in figure 4.5.



m_j: anti-clockwise direction of edges
p_i : clockwise diection of edges
If direction of edges dierect away from the touching point, the piece is rotated counter-clockwise direction by angle θ
If direction of edges direct towards the touching point, the piece is rotated clockwise direction by angle $\emptyset$

Figure 4.5: Tuning the orientation angle

Figure 4.5(a) shows an edge-vertex combination, Figure 4.5(b) shows a vertex-vertex combination and figure 4.5(c) shows an edge-edge combination. In each case, piece p is rotated in a counter-clockwise direction by angle θ and in a clockwise direction by angle ϕ. The second example in figure 4.5(a), shows two separate points of contact and as a result there are four new orientations. These form the new candidate orientations and following the placement procedure, we find the best placement position for the pieces in these orientations. These are compared with the temporary position and orientation, if none of these new angles provides a better placement position then the algorithm takes the temporary position and orientation as the placement of the piece. If the next piece touches the boundary of a bin, then we also consider the angles created by the edges of the bin (see Figure 4.6).



p_i : clockwise direction of edges
b_j : clockwise direction of edges

If direction of edges dierect away from the touching point, the piece is rotated counter-clockwise direction by angle θ

If direction of edges direct towards the touching point, the piece is rotated clockwise direction by angle $\emptyset$

Figure 4.6: Angle tuning at the edges of bins

4.4.2 Placement policy

The above procedures provide a set of feasible placement positions and orientations for piece p_i. The placement policy sets the criteria that determine which position is chosen. Note that for some of the placement rules, we can often avoid the computational cost of assessing all positions by controlling the order in which they are assessed. In the following, we describe the placement rules in the context of packing from the left-hand end of the strip extending towards the right. However, Jostle also packs from right to left, in which case simply swap left and right in the following explanation.

We evaluate three well-known placement rules; bottom left placement, minimum length placement and maximum utilisation placement, to determine the position of each piece. However, these rules are dominated by first trying to fill holes in the layout.

The first step when placing p_i is to sort all the holes in H_k in order of leftmost x-coordinate. These holes are evaluated in that order using the hole filling procedure described in Section 4.4.1.1. If p_i fits in a given hole, it is placed and no further holes are considered. If p_i does not fit within any hole, the algorithm attempts to find a feasible placement position in each of the bin regions b_j starting with the left most. Using the procedure described in Section 4.4.1.1 for finding placement position in a partially packed bin, the candidate positions are found for each bin region b_j in order. Depending on the placement policy, once a feasible position is found, the search stops after identifying the best orientation for that position.

The placement policies are as follows:

- Bottom-left (BL) prioritises placing the reference point of the piece in the leftmost feasible position on the stock sheet breaking ties by selecting the bottom-most of the leftmost positions. This is a well-known single pass placement rule applied in

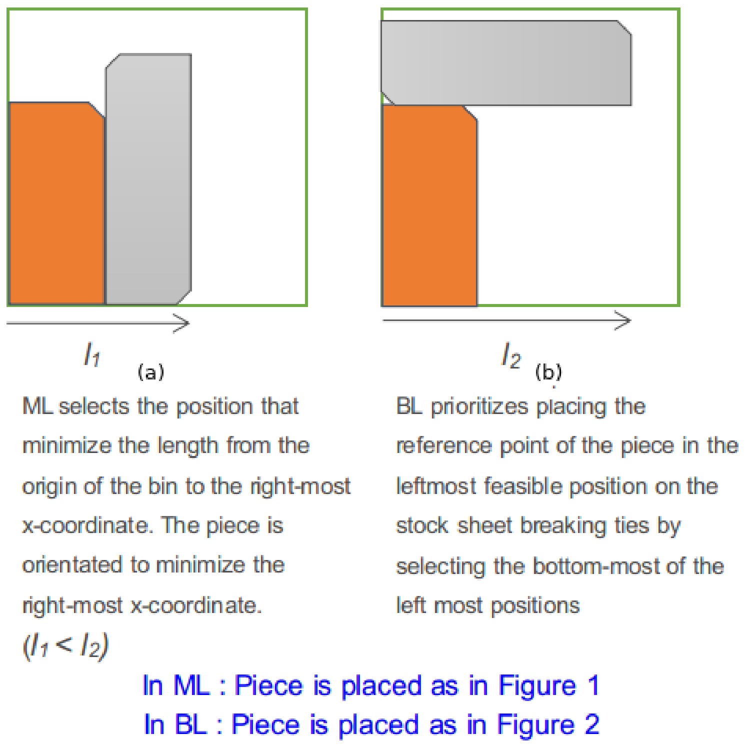

l_1
(a)
ML selects the position that minimize the length from the origin of the bin to the right-most x-coordinate. The piece is orientated to minimize the right-most x-coordinate.
$(l_1 < l_2)$
l_2
(b)
BL prioritizes placing the reference point of the piece in the leftmost feasible position on the stock sheet breaking ties by selecting the bottom-most of the left most positions
In ML : Piece is placed as in Figure 1
In BL : Piece is placed as in Figure 2

Figure 4.7: BL and ML placement

many cutting and packing papers (Albano and Sapuppo, 1980; Błażewicz et al., 1993; Dowsland et al., 1998). By applying this rule when hole filling fails, once a feasible position is found in the region b_j the search can stop. If more than one orientation of p_i gives the same BL position, select the orientation that minimises the width of the bounding box of the piece. If there is still a tie, select the orientation randomly. Apply the same tie breaker rules for placing p_i in a new bin.

- Minimum Length (ML) placement rule selects the position that minimizes the length from the origin of the bin to the right-most x-coordinate of p_i. The piece is orientated to minimise the right-most x-coordinate. In the case of ties, we select the bottom-left placement position.

- Maximum Utilisation (MU) placement rule selects the position that provides the maximum area utilisation in the earliest bin. We assume the occupied area of a region b_j is the area of the convex hull of m_j. Let the area of the convex hull of m_j be CHarea(m_j), the area of p_i be area(p_i) and the area of the convex hull of the newly merged polygon after p_i is placed in a certain position be CHarea $(m_j + p_i)$. Then the utilisation of the placement position (Utp) is calculated as follows.

$$Utp = \frac{\text{CHarea}(m_j) + \text{area}(p_i)}{\text{CHarea}(m_j + p_i)}$$

Given the set of feasible placement positions on the boundary of the NFP, we select each feasible vertex of the NFP as a candidate placement position. This is illustrated in Figure 4.8. Placement positions are evaluated bin by bin starting at the left-most bin. If a feasible placement is found in a bin then no further bins are considered. Ties are broken by selecting the bottom left position and minimum length orientation.

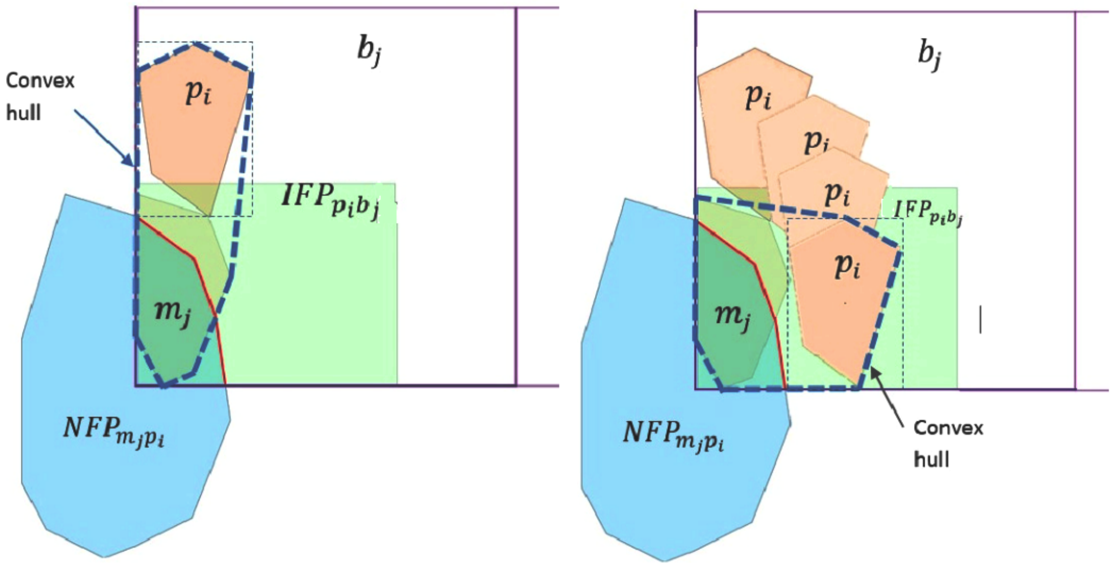


Figure 4.8: Placement of piece on different vertices along the feasible placement segment

Algorithm 3 denotes the constructive algorithm discuss in this chapter.

Algorithm 3: Constructive Algorithm

```
Input  : $P_0$, Placement rule
Output: $P^*$: A feasible packing of all pieces in $N$ Bins
Set $N = 1$;
for $i = 1 : n$ do
    Set $l = 1$, $placed = false$, $j = 1$;
    while $l \leq q$ and $placed = false$ do
        if $p_i$ fits in $h_l$ with any rotation in $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$) then
            Place $p_i$ with the best angle derived from the Placement rule used ;
            Update $H_{i-1} \rightarrow H_i$ and Update $q$;
            Sort all the holes in $H_i$ in order of left most x-coordinate;
            $placed = true$;
        else
            $l = l + 1$;
        end
    end
    while $placed = false$ do
        if $p_i$ fits in $b_j$ with any rotation in $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$) then
            Compute best angle $\alpha$ in $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$ according to the Placement rule used;
            Compute new angles by angle tuning mechanism starting with $\alpha$ denoted as $\theta$ and
              $\phi$; Place $p_i$ in $b_j$ with the best angle between $\alpha$, $\theta$ and $\phi$ according to the
              Placement rule used;
            Merge $p_i$ to $m_j$;
            Capture any usable hole created (area of the hole $\geq$ area of the smallest piece);
            Update $H_{i-1} \rightarrow H_i$ and Update $q$;
            Sort holes $H_i$ by left most x-coordinate;
            If $(j > N)$ $N = j$ ;
            $placed = true$ ;
        else
            $j = j + 1$;
        end
    end
end
```

4.5 Solution Improvement

4.5.1 Jostle heuristic

In this section, we discuss the design of the Jostle heuristic to address the 2D ISBSBPP. The Jostle heuristic is used to determine the piece sequence that feeds into each iteration of the constructive algorithm. The first application of the constructive algorithm uses an arbitrary order of pieces and packs the pieces from left to right along the strip. After generating a complete solution, all the pieces are re-ordered according to the piece with the right-most x-coordinates continuing to the left-most. Following this order, pieces are packed starting at the right-most position of the strip building the packing layout from right to left. Jostle continues to oscillate from packing left to right then right to left until the termination criterion is met.

Jostle is analogous to a local search where the search occurs over the sequence of the pieces and each Jostle iteration generates a neighbour sequence. The constructive algorithm takes a sequence of pieces and creates a layout. The layout is converted back to a sequence according to the x-coordinate order. Hence, the new layout defines a new sequence of pieces, this is a neighbour solution based on the arrangement of pieces of the previous packing layout. Each iteration generates only one neighbour, which is always accepted. The optimisation arises from the neighbour generation. Instead of applying a fixed rule to the sequence i.e. swap or insert, the constructive algorithm changes the sequence by optimising the packing arising from the current piece sequence. Each complete solution is evaluated according to K and F defined in Section 4.3. At the end of the search, the best solution according to these measures is kept.

Since the constructive algorithm includes filling of holes and multiple alternative orientations, the piece sequence and placement positions change between iterations. However, our experiments show that the search can converge and get stuck in local optima. In order to overcome this, we incorporate the diversification ideas from iterated local search to escape from local optima. The new variant is called Iterated Jostle.

4.5.2 Iterated Jostle

The Iterated Jostle algorithm combines the basic Jostle procedure and a kick intended to move the solution to a new area of the solution space and escape local optima. This feature is motivated by our observation that the Jostle heuristic becomes trapped in local optima after running several Jostle cycles. This is more commonly found for data sets that are small and have similar size pieces. We perform a kick from the best local optimum found so far. We consider that we are at a local optima after c_{max} Jostle cycles with no improvement. We evaluate alternative values for c_{max} in the results section. We investigate two kick strategies, described below. Note that in cutting and packing, a small change in the permutations can lead to a significant change in the layout of the pieces once it has been decoded by the constructive algorithm. In the case of their first kick, we check that the move does not produce an identical permutation, which can occur when there are multiple pieces of the same type,

1-piece insert move: We perform a single insert move as follows:

1. Select two random positions along the permutation of pieces,
2. Select the piece corresponds to the highest position selected in step 1,
3. Remove the selected piece from the permutation and insert it in front of the other position selected in step 1,
4. Order the remaining pieces following the existing permutation.

Bin-swapping move: This strategy changes the order of pieces by effectively swapping two bins as follows:

1. Select the bin with the lowest utilisation and let sub-sequence A be the pieces in that bin in x-co-ordinate order,
2. Select another bin randomly and let sub-sequence B be the pieces in that bin in x-coordinate order,
3. Swap sub-sequence A and sub-sequence B in the permutation,
4. Obtain the new permutation of pieces.

Algorithm 4 describes the iterated Jostle procedure; IJS. Let t be the Jostle cycle, P_t^L be the permutation of pieces to be packed from left to right in cycle t, P_t^R be the permutation of pieces to be packed from right to left in cycle t and P_0 be the initial random permutation. P^* stores the permutation of pieces of the solution with the smallest objective function value found so far, K^*. The variable c counts the number of consecutive Jostle iterations with no improvement in solution quality (K). We set c_{max} as the maximum value of c. The termination criterion is the maximum total number of Jostle cycles that the algorithm will run, T_{max}.

Iterated Jostle with join and release of pieces This final version of Iterated Jostle draws on the ideas of Adamowicz and Albano (1972), which was more recently applied by Elkeran (2013). They combine pieces that fit together well reducing the overall number of pieces. In our approach we allow two pieces to be combined into one merged piece for some of the Jostle iterations. The algorithm scans the arrangement of pieces in each bin and attempts to join adjacent pieces in their current position. If the combined piece is convex with no holes then they are merged for the following Jostle iterations. Each scan only permits pairwise merging. However, scans in later iterations may join a piece to a merged piece resulting in three or more original pieces joining together. Pieces are joined after each Jostle iteration until a kick is performed, at which time they are separated again. The type of kick alternates between 1-piece insert and bin-swapping.

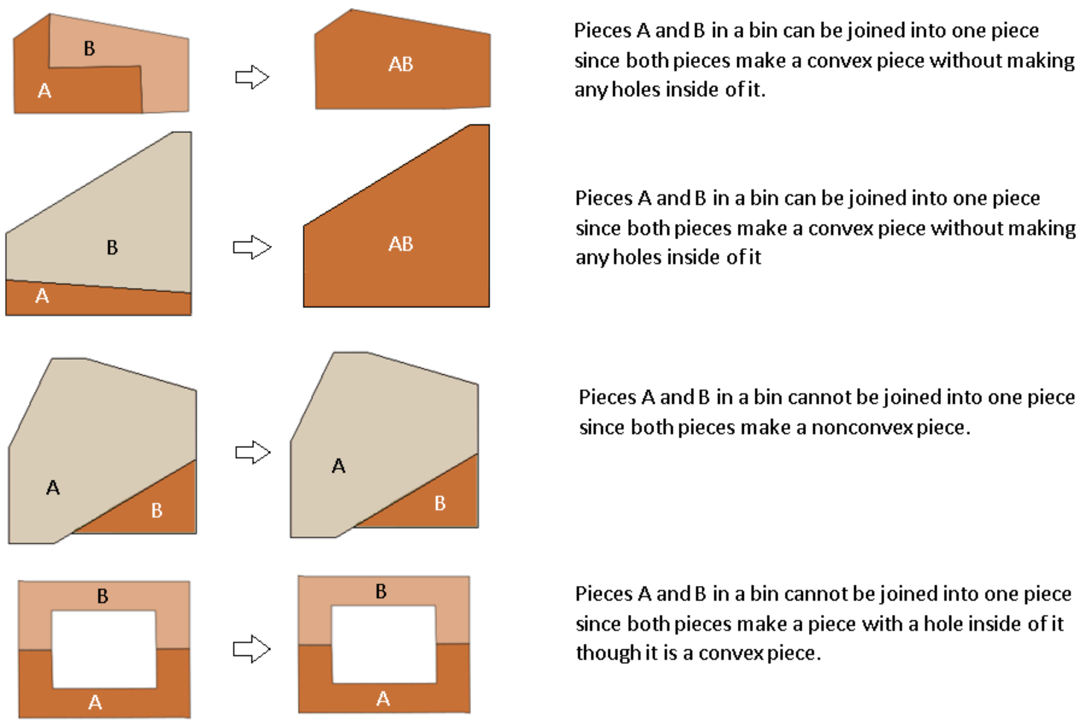
The following steps outline the IJS with join and release procedure.

1. From a given initial permutation, apply Jostle until the kick criterion is met.
2. Apply a kick alternating between bin-swap and 1-piece insert,
3. Jostle for one iteration
4. Execute pairwise joining of pieces for the existing layout. In this case the algorithm scans the whole layout from left to right and searches for the possible pairs of pieces to join as described in Figure 4.9.
5. If stop criteria met then stop else

Algorithm 4: IJS1

```
Input  : $P_0$, Placement rule, kick type $c_{max}$, $T_{max}$, $D^* = Left(L)\ or\ Right(R)$
Output: $P^*$, $K^*$,
Set $t = 1$;
$P^* = P_t^L = P_0$;
Set $D^* = L$;
Set $K^* =$ a suitably large value;
Initialize $c = 0$;
while $t \leq T_{max}$ do
    Generate solution layout from $P_t^L$;
    Evaluate the solution and find $K_t^L$;
    Derive $P_t^R$ from solution;
    Generate solution layout from $P_t^R$;
    Evaluate the solution and find $K_t^R$;
    if $K_t^L < K_t^R$ then
        if $K_t^L < K^*$ then
            Set $K^* = K_t^L$, $P^* = P_t^L$, $D^* = L$;
            Reset $c = 0$;
        else
            $c = c + 1$;
        end
    else
        if $K_t^R < K^*$ then
            Set $K^* = K_t^R$, $P^* = P_t^R$, $D^* = R$;
            Reset $c = 0$;
        else
            $c = c + 1$;
        end
    end
    if $c > c_{max}$ then
        Apply kick to $P^*$ and obtain $P_t^{D*}$;
        Generate solution layout from $P_t^{D*}$;
        Derive $P_t^L$ from solution;
    end
    $t = t + 1$;
end
return $P^*$, $K^*$, $D^*$;
```

6. If kick criteria met, release all joined pieces and go to step 2, else
7. go to step 3



Pieces A and B in a bin can be joined into one piece since both pieces make a convex piece without making any holes inside of it.

Pieces A and B in a bin can be joined into one piece since both pieces make a convex piece without making any holes inside of it

Pieces A and B in a bin cannot be joined into one piece since both pieces make a nonconvex piece.

Pieces A and B in a bin cannot be joined into one piece since both pieces make a piece with a hole inside of it though it is a convex piece.

Figure 4.9: Possibilities when pieces are paired by the Join and Release mechanism

4.6 Experimental Design

All the algorithms were coded in C++11 and utilised version 1.58.0 of the 'boost computational geometry library' to implement some of the geometric operations. The tests were performed by 2.6 GHz processor and 4 GB of RAM to execute the program.

4.6.1 Data instances

Our experiments utilise two sets of benchmark instances: the nesting instances used by a wide variety of authors for the irregular strip packing problem and instances proposed by Lopez-Camacho et al. (2013a) that are jigsaw puzzle instances as they have known optimal solutions of 100% utilisation of the stock sheet. Both are available from the ESICUP (EURO Special Interest Group on Cutting and Packing) website `http://paginas.fe.up.pt/~esicup/datasets`.

The nesting instances contain 23 irregular strip packing data sets and contain both convex and non-convex pieces. As these are strip packing instances we need to define bin sizes for each data set. For each instance, we define three different bin sizes based on the maximum dimension of the enclosing rectangles of each piece in their original orientation denoted as m^d. The bin sizes are: small bin ($1.1m^d$ for length and width), medium bin (1.5m^d for length and width) and large bin (2m^d for length and width). These bin sizes were proposed by Martinez-Sykora et al. (2017).

The jigsaw puzzle (JP) instances contain two sets, which we denote as JP1 and JP2. JP1 has a collection of 540 instances in which all the pieces are convex. These instances are divided into 18 classes with 30 problems and each class carries a different number of pieces. The second set, JP2, has 480 instances (16 classes with 30 problems in each class) in which the pieces have concavities.

4.6.2 Performance of different solution strategies

Our solution strategy includes a number of alternative approaches in terms of the placement policy in the constructive heuristic, the permitted orientations and the kick strategies in Jostle. Table 4.1 and Table 4.2 provide a summary and the notation for the alternative approaches. Any variant that includes a random input is repeated for 10 trials and we report the average result. We conduct the following set of computational experiments to compare the performance of each approach.

Table 4.1: Notation of Algorithms: placement rule and rotation

Notation	Placement Rule	Piece orientation
BL4	Bottom-Left	Restricted $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$
BL∞	Bottom-Left	Unrestricted
ML4	Minimum-Length	Restricted $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$
ML∞	Minimum-Length	Unrestricted
MU4	Maximum Utilization	Restricted $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$
MU∞	Maximum Utilization	Unrestricted

Table 4.2: Notation of Algorithms: algorithm and kick type

Notation	Description	Kick-move strategy
JS	Jostle	-
IJS1	Iterated Jostle	1-piece insert move
IJS2	Iterated Jostle	Bin swapping move
IJS3	Iterated Jostle	1-piece insert move , Bin swapping move , Join and Release of pieces

Test No. 1: Evaluation of the constructive algorithm and the proposed Jostle heuristic.

These experiments compare the performance of the two rotation criteria in terms of quality of packing and computational time and the alternative placement policies. A trial is made up of 500 random permutations of pieces, each packed using a constructive heuristic. The best layout is recorded from each trial. For each constructive heuristic, there are ten trials and we report the average of the ten trials in Table 4.3. We compare constructive algorithm variants $BL4$, $BL\infty$, $ML4$, $ML\infty$, $MU4$, and $MU\infty$. In addition, we evaluate the Jostle heuristic and compare with packing random permutations of pieces using the constructive algorithm. These experiments intend to investigate whether the Jostle heuristic provides a good search mechanism that improves on random trials. We report the average of ten random permutations initialise ten trails of the basic Jostle heuristic where each trial is run for 250 Jostle cycles (which generates 500 layouts). Table 4.3 details the results and considers measure K and F. Note we wish to minimise K and maximise F.

This is a preliminary test to validate that Jostle is more effective than the traditional multi-start approach. We test it using the nesting instances which are generally used as the standard set of instances in 2D irregular shape packing problems, rather being biased more on the specific category of instances like jig-saw puzzle. Table 4.3 contains results from instances with a substantial variety of pieces representing convex pieces, non-convex pieces, pieces with a higher number of edges and jig-saw puzzle type pieces.

Test No. 2: Comparative performance analysis of IJS and JS
We report the results for four strategies; i.e. JS, IJS1, IJS2 and IJS3. Ten random permutations are used as initial permutations for each variant and each trial runs for 500 Jostle cycles. The average is reported in Table 4.2. These experiments evaluate the impact of introducing a kick and compare the types of kick. After initial experiments with a range of values for c_{max}, we set $c_{max} = 4, 6, 6$ Jostle cycles for IJS1, IJS2 and IJS3 respectively.

Test No. 3: Comparative analysis of the performance of IJS algorithms with literature best.

In this case we compare our results with the work of Lopez-Camacho et al. (2013a) and Martinez-Sykora et al. (2017). Since these algorithms are run on different computing platforms, it is not possible to run for the same amount of time. Hence we retain our termination criteria of 500 Jostle cycles and report the average result and computational time for the ten random trials.

4.7 Experimental Results

4.7.1 Test 01: Jostle search vs. Random order of pieces

A key component of the algorithms is the level of freedom to rotate the pieces. We evaluated the benefits of different rotation criteria; 1) finite angles of rotation and 2) free rotation, for building packing solution. The experimental results in Table 4.3 show that allowing free rotation of pieces can significantly improve the solution quality. Considering a pairwise comparison of four angles of rotation and free rotation for each placement rule (BL, ML, MU), 16 of 18 comparisons produced better solutions using free rotation for both K and F, and the remaining two comparisons showed free rotation is better for measure K but not for F.

Considering the performance of Jostle against the randomly generated permutations, the experimental results are shown in Table 4.3 reveal that the Jostle heuristic performs

Table 4.3: Performance of construction algorithm for different rotation criteria

	Random order						Jostle					
	Small bins		Medium bins		Large bins		Small bins		Medium bins		Large bins	
	K	F	K	F	K	F	K	F	K	F	K	F
$BL4$	9.91	0.379	4.792	0.389	2.601	0.395	9.61	0.387	4.712	0.392	2.602	0.398
$BL\infty$	9.73	0.383	4.632	0.394	2.555	0.392	9.48	0.389	4.612	0.397	2.487	0.399
$ML4$	9.94	0.378	4.794	0.390	2.613	0.396	9.52	0.382	4.703	0.394	2.551	0.404
$ML\infty$	9.76	0.384	4.612	0.393	2.551	0.392	9.42	0.389	4.511	0.398	2.464	0.407
$MU4$	9.89	0.383	4.944	0.385	2.592	0.391	9.65	0.387	4.786	0.402	2.561	0.401
$MU\infty$	9.62	0.385	4.614	0.393	2.524	0.392	9.47	0.402	4.573	0.408	2.470	0.413

better for all pairwise comparisons between placement rules and rotation criteria. The best solutions are produced when pieces are allowed to rotate freely. Since there is no significant difference between the computational time in both cases, we can also conclude that the time consumption of the Jostle heuristic is negligible and time is mainly spent on the execution of the construction algorithm. The results are aggregating over many problem instances. A detailed look at the instances reveals that often the number of bins are reduced and in other cases the piece are packed more tightly across the same number of bins. We also observed that Jostle struggles to diversify from the first local optima, so the improvements are local with respect to the initial solution. With this in mind, we conclude that Jostle can perform as an optimisation procedure with more intelligence than a random search.

The final observation from these results concerns the placement rule. For random permutations BL performs the best once on measuring F, ML performs the best twice and MU performs the best three times. For Jostle the result is more consistent where ML performs the best for each measure of K and MU performs the best for each measure of F. This result is intuitive since K values minimising length to provide a greater residual in the last bin, while F is based on utilisation. It is interesting to note the BL is consistently adopted in the literature, but in our investigation, it is consistently inferior.

4.7.2 Test 02: IJS vs JS

Though Jostle yields better solutions than packing random permutations of pieces, one major limitation is the tendency to get trapped in local optima. This occurs more frequently with small data sets and with those with less irregularity in their shapes. As an example, Figure 4.10(a) illustrates solution F values received at each iteration for a certain trial for Fu data instance using JS. In this case, the Jostle heuristic has taken 9-12 cycles to reach for a better F value (i.e. $F = 0.488$). When jostling continues, the solution rarely reaches $F = 0.488$ and the best value is recorded at the 143rd cycle. After the 176th cycle, the Jostle has reached a steady state generating the same set of solutions repetitively.

Figure 4.10(b) demonstrates the benefit of using a kick. Here IJS1 is run on the same data. Introducing a kick has avoided the repetitive cycle and produced better solutions (i.e. $F > 0.499$) within fewer iterations.

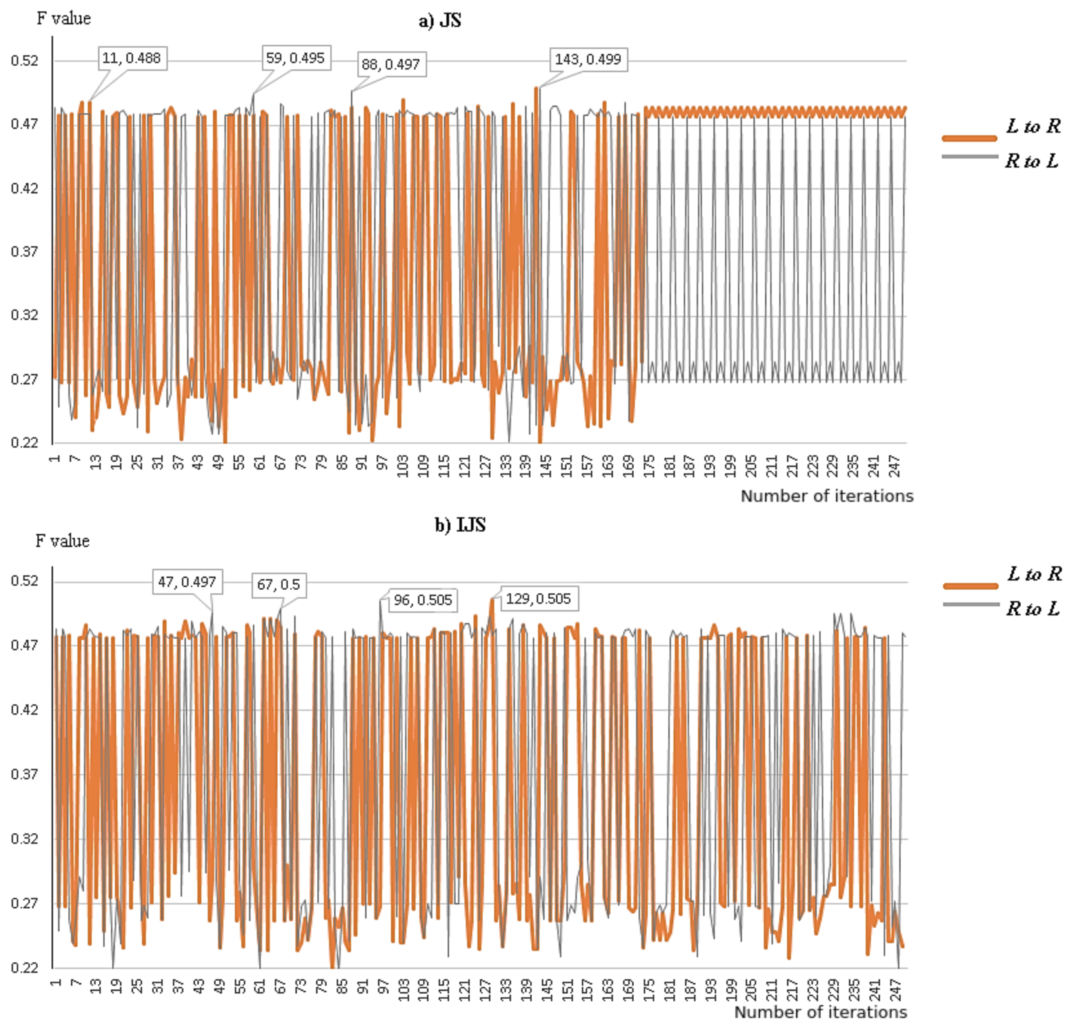


Figure 4.10: JS and IJS1 iterations

In Table 4.4 we present the average results obtained by JS and the three IJS procedures for the 23 Nesting instances. For all placement criteria and angles of rotation, the three IJS algorithms produce better results when compared with JS. For nesting data instances, IJS1 provides the best results for all bin sizes when averaging over all placement/rotation variants and, with one exception for medium bins, IJS1 contains the overall best result. Focusing just on the IJS algorithms, there are 18 variants (3 placement rules $\times$ 2 angles of rotation $\times$ 3 IJS algorithms). Out of those 18 methods, IJS1-MU∞ produces the best results irrespective of the bin sizes.

In Table 4.5 we present the average results obtained by JS and the three IJS procedures for the 540 JP1 and 480 JP2 instances. Again the IJS approaches are clearly better than the values of JS. On average IJS3 performed better for both JP1 and JP2 instances. However, in this case, IJS3-MU4 has produced the best results. For this data the optimal

Table 4.4: Performance results JS, IJS1, IJS2 and IJS3 for nesting data instances

		JS		IJS1		IJS2		IJS3	
		F	T (sec)	F	T (sec)	F	T (sec)	F	T (sec)
Small bin	BL4	0.384	304.4	0.394	322.4	0.388	310.7	0.394	319.4
	BL∞	0.389	502.7	0.408	528.5	0.394	512.1	0.404	534.8
	ML4	0.386	278.6	0.393	313.4	0.390	285.8	0.397	320.5
	ML∞	0.389	509.1	0.419	526.6	0.392	533.4	0.402	575.3
	MU4	0.388	281.4	0.393	294.0	0.389	286.4	0.403	335.0
	MU∞	0.407	504.9	**0.437**	518.6	0.427	534.4	0.433	540.5
Avg.		0.391		**0.408**		0.397		0.406	
Medium bin	BL4	0.392	287.5	0.403	298.4	0.409	298.1	0.407	315.9
	BL∞	0.398	484.7	0.409	506.1	0.403	496.3	0.404	505.4
	ML4	0.394	269.5	0.409	301.8	0.400	281.2	0.402	306.2
	ML∞	0.398	476.6	0.422	515.9	0.418	488.7	0.414	530.6
	MU4	0.404	267.4	0.418	281.2	0.410	272.0	0.409	306.8
	MU∞	0.408	487.9	**0.436**	499.0	0.415	503.4	0.429	510.1
Avg.		0.399		**0.416**		0.409		0.410	
Large bin	BL4	0.398	248.9	0.403	266.5	0.402	255.6	0.410	280.7
	BL∞	0.400	421.6	0.404	450.1	0.402	430.7	0.415	485.3
	ML4	0.404	239.4	0.422	259.7	0.400	249.4	0.410	272.6
	ML∞	0.408	426.3	0.425	451.3	0.409	441.6	0.415	477.6
	MU4	0.402	240.4	0.425	253.9	0.404	247.1	0.410	268.9
	MU∞	0.414	426.8	**0.431**	453.9	0.413	436.8	0.407	480.7
Avg.		0.404		**0.418**		0.405		0.411	

Table 4.5: Performance results of IJS1, IJS2 and IJS3 vs JS for JP1 and JP2 instances

F values		JS		IJS1		IJS2		IJS3	
		F	T (sec)	F	T (sec)	F	T (sec)	F	T (sec)
JP1	BL4	0.589	76.5	0.652	81.2	0.648	76.3	**0.700**	76.2
	BL∞	0.527	148.5	0.564	161.4	0.579	167.4	0.594	153.8
	ML4	0.508	72.7	0.625	79.6	0.647	80.8	**0.693**	75.8
	ML∞	0.522	150.3	0.634	161.9	0.564	152.0	0.590	162.4
	MU4	0.566	72.1	**0.691**	78.1	0.678	79.6	**0.732**	74.9
	MU∞	0.552	159.7	0.677	163.8	0.639	166.7	0.644	161.2
Avg.		0.544		0.641		0.626		0.659	
JP2	BL4	0.572	56.3	0.652	59.3	0.682	62.3	0.725	63.8
	BL∞	0.506	98.4	0.522	101.2	0.532	102.4	0.547	103.3
	ML4	0.549	55.7	0.677	61.3	0.697	61.2	0.714	61.5
	ML∞	0.505	98.1	0.632	103.4	0.607	104.6	0.516	100.1
	MU4	0.617	54.9	0.701	62.1	0.680	59.2	**0.747**	58.3
	MU∞	0.584	97.8	0.655	107.8	0.626	112.5	0.613	101.5
Avg.		0.556		0.640		0.637		0.644	

solution can only be found at the zero rotation angle, hence it is intuitive that restricting the rotation is beneficial.

4.7.3 Test 03: Comparing with benchmark results in literature

Referring to the previous tables 4.4 and 4.5 we can compare these results with the results from the literature. Martinez-Sykora et al. (2017) report a best value of $F = 0.723$ with computational time 161 seconds for JP1 and $F = 0.729$ with computational time 123 seconds for JP2. In both cases our algorithm (IJS3-MU4) provides better results, specifically $F = 0.732$ with computational time 74.9 seconds for JP1 and $F = 0.747$ with computational time 58.3 seconds for JP2. Also, our approach has achieved better results for 11 out of 18 groups of instances for JP1. In the case of JP2 instances, our method has achieved better results for 11 out of 16 groups of instances. A detail description of these results are provided in table 4.6 and 4.7. The only other existing results for this data is by Lopez-Camacho et al. (2013a), who only report results for JP1. They achieve $F = 0.690$ with computational time 50 seconds. We revised the termination of our algorithm to 50 seconds and achieved the improved result of $F = 0.703$.

For the nesting instances, we consider the average over all the instances for each bin size. The only benchmark results are from Martinez-Sykora et al. (2017). For small bins, they achieve $F = 0.434$ with a computation time of 1721 seconds, for medium bins they achieve $F = 0.452$ with a computation time of 3172 seconds and for large bins they achieve $F = 0.403$ with a computation time of 762 seconds. IJS1-MU∞ performs better for small and large bins and takes less time for all three bin types. The overall results of Martinez-Sykora et al. (2017) demonstrate that their algorithm has lower performance when the average number of pieces per bin is higher. However in our case, we achieve significantly better results for large bins in most of our approaches; i.e. IJS1, IJS2 and IJS3. Even for small and medium sized bins, we achieved better or close results with no significant drop in average F values, with less computational time than Martinez-Sykora et al. (2017). Table 4.8, 4.9 and 4.10 present the results we achieved for each data instance using IJS1 approach.

Table 4.6: Results of the IJS1, IJS2 and IJS3 algorithm applied for JP1 instances

BVL: Best values of Lopez-Camacho et al. (2013a), BVM: Best values of Martinez-Sykora et al. (2017)

	BVL(F)	BVM(F)	IJS1 (F values)						IJS2 (F values)						IJS3 (F values)					
			BL4	BL∞	ML4	ML∞	MU4	MU∞	BL4	BL∞	ML4	ML∞	MU4	MU∞	BL4	BL∞	ML4	ML∞	MU4	MU∞
A	0.605	0.614	0.609	0.583	0.606	0.607	**0.614**	0.606	0.604	0.575	0.606	0.582	**0.614**	0.601	**0.620**	0.575	**0.624**	0.595	**0.632**	0.597
B	0.929	0.878	0.770	0.675	0.744	0.795	0.890	0.801	0.675	0.709	0.745	0.712	0.830	0.796	0.827	0.739	0.817	0.724	0.880	0.812
C	0.763	0.736	0.708	0.584	0.645	0.649	0.732	0.725	0.695	0.605	0.696	0.576	0.760	0.664	**0.772**	0.614	**0.763**	0.623	**0.797**	0.701
D	0.579	0.591	**0.593**	0.576	**0.595**	0.587	**0.593**	0.582	0.587	0.573	0.58	0.567	**0.594**	0.587	**0.603**	0.576	**0.602**	0.584	**0.610**	0.576
E	0.412	0.529	0.451	0.381	0.447	0.486	0.478	0.455	0.453	0.405	0.425	0.396	0.465	0.467	0.496	0.427	0.485	0.434	**0.540**	0.442
F	0.496	0.564	0.517	0.483	0.481	0.524	0.532	0.520	0.509	0.486	0.498	0.481	0.513	0.507	0.538	0.493	0.533	0.501	0.548	0.498
G	0.814	0.809	0.731	0.671	0.711	0.742	0.804	0.807	0.722	0.665	0.721	0.671	0.751	0.760	0.788	0.681	0.781	0.697	**0.819**	0.762
H	0.928	0.868	0.759	0.683	0.743	0.771	0.877	0.870	0.765	0.696	0.768	0.707	0.808	0.793	0.836	0.719	0.822	0.716	0.883	0.812
I	0.627	0.652	0.643	0.629	0.644	0.644	0.648	0.641	0.634	**0.653**	0.636	0.636	0.651	0.640	**0.679**	**0.673**	**0.681**	0.644	**0.697**	**0.653**
J	0.665	0.701	0.672	0.656	0.670	0.665	0.672	0.668	0.667	0.644	0.667	0.653	0.675	0.668	0.690	0.652	0.689	0.665	0.698	0.661
K	0.718	0.763	0.704	0.609	0.662	0.683	0.76	0.762	0.711	0.62	0.664	0.601	0.726	0.685	0.752	0.631	0.744	0.647	**0.768**	0.682
L	0.512	0.619	0.550	0.409	0.539	0.583	0.596	0.588	0.587	0.446	0.592	0.425	0.586	0.552	0.618	0.477	**0.621**	0.489	**0.630**	0.551
M	0.589	0.655	0.599	0.506	0.575	0.581	0.627	0.617	0.609	0.506	0.598	0.479	0.622	0.559	**0.676**	0.522	**0.669**	0.508	**0.723**	0.568
N	0.503	0.668	0.534	0.493	0.519	0.497	0.534	0.509	0.517	0.494	0.501	0.492	0.519	0.525	0.543	0.497	0.531	0.496	0.549	0.495
O	0.823	0.848	0.741	0.519	0.648	0.575	0.845	0.721	0.764	0.509	0.801	0.444	0.841	0.659	**0.849**	0.519	0.838	0.444	**0.937**	0.724
P	0.678	0.852	0.638	0.445	0.605	0.639	0.704	0.683	0.661	0.542	0.654	0.487	0.677	0.615	0.696	0.559	0.689	0.511	0.717	0.628
Q	1	0.965	0.807	0.639	0.723	0.689	0.805	0.868	0.783	0.676	0.792	0.653	0.842	0.727	0.865	0.699	0.858	0.699	0.956	0.729
R	0.771	0.767	0.711	0.609	0.687	0.697	0.733	0.754	0.721	0.623	0.697	0.588	0.733	0.689	0.758	0.646	0.735	0.645	**0.786**	0.694
Av.	0.690	0.723	0.652	0.564	0.625	0.634	0.691	0.677	0.648	0.579	0.647	0.564	0.678	0.639	0.700	0.594	0.693	0.590	**0.732**	0.644

Table 4.7: Results of the IJS1, IJS2 and IJS3 algorithms applied for JP2 instances

BVM: Best values of Martinez-Sykora et al. (2017)

	BVM(F)	IJS1						IJS2						IJS3					
		BL4	BL∞	ML4	ML∞	Ut4	Ut∞	BL4	BL∞	ML4	ML∞	Ut4	Ut∞	BL4	BL∞	ML4	ML∞	Ut4	Ut∞
A	0.680	0.587	0.546	0.584	0.574	0.600	0.582	0.588	0.552	0.594	0.579	0.578	0.571	0.609	0.569	0.602	0.562	0.604	0.569
B	0.760	0.715	0.635	0.737	0.753	0.743	0.755	0.748	0.645	0.743	0.684	0.729	0.701	**0.768**	0.656	**0.768**	0.694	**0.804**	0.701
C	0.691	0.613	0.553	0.629	0.752	0.636	0.673	0.624	0.567	0.636	0.598	0.624	0.697	0.680	0.574	0.644	0.542	**0.692**	0.594
F	0.551	0.495	0.477	0.503	0.529	0.511	0.529	0.495	0.473	0.512	0.489	0.503	0.493	0.533	0.489	0.518	0.486	0.525	0.489
H	0.828	0.709	0.651	0.788	0.702	0.792	0.743	0.741	0.661	0.789	0.745	0.744	0.762	0.792	0.673	0.804	0.652	0.809	0.719
L	0.576	0.546	0.418	0.541	0.532	0.539	0.532	0.536	0.422	0.547	0.497	0.547	0.511	**0.582**	0.449	**0.581**	0.435	**0.588**	0.521
M	0.681	0.569	0.485	0.578	0.579	0.581	0.583	0.576	0.496	0.581	0.517	0.539	0.531	0.617	0.508	0.596	0.368	0.627	0.543
O	0.788	0.708	0.469	0.769	0.651	**0.803**	0.672	0.761	0.474	**0.800**	0.636	0.781	0.672	**0.823**	0.484	**0.790**	0.699	**0.826**	0.624
S	0.537	0.522	0.378	**0.539**	0.526	**0.623**	0.531	**0.621**	0.386	**0.623**	0.512	**0.602**	0.518	**0.621**	0.412	**0.591**	0.308	**0.642**	0.474
T	0.921	0.829	0.538	0.875	0.483	0.908	0.883	0.865	0.55	0.901	0.532	0.874	0.747	0.907	0.563	0.906	0.435	**0.959**	0.747
U	0.764	0.651	0.417	0.73	0.678	0.728	0.678	0.711	0.432	0.735	0.712	0.728	0.674	**0.777**	0.443	**0.772**	0.303	**0.823**	0.596
V	0.989	0.775	0.462	0.803	0.593	0.922	0.593	0.811	0.482	0.838	0.589	0.894	0.548	0.915	0.503	0.878	0.582	**0.993**	0.645
W	0.669	**0.696**	0.647	**0.689**	0.648	**0.705**	0.693	**0.735**	0.665	**0.704**	0.666	**0.678**	0.641	**0.782**	**0.678**	**0.787**	**0.677**	**0.843**	0.643
X	0.632	0.588	0.496	0.589	0.603	0.599	0.604	0.599	0.51	0.599	0.512	0.573	0.519	**0.632**	0.538	**0.641**	0.497	**0.649**	0.581
Y	0.726	0.657	0.584	0.672	0.714	0.706	0.714	0.683	0.596	0.721	0.648	0.688	0.698	**0.729**	0.599	**0.727**	0.577	**0.735**	0.633
Z	0.893	0.779	0.590	0.811	0.801	0.826	0.715	0.814	0.603	0.826	0.803	0.802	0.729	0.829	0.615	0.818	0.446	0.839	0.728
Av.	0.729	0.652	0.522	0.677	0.632	0.701	0.655	0.682	0.532	0.697	0.607	0.680	0.626	0.725	0.547	0.714	0.516	0.747	0.613

Table 4.8: Results of the IJS1 algorithm for each nesting instances with $2m^d$ Large bins

Values include the average of F values of 10 trials, t : average time for 500 iterations

	BL4		BL∞		ML4		ML∞		MU4		MU∞	
	F	t	F	t	F	t	F	t	F	t	F	t
Albano	0.383	77.8	0.395	162.2	0.402	74.8	0.387	154.2	0.404	83.2	0.388	163.3
Shape2	0.437	56.2	0.424	107.5	0.446	51.2	0.447	103.7	0.440	60.7	0.451	110.3
Trousers	0.451	294.7	0.414	683.5	0.459	314.6	0.44	718.8	0.462	358.3	0.462	754.3
Shape0	0.294	245.9	0.294	504.2	0.294	248.4	0.294	521.7	0.294	263.7	0.294	548.9
Shape1	0.304	416.5	0.306	740.4	0.304	408.3	0.308	751.4	0.304	408.3	0.318	748.4
Shirts	0.647	710.9	0.651	1221.4	0.649	626.8	0.652	1143.2	0.646	618.3	0.652	1118.4
Dighe2	0.260	14.4	0.260	35.2	0.260	17.0	0.260	37.6	0.260	16.5	0.260	34.6
Dighe1	0.329	17.9	0.329	33.6	0.329	18.2	0.329	34.6	0.329	17.6	0.329	35.1
Fu	0.487	4.9	0.502	6.3	0.502	5.0	0.505	7.0	0.503	4.4	0.505	8.2
Han	0.311	83.6	0.314	172.6	0.316	86.8	0.356	176.4	0.357	79.2	0.412	144.3
Jakobs1	0.617	52.4	0.609	92.4	0.613	50.9	0.616	98.3	0.617	48.9	0.616	92.4
Jakobs2	0.455	46.0	0.452	89.8	0.45	45.8	0.454	92.2	0.456	48.4	0.448	96.8
Mao	0.256	113.4	0.256	191.3	0.61	119.2	0.61	188.3	0.61	106.7	0.610	176.7
Poly1a	0.368	22.5	0.368	41.7	0.368	23.2	0.368	43.5	0.368	24.6	0.368	42.0
Poly2a	0.376	79.5	0.384	126.3	0.382	72.4	0.384	131.7	0.381	74.3	0.386	128.8
Poly3a	0.395	149.3	0.389	247.0	0.387	131.4	0.416	258.9	0.392	133.6	0.416	246.8
Poly4a	0.399	243.3	0.416	412.5	0.394	238.6	0.407	426.4	0.399	254.9	0.418	420.7
Poly5a	0.406	311.1	0.409	526.5	0.399	336.9	0.408	556.9	0.405	328.5	0.408	589.5
Poly2b	0.443	81.8	0.455	131.0	0.454	76.6	0.45	132.9	0.454	79.4	0.459	126.3
Poly3b	0.387	171.6	0.401	252.9	0.403	177.4	0.399	249.9	0.414	156.8	0.418	239.7
Poly4b	0.402	295.8	0.405	491.2	0.397	286.8	0.401	484.7	0.393	276.1	0.406	465.7
Poly5b	0.468	433.1	0.471	652.9	0.492	428.9	0.471	671.6	0.471	408.5	0.473	701.6
Swim	0.398	2206.5	0.393	3430.0	0.404	2134.8	0.404	3396.6	0.416	1988.1	0.408	3447.3
Avg.	0.403	266.5	0.404	450.1	0.422	259.7	0.425	451.3	0.425	253.9	0.431	453.9

Table 4.9: Results of the IJS1 algorithm for each nesting instances with $1.5m^d$ Medium bins

Values include the average of F values of 10 trials, t : average time for 500 iterations

	BL4		BL∞		ML4		ML∞		MU4		MU∞	
	F	t	F	t	F	t	F	t	F	t	F	t
Albano	0.495	79.2	0.504	0.504	0.506	84.5	0.523	180.5	0.525	87.3	0.532	178.0
Shape2	0.289	78.1	0.331	0.331	0.315	56.9	0.335	122.3	0.338	60.7	0.339	123.6
Trousers	0.557	374.6	0.562	0.567	0.564	349.2	0.565	783.5	0.590	379.8	0.574	814.6
Shape0	0.268	302.5	0.268	0.268	0.268	275.7	0.268	584.3	0.268	295.3	0.268	592.8
Shape1	0.270	552.6	0.369	0.369	0.270	494.0	0.376	766.5	0.275	494.0	0.369	823.2
Shirts	0.518	740.8	0.526	0.546	0.634	727.1	0.557	1337.6	0.665	680.1	0.664	1230.2
Dighe2	0.322	15.4	0.277	0.277	0.302	20.9	0.317	45.9	0.281	18.8	0.322	41.5
Dighe1	0.350	18.1	0.325	0.325	0.352	19.4	0.354	39.5	0.314	20.1	0.365	39.0
Fu	0.410	5.6	0.421	0.421	0.410	5.3	0.432	8.2	0.427	4.9	0.431	9.2
Han	0.387	94.6	0.376	0.376	0.384	89.4	0.405	215.2	0.402	84.8	0.402	161.6
Jakobs1	0.570	59.1	0.519	0.519	0.558	59.1	0.541	115.0	0.565	56.2	0.579	109.0
Jakobs2	0.408	49.0	0.401	0.401	0.398	55.8	0.417	93.1	0.413	57.1	0.438	96.8
Mao	0.494	119.1	0.493	0.493	0.496	119.2	0.494	201.5	0.495	124.8	0.501	192.6
Poly1a	0.320	23.0	0.322	0.329	0.335	24.1	0.322	46.2	0.320	26.8	0.349	46.2
Poly2a	0.361	96.9	0.358	0.358	0.353	78.9	0.368	131.7	0.346	80.2	0.366	150.7
Poly3a	0.424	164.3	0.431	0.431	0.427	142.0	0.435	310.7	0.425	147.0	0.442	273.9
Poly4a	0.398	282.2	0.412	0.412	0.392	243.3	0.433	464.7	0.401	295.7	0.437	441.8
Poly5a	0.405	345.4	0.439	0.459	0.394	397.5	0.457	640.5	0.458	335.0	0.465	613.0
Poly2b	0.378	86.7	0.381	0.381	0.381	78.2	0.389	154.2	0.396	93.7	0.402	154.1
Poly3b	0.434	185.4	0.432	0.432	0.433	195.2	0.431	249.9	0.428	175.6	0.462	292.4
Poly4b	0.461	343.2	0.465	0.465	0.453	349.9	0.465	533.2	0.401	289.9	0.483	568.2
Poly5b	0.415	441.7	0.439	0.479	0.438	450.3	0.478	698.4	0.478	412.6	0.479	834.9
Swim	0.339	2405.1	0.351	0.351	0.342	2625.8	0.351	4143.9	0.402	2246.6	0.368	3688.6
Avg.	0.403	298.4	0.409	0.413	0.409	301.8	0.422	515.9	0.418	281.2	0.436	499.0

Table 4.10: Results of the IJS1 algorithm for each nesting instances with $1.1m^d$ small bins

Values include the average of F values of 10 trials, t : average time for 500 iterations

	BL4		BL∞		ML4		ML∞		MU4		MU∞	
	F	t	F	t	F	t	F	t	F	t	F	t
Albano	0.597	80.8	0.590	168.4	0.591	89.6	0.590	180.5	0.594	96.0	0.605	185.1
Shape2	0.300	85.1	0.300	121.9	0.293	58.0	0.300	134.5	0.300	61.9	0.300	133.5
Trousers	0.679	408.3	0.678	840.8	0.684	363.2	0.681	807.0	0.681	387.4	0.684	839.0
Shape0	0.244	296.4	0.244	560.2	0.244	295.0	0.244	654.5	0.244	292.4	0.244	616.5
Shape1	0.246	569.2	0.327	957.4	0.282	479.2	0.317	766.5	0.282	563.1	0.320	922.0
Shirts	0.601	829.7	0.590	1448.3	0.473	763.4	0.611	1377.7	0.465	680.1	0.611	1279.4
Dighe2	0.336	17.2	0.373	40.3	0.341	23.6	0.381	44.5	0.373	18.2	0.394	46.1
Dighe1	0.421	19.3	0.428	39.4	0.424	19.4	0.428	41.9	0.422	21.3	0.427	41.7
Fu	0.358	6.1	0.351	6.9	0.352	5.3	0.358	9.0	0.352	5.1	0.448	9.1
Han	0.437	105.9	0.430	206.9	0.447	99.3	0.445	241.0	0.461	90.7	0.446	168.0
Jakobs1	0.358	59.1	0.429	114.4	0.368	57.9	0.433	111.5	0.358	56.2	0.433	118.8
Jakobs2	0.365	48.5	0.415	102.8	0.401	58.1	0.415	97.8	0.401	58.8	0.418	101.7
Mao	0.437	120.3	0.450	229.1	0.447	131.1	0.450	205.5	0.449	138.5	0.472	190.6
Poly1a	0.300	25.9	0.402	44.2	0.310	27.3	0.451	49.4	0.309	29.5	0.454	47.6
Poly2a	0.349	104.7	0.353	131.4	0.352	78.9	0.363	134.3	0.367	80.2	0.452	150.7
Poly3a	0.384	185.6	0.368	320.1	0.373	154.7	0.382	326.3	0.367	148.5	0.451	304.1
Poly4a	0.402	285.0	0.349	489.7	0.384	267.7	0.351	520.5	0.383	337.1	0.409	437.3
Poly5a	0.382	335.0	0.364	578.5	0.367	437.3	0.399	704.5	0.358	355.1	0.413	692.7
Poly2b	0.371	95.4	0.412	145.7	0.37	80.5	0.412	161.9	0.339	100.3	0.419	154.1
Poly3b	0.373	207.6	0.384	292.0	0.381	218.6	0.452	247.4	0.378	172.1	0.456	304.1
Poly4b	0.406	332.9	0.407	551.5	0.407	356.9	0.418	549.2	0.405	292.8	0.428	562.5
Poly5b	0.397	503.6	0.422	766.4	0.419	490.8	0.422	726.4	0.415	416.8	0.435	935.1
Swim	0.330	2693.7	0.325	3999.4	0.326	2652.1	0.338	4019.5	0.338	2358.9	0.337	3688.6
Avg.	0.394	322.4	0.408	528.5	0.393	313.4	0.419	526.6	0.393	294.0	0.437	518.6

4.8 Concluding Remarks

In this chapter, we address a new solution method to solve the two-dimensional irregular shape bin packing problem which has received little attention in the literature despite clear industry relevance. We modify the existing use of the Jostle procedure of Dowsland et al. (1998) applied to strip packing and convert the heuristic for the homogeneous bin packing problem. Further, we introduced a diversification strategy into Jostle that significantly improved the results. The proposed heuristic supports the use of restricted rotation and unrestricted rotation of pieces.

The findings of the study are as follows:

Jostle is a promising packing heuristic that can be applied to bin packing problems. It is demonstrated to be a low-cost search strategy that self-governs the neighbourhood structure that modifies the solution. The designed algorithms have been shown to improve on the results in the literature with lower computation time and do not require the use of expensive MIP (Mixed Integer Programming) software, making it attractive to small companies.

We have introduced a new way of dealing with free rotation of pieces. We have demonstrated the value of free rotation for nesting instances and justified the results of restricted rotation for the jigsaw puzzle instances.

We have designed and tested a number of placement policies and shown that despite the extensive use of bottom-left (BL) this does not produce the best results when compared with minimising length or maximising utilisation.

Chapter 5

Irregular shape Bin Packing in Heterogeneous Bins

5.1 Introduction

This chapter discusses the solution methods to solve the two-dimensional irregular shape bin packing problem with heterogeneous bins. The problem mainly applies in industries such as sheet metal, foam cutting, furniture, plastic, paper and leather; where a set of polygonal shaped pieces need to be cut out of rectangular stock sheets (bins). The bin sizes are heterogeneous and more than one bin is usually required to satisfy the total demand for pieces. All the demanded pieces must be placed in the bins in a way that the total area of the utilised bins is minimised. We denote this problem as two-dimensional irregular shape multiple bin size bin packing problem (2D IMBSBPP).

The problem selection is particularly motivated by the set of real-life characteristics common for several industries. The typical form of sheet metal, foams, and timber sheets are readily available to purchase in their standard sizes due to the convenience of handling and storage. Most of these sheet surfaces are homogeneous and therefore do not require to restrictions on the orientation of the pieces when placing them in the layout. Therefore, restrictions on piece rotation are not needed.

Instead of using one standard size, companies usually use several standard sheet sizes which are economical, in satisfying the customer demand. The aim is not only to reduce the waste generated in the cutting process, but also to charge a competitive price to the customers, which usually depends on the area of material used to meet the demand. In this study, we present an algorithm which is capable of exploiting the fact of having different standard stock sheet sizes when assigning, rotating and placing irregular pieces so that the trim loss can be significantly reduced.

As reviewed in Chapter 2, the previous studies on 2D irregular shape bin packing were limited. Only a few studies have been published recently for 2D irregular bin packing with homogeneous rectangular bins (e.g. Lopez-Camacho et al. (2013a), Lopez-Camacho et al. (2013b), Lopez-Camacho et al. (2014), Martinez-Sykora et al. (2017)). Out of those, only Martinez-Sykora et al. (2017) considered the continuous rotation of pieces that added more complexity to the problem. To the best of our knowledge, Babu and Babu (2001) and Baldacci et al. (2014) were the only papers that had discussed packing irregular shapes in heterogeneous bins. Both these studies presented results for packing pieces in irregular shaped bins. In real-life, the number of applications of this problem is limited (i.e. Leather industry) compared to the number of applications of packing irregular pieces into heterogeneous rectangular sheets. Both papers adapted the raster method and approximated the geometric shapes of pieces and stock sheets which cause low accuracy in piece placement. However, the study discussed in this chapter focuses on irregular shapes packing in heterogeneous rectangular bins with the unrestricted rotation of pieces. Using the geometric representation method the study improves the accuracy of solutions.

5.2 Contribution

The main contribution of this chapter is to develop methods to solve the 2D IMBSBPP considering the unrestricted rotation of pieces.

This chapter describes three main computational methods to solve 2D IMBSBPP and identifies an efficient method to produce good solutions. The first two methods are based on Jostle approach discussed in Chapter 4 and the third method is inspired by the idea of improving the overall utilisation by selecting the most appropriate bin types using a classic bin selection heuristic.

As explained in Chapter 4, compared to other available algorithms, one of the most important properties of the Jostle is the exploration of the solution space in a relatively low computational effort, especially when the piece rotation is not restricted. The Jostle heuristic explores the search space over the permutation of the pieces and it applies in a scenario where the sequence of bins and the placement rule are fixed. We propose a modification to the constructive algorithm discussed in Chapter 4 for the 2D ISBSBPP, and this is described in Section 5.4. We then propose the Jostle heuristic to explore the search space over the permutation of the pieces, within a given sequence of bins.

The first main computational method extracts the features of Jostle as well as the power of Genetic algorithm (GA) to solve 2D IMBSBPP. Babu and Babu (2001) considered GA to search over the solution space of the problem. They coded a long chromosome representing the sequence of bins, the sequence of pieces and sequence of the orientation angles of the pieces. Each chromosome represented a solution. This representation is inefficient when the number of pieces and bin types are high. Also, the piece orientations of this approach were limited to a restricted set of angles. In the proposed new approach, we involve the efficient Jostle heuristic to handle the placement and orientation decisions of pieces within the GA implementation so that the overall GA/Jostle implementation can solve 2D IMBSBPP with either restricted rotation of pieces or unrestricted rotation of pieces.

As the second main computational method, we propose an iterative search procedure for the Jostle heuristic by applying *kicks* for the sequence of bins at frequent intervals so that the bin sequence can be changed during the search process. In this case, we extend the Jostle heuristic developed for 2D ISBSBPP to generate solutions for 2D IMBSBPP.

In the third algorithm, for each permutation of the pieces, the solutions are generated using a classic bin selection heuristic and some of the features of the constructive method proposed in Chapter 4. In this case, the quality of a solution is based on the piece permutation and the placement rule only, as the sequence of used bins is determined by the classic bin heuristic. By searching over different permutations of the pieces, the overall computational method can find a good solution.

In comparison to the other approaches in the literature, to our knowledge, this is one of the few studies available for solving 2D IMBSBPP and the first study addressing unrestricted rotation of pieces in 2D IMBSBPP.

5.3 Problem Description

Let N' be the given number of rectangular bin types (stock sheets) and let L_k and W_k be, the length and the width of bin type $k \in \{1, \dots, N'\}$, respectively. Let $P = \{p_1, \dots, p_n\}$ be the set of pieces to be cut from the bins where n be the number of pieces to be cut.

A solution $s = \{B^s_k |\ k = 1, \dots, N'\}$ is represented by a set of bins of each type used in the solution, where each single bin $b^s_{jk}(P_{jk}, O_{jk}, X_{jk}, Y_{jk}) \in B^s_k$ is determined by the subset of pieces placed in the bin (P_{jk}), the set of orientations used for each piece (O_{jk}) and the X and Y coordinates of the reference point of each piece (X_{jk} and Y_{jk}), respectively. In this case, j denotes the index of each single bin of type k. A solution s is a feasible solution if all the pieces are placed, $\bigcup_{k=1}^{N'} \bigcup_{j|\ b_{jk} \in B^s_k} \overline{P}_{jk} = P$, there is no overlapping between each pair of pieces placed in the same bin and no piece exceeds the bin dimension.

Initially, we consider an unlimited number of bins of each type k, (i.e. There are enough bins of each type to place all the pieces). We denote N^s_k as the number of used bins of type k in solution s.

The position of the pieces inside the bins is given by the coordinates of the reference point, which we assume as the bottom-left corner point of the enclosing rectangle. Note that the reference point may change if the piece is rotated.

5.3.1 Evaluation function of a solution

We use two measurements to evaluate the quality of a complete solution s.

1) The overall utilisation U_s is defined as;

$$U_s = \frac{\sum_{i=1}^{n} Area(p_i)}{\sum_{k=1}^{N'} W_k L_k N_k^s}$$

where $Area(p_i)$ denotes the area of the i^{th} piece. Since $\sum_{i=1}^{n} Area(p_i)$ is constant, then maximizing the utilization is equivalent to minimizing the total area of bins used in s. In this chapter our aim is to maximize U_s.

2) Let σ_s be the standard deviation of absolute bin waste of the used bins. This measurement is used during the algorithm to break ties when two solutions have the same overall utilisation. A low σ value shares the waste among the bins with low variance and it balances the individual bin utility. A higher σ_s leads to a higher variation in bin space utilisation. This effect is prominent in situations where both dense and loose packing is possible. Therefore, within the same solution, a set of bins gets a higher utility while another set of bins gets a lower utility. We encourage this types of solutions (i.e. a higher σ_s values when there is a tie in the U_s values) during the search process of the algorithm to move the few pieces packed inside a large bin (i.e. a lower utilised bin) to another used bin. Therefore the σ is a significant measurement at the improvement stage of the algorithm.

In literature for 2D MBSBPP with rectangular items, the cost of bins is basically defined in two ways; 1) The value of the bin is proportional to its area (e.g. Ortmann et al. (2010), Hong et al. (2014)), 2) The value of bin is not proportional to its area (e.g. Pisinger and Sigurd (2005)). In the second case, there are possibilities of incurring a higher cost for large bins than the cost proportional to its area or incurring a relatively higher cost for small bins than the proportional value corresponds to its area. In this study, we selected the first option when deciding the cost of bin usage by valuing only the usage of sheet material area. Therefore, we use U_s as the main measurement when deciding the good solutions

5.4 Packing Procedures

In this section, we introduce the three computational methods used to solve 2D IMBSBPP. The packing procedure discussed for 2D ISBSBPP in Chapter 4 follows the strip packing approach while imposing division between bins. This made bin allocation implicit within the placement decisions. However, 2D IMBSBPP contains rectangular bins with different dimensions. When a sequence of bins is fixed with a set of heterogeneous bins, then the 2D IMBSBPP requires us to address two decision problems; *assigning pieces to bins* and *placing those pieces in the bin*, similar to the packing procedure discussed for 2D ISBSBPP. Therefore, for a given sequence of heterogeneous bins, our approach stimulates jostling of pieces to solve both the decision problems.

As explained in Chapter 4, the idea is to begin at a random order of pieces and apply a constructive heuristic. The Jostle heuristic determines the placement order of pieces for the next iteration, by scanning the piece positions of the previous layout. In order to adapt the Jostle for 2D IMBSBPP, we first define a very long packing strip of which the width is equal to the maximum width of the available bin types; W_{max}. The length of the strip is decided based on how the number of bins is arranged along the strip. The strip length $2n \times \sum_{k=1}^{N'} L_k$ was set to arrange $2 \times n \times N'$ bins along the strip as follows.

For the first step, the bin order is determined randomly and this represents a possible random order of given heterogeneous bin types. As an example, the strip represented in Figure 5.1 denotes a set of bins joined together. We defined bin spaces by assigning a set of barrier lines at different intervals of length and width, to denote each bin arranged along the strip, following the generated random order of bins. The bottom left corner of bin $v \in \{1, \ldots, n \times N'\}$ on the strip is placed at $(\sum_{i'=0}^{(v-1)} L_{i'}, 0)$ (see Figure 5.1). We directly applied the Jostle procedure to the strip with additional constraints that pieces may not be placed across a barrier and should always be placed entirely within the bins. As illustrated in Figure 5.1, the algorithm arranges bins to facilitate jostling between the both ends iteratively. Dowsland et al. (1998)'s method shuffles pieces along a homogeneous strip towards left and right. However, in our case, since the bin arrangement is heterogeneous, we considered the mirror image of one side to the other of the strip, so that same bin configuration can be established at both ends when the Jostle performs within a given bin configuration.

5.4.1 Constructive Algorithm 1 (CA1)

Following this procedure, for a given order of bins, we apply the constructive algorithm described in Section 4.4 when placing pieces in the bins. The algorithm facilitates loading of pieces into the bins, allowing feasible placements of pieces, hole-filling and unrestricted rotation of pieces. The main difference of this constructive method is its consideration of heterogeneous bins arranged in the Jostle strip. In each of our designed

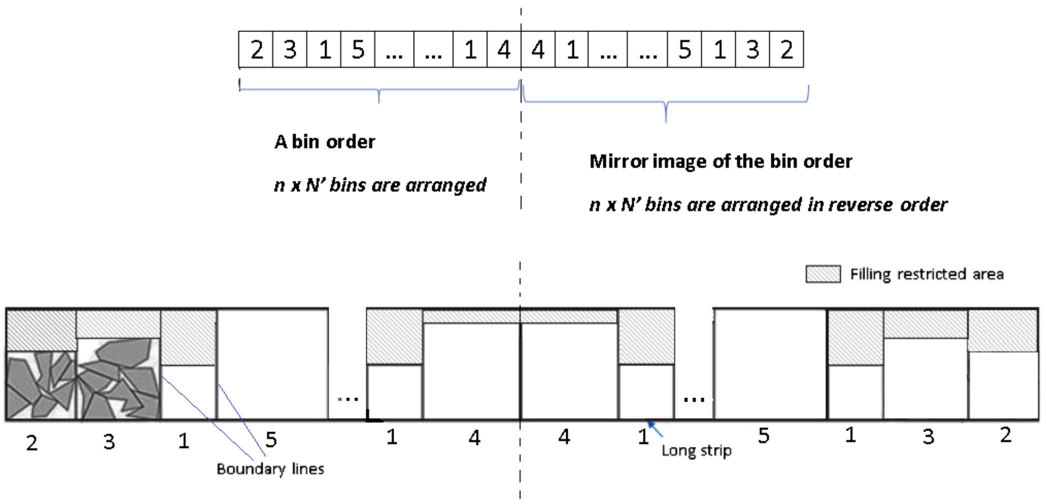


Figure 5.1: Strip to Bins

computation methods, a complete packing solution is evaluated according to the quality of the solutions, which is measured by U_s. As a part of solution construction, both U_s and σ_s are calculated for each solution, to differentiate between solutions with the same U_s values. For the forthcoming uses of this constructive method, we denote it in this chapter as Constructive Algorithm 01 (CA1) for solving 2D IMBSBPP.

5.4.2 Solution improvement by Jostle heuristic

In this section we describe how we adapted CA1 within the sequence search algorithm; Jostle. In this case, our proposed Jostle heuristic starts with a random order of pieces and a random order of bins to construct the first layout. Subsequently, the piece permutation is decided by the Jostle heuristic when jostling continues with pieces within the selected order of bins. The placement rule sets the criteria to compare the orientation and placement position of a candidate piece in a feasible manner. As described in Chapter 4, Jostle is analogous to a local search where the search occurs over the sequence of the pieces and each Jostle iteration generates a neighbour sequence. The constructive algorithm takes a certain sequence of pieces and creates a layout for a certain configuration of heterogeneous bins. In other words, within a given configuration of bins, Jostle allows to generate a new layout, by defining a new sequence of pieces. This is a neighbour solution based on the arrangement of pieces of the previous packing layout. The optimization arises from generating neighbour solutions in which the constructive algorithm generates solution when there is a change in the piece permutation. Despite its performance by finding the promising orientations and placements of pieces, the approach is limited to finding a good piece arrangement within a given configuration of bins only. This is a major concern for our problem since the search cannot explore the different configuration of bins.

We proposed two search mechanisms to explore the bin permutation and tested their performance. The first method involves both Jostle and Genetic Algorithm to search through the different configuration of bins considering a population of solutions at a time. The second is a single point search method which allocates a separate kick-move mechanism to change the bins configuration while Jostle changes the permutation of the pieces.

5.4.3 Hybrid approach of GA and Jostle

The first computational method implements a Genetic Algorithm (GA), to work with multiple solutions in parallel and to combine them together to generate a new solution. In the C&P literature, GA has been implemented for SBSBPP with rectangular pieces by Gonçalves and Resende (2013), for MBSBPP with rectangular pieces by Babu and Babu (1999) and for packing irregular pieces in irregular stock sheets by Babu and Babu (2001). In our implementation, a candidate solution of 2D IMBSBPP is represented by a sequence of input bins and a permutation of the pieces. For each chromosome, the initial input is a random permutation of bins and a random permutation of pieces. This search algorithm applies the constructive algorithm CA1, for each decided permutation to generate solutions. The algorithm stores information related to each candidate chromosome and their corresponding solutions; i.e. permutation of bins, pieces, piece orientations, and associated fitness of a solution.

An example coding of a chromosome when packing 16 pieces into the bins having five different types (sizes) are presented in Figure 5.2. The length of the bin order is equal to $n \times N'$. In Figure 5.2 the numbers 2,4,4,1,...,3,4,1,2 represent the ordered bin types (IDs) of the input bins. The second part of the code represents the sequence of pieces. A chromosome corresponds to a certain bin packing solution that can be generated using the constructive algorithm. The quality of each chromosome is evaluated using the fitness function $f(B^*, P^*)$. In this case, the value of the function is determined using U_s and σ_s measurements which we discuss in detail at the solution evaluation stage of this method.

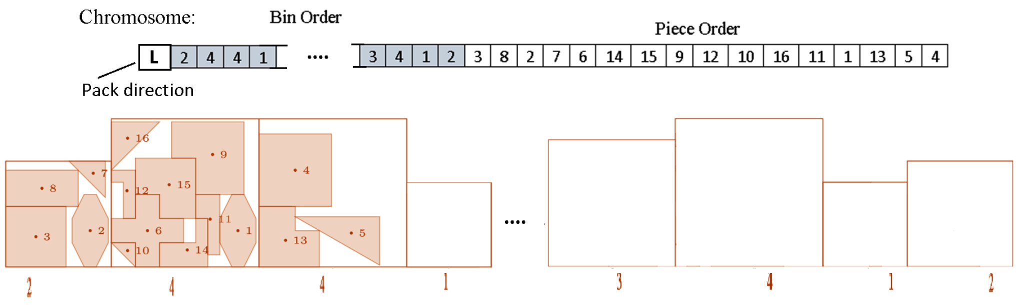


Figure 5.2: Representation of solutions

5.4.3.1 Outline of HGAJ

Algorithm 5 describes the general working procedure of HGAIJ. Let *popSize* be the number of chromosomes in the population progresses from one generation to another. As explained above, initially *popSize* number of solutions are generated using CA1 by taking a set of random order of bins and random permutation of the pieces. In our implementation, these set of chromosomes represents the population at first generation ($GenN = 1$). Using these chromosomes, the algorithm finds their fitness values by generating the corresponding solutions. As the next step, the crossover and mutation operators are applied to this population of chromosomes and generates a new set of offspring solutions. The generated offspring solutions are further processed using the Jostle. Once this is done, all the generated offspring are added to the population so that the total group includes the new set of offspring as well as the chromosomoes stayed at the start of the generation. As the next step, the selection mechanism decides which chromosomes become parents in the next generation. The selection mechanism selects *popSize* number of chromosomes from the existing group of chromosomes, and those who do not get selected are considered as discarded. In this way *popSize* number of chromosomes are transfered from one generation to another. Also, the chromosomes transferred to the next generation can contain some of the chromosomes at the start of the generation and some of the new offspring chromosomes made by the GA operators and improved by the Jostle heuristic. The purpose of the crossover and mutation operators is to create new solutions by changing the bin order and piece order by using the parent chromosomes available in the current population. Likewise, the GA cycle works with a population of solutions, and members in the population are updated or changed at the selection stage. Next, we describe each of these steps in detail.

Algorithm 5: General structure of GA Jostle method

```
1  Initialise a population of chromosomes;
2  Evaluate each chromosome in the population;
3  Main loop: Generations;
4  $GenN = 1$;
5  while $GenN < MaxGen$ do
6      Create new chromosomes by mating chromosomes using crossover and mutation;
7      Improve offspring solutions by Jostle;
8      Evaluate and Insert new members into the population;
9      Select set of chromosomes to the next generation from new offspring and old
         parents;
10     $GenN = GenN + 1$;
11 end
```

5.4.3.2 Crossover

Once parent chromosomes are selected from the chromosomes by the selection mechanism, the crossover parameter P_c determines whether those selected parents will undergo crossover or not. The parents who do not undergo crossover will be cloned (i.e copied) as offsprings. In this case, P_c denotes the probability that a parent undergoes crossover. The selection mechanism selects parents for mating in the current generation from the chromosomes in the previous generation. We describe this procedure later; under the sub topic called *Selection for the next population*.

The crossover operator is applied for two parent chromosomes and creates two new offspring chromosomes. The underlying idea of crossover is to produce offspring chromosomes, which can inherit common genes from their parents. Following this, our objective was to include the key genes or sections of genes shared by many of the fittest chromosomes inside the offspring chromosomes. We use two forms of crossover operators in this study; uniform crossover for the bin part of the chromosome and ordered crossover for the piece permutation, as follows.

First, we use the *uniform crossover* operator, which facilitates a search through most of the possibilities of re-combining different genes in parents (Falkenauer, 1999). An example, Figure 5.3 illustrates the uniform crossover operation to generate a child chromosome. The operator first constructs a binary mask, with random bits of 0 and 1. The first child then inherits the genes (in the bin order) of the first parent if there is a "1" in the mask; otherwise, the first child inherits the genes of the second parent. The second child is formed following the opposite way so that the genes are copied from the second parent if there is a "1" in the mask and from the first parent if there is a "0" in the mask.

The second crossover operator is used to change the piece order (in this case the bin string related for piece permutation only). The procedure of it applies as follows. First, from the starting point of the bit-string up to a randomly selected point is copied from the first parent to the first offspring as illustrated in Figure 5.3. The second parent's piece permutation is then scanned and finds the bits which are not yet included in the offspring to copy them to the offspring. The same procedure is used to generate the second offspring, in which case copying genes are started from the second parent and then from the first parent.

5.4.3.3 Mutation

After crossover, the mutation parameter P_m determines whether those selected offspring will undergo mutation or not. The purpose of implementing mutation in this algorithm is to maintain diversity within the population and inhibit premature convergence. In

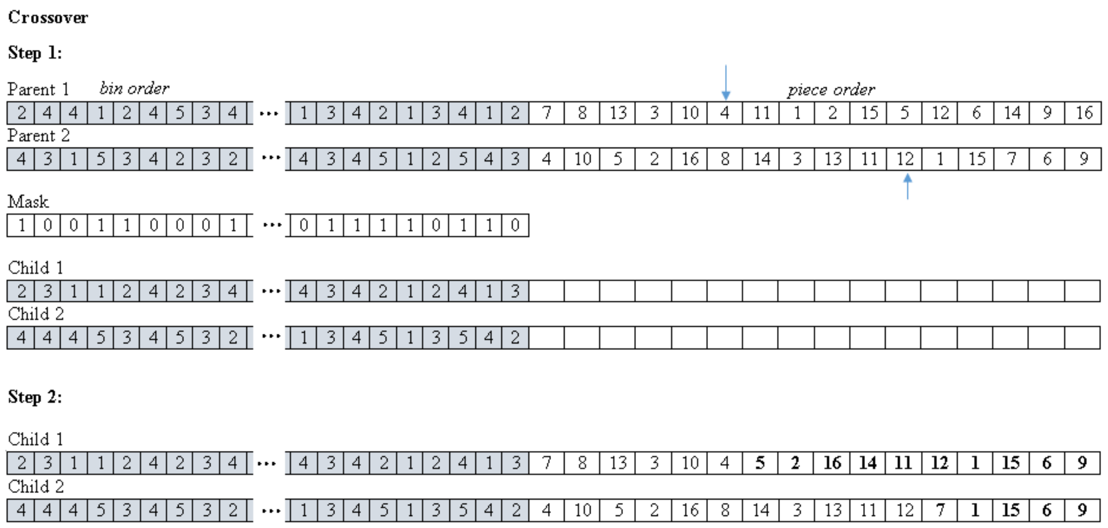


Figure 5.3: Crossover operation

our approach, we expect a major change in the bin configuration of the chromosomes selected for mutation. In this case, we adopt *inversion mutation* technique to change the bin permutation of offspring chromosomes subject to the probability parameter (P_{mu}), as described in Eiben and Smith (2013) and Mutingi and Mbohwa (2016).

If a chromosome undergoes mutation, then the inversion mutation is implemented as follows. As described in Eiben and Smith (2013), the inversion mutation selects two positions of the gene string randomly and reverses the order of genes in that selected substring. In our case, we only target the bin bit array of the chromosome. As the two positions of the gene arrays, we select one position randomly from the genes corresponds to the used bins and one position from any other random place in the gene array corresponds to the bins of the chromosome. The operator then reverses this selected string of genes and updates the gene array of the chromosome.

By doing this, we noticed a major change in the corresponding solution as bin spaces of the layout is dramatically changed. Accordingly, the piece placements have also been changed. This explores solutions in the search space at a greater distance and makes the search mechanism productive by inhibiting premature convergence.

5.4.3.4 Implementation of Jostle within the GA framework

Once a child chromosome is made using a certain permutation of bins and pieces, the Jostle procedure is used to improve the piece layout within the sequence of bins of that chromosome. Jostle improves the child chromosome by changing the orientation and placement position of pieces. Since Jostle is efficient in managing orientations of pieces, we do not propose any change for the piece orientations through genetic operators. On the other hand, handling orientation angles by genetic operators and generating solutions accordingly will require higher computational time as it requires us to consider a higher number of orientation options. Instead, in Chapter 4 we already described a new *angle tuning mechanism* which finds promising orientation angles efficiently once it is embedded with Jostle. Therefore, we implemented Jostle to improve offspring chromosomes generated by the GA operators. The number of Jostle cycles ($maxJS$) used to improve the offspring which was found by a parameter evaluation study.

5.4.3.5 Selection for the next population

The next population is selected from the *Pool* that currently contains both the parent and children solutions. The proposed selection strategy is set to ensure both quality and diversity of the next population. The chromosomes in the next population is selected from the chromosomes in *Pool*, using *elitist* and *tournament* selection methods.

The selection process uses three measurements; U, σ and *dist* to select the chromosomes that move forward to the next generation. The chromosomes in *Pool* are sorted in descending order, first by their U value and then by the σ value. The measurement d is set to maintain the diversity among the solutions having an equal number of used bins. If two solutions have the same bin permutation with the same number of used bins, then our objective is to make sure one of them is not in the next population. This provides an opportunity for another solution which is having the same U_s value, in the next generation. The following example presents such incident.

Example:

In this example, the bins in bold font denote the used bins in the bin sequence. The first bin size is $2.5d_{max}$ in solution A; where the length and width of the bins are equal to $2.5d_{max}$. The measurement d_{max} denotes the maximum dimension of the enclosing rectangle of each piece in their original orientation.

Solution A: Bin arrangement $\mathbf{2.5}d_{max}$ $\mathbf{1.5}d_{max}$ $\mathbf{1.0}d_{max}$ $\mathbf{2.0}d_{max}$ $1.0d_{max}$....$2.5d_{max}$

Solution B: Bin arrangement $\mathbf{2.5}d_{max}$ $\mathbf{1.5}d_{max}$ $\mathbf{1.0}d_{max}$ $\mathbf{2.0}d_{max}$ $1.0d_{max}$....$2.5d_{max}$

Solution C: Bin arrangement $\mathbf{2.5}d_{max}$ $\mathbf{1.5}d_{max}$ $\mathbf{1.25}d_{max}$ $\mathbf{1.75}d_{max}$ $1.0d_{max}$....$2.5d_{max}$

Both solutions A and B have the same utilisation, the same number of used bins and probably different σ values due to the arrangement of pieces. In this example, assume sigma of A is higher than sigma of B. The solution C also has the same Utilisation, the same number of used bins and probably different σ, lower than σ of A and σ of B. Then, out of those three, in the case where only two need to be selected, we set the procedure to promote selecting A and C by defining a distance (*dist*) value for each chromosome as follows.

The best solution of the *pool* gets a distance value '0', and compared to its bin configuration, The other chromosomes get *dist* value. The distance of a chromosome x; $dist(x)$ is calculated as:

$$dist(x) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_{Nx} - y_{Nx}|$$

where x_i denote the area of the i^{th} bin located in the bin permutation of the chromosome, y_i denote the area of the i^{th} bin located in the bin permutation of the chromosome who got distance value '0'. In a situation where the chromosomes x and y have a different number of used bins (different N values), then the distance is calculated up to the N value of chromosome x (denoted as N_x). Following this mechanism, both Solutions A and B gets same *dist* value and C gets a different *dist* value compared to A and B.

When selection sorts the best chromosomes, the calculated d values are considered in order to break the ties of solutions with equal U values as explained in the above example. In this step of the algorithm, σ values have not been considered since the priority is assigned first to get a better U_s and then to pick a higher *dist* value at ties in U_s. In which case, however there is an exception in picking the higher *dist* value only at the selection which involves the best chromosome in *pool* whose $dist = 0$.

Elitist selection scheme

In order to keep a balance between the diversity and convergence speed, the elitist selection scheme copies a fraction ($\gamma\%$) of the best chromosomes in *Pool* to the new population. $\gamma\%$ is a parameter of this algorithm. When arranging best chromosomes, quality of chromosomes is evaluated by the priority order of U , *dist* and σ values. Once a $\gamma\% \times popSize$ of chromosomes are selected for the next generation, the remaining $100 - \gamma\% \times popSize$ are selected using the tournament selection.

Tournament selection scheme

The tournament selection makes a *tournament* among few of chromosomes chosen at random from the available chromosomes and selects the best out of that tournament so that it will be present in the next generation. The tournament size T_s is determined usually at the parameter tuning stage.

In our study, we use both elitist and tournament selection schemes to promote the selection of fit chromosomes, ensuring a reasonable diversity in bin configurations so that we can reduce the early convergence. Through the tournament selection we expect to preserves diversity in the selected chromosomes by increasing a chance to all chromosomes to be selected. In this case we test for a good T_s value to work with. In our studies, we realised that higher tournament size provides less chance for weak chromosomes to be selected and this leads possibly losing diversity.

5.4.3.6 Addressing some issues of GA Jostle approach

One major issue of this algorithm is the high computational time it takes to achieve good solutions. The algorithm takes a longer time to transfer from one generation to the next, hence reduces the frequency of bin-moves. At each generation, a significant amount of computational time is devoted for Jostle to improve at each solution and this effort is multiplied by the number of offsprings generated at each generation.

In order to reduce this computational effort, we first tried to reduce the depth of Jostle search by reducing the $MaxJS$. This results in a drop in the solution quality determined by Jostle.

Next, we tried to apply Iterated Jostle procedure discussed in Chapter 4, only for the newly generated best offspring. This minimises the computational effort of Jostle on other offspring. The second change has improved the performance of the algorithm by saving time with a lower compromise of its exploring ability. Accordingly, the GA framework is implemented on a population of chromosomes to explore different bin configurations, hence improving the piece arrangement of the best offspring. In summary, while crossover and mutation operators search over the bin arrangements and the piece permutations, the Iterated Jostle focuses on exploring permutation of pieces and orientation angles. We denote this approach (the second version of HGA: HGA_v2) as the Hybrid approach of Genetic Algorithm and Iterated Jostle (HGAIJ).

5.4.4 Iterated Jostle with Random Assignment of Bins (IJRAB)

This is the second main computational method proposed in this chapter to solve 2D-IMBSBPP. As the name implies, IJRAB uses the Jostle approach discussed in Section 5.4.2. The constructive method CA1 described in Section 5.4 is involved in generating solutions for a given order of bins and for a given permutation of the pieces. IJRAB is a single point search algorithm working over the sequence of bins and pieces. The previous Hybrid approach of GA and Jostle considers multiple solutions in parallel. In contrast, the IJRAB method involves jostling of pieces within a given order of bins and changes the bin order at frequent intervals while Jostle continues.

The IJRAB starts with a solution generated by the constructive algorithm CA1 using an arbitrary order of bins and an arbitrary order of the pieces. Then the Jostle heuristic determines a neighbourhood structure by shaking pieces from left to right and right to left along the bin order, so that each Jostle iteration generates a neighbour sequence of the pieces. As discussed in Chapter 4, Jostle finds a local optimum after a pre-defined number of Jostle cycles (K_p) with no improvement. Within the same bin order, the Iterated Jostle (IJS) is implemented to find a good solution, by applying kicks to the piece order as described in Chapter 4. In this algorithm, we denote these kicks as *piece* *kicks*, which are applied when the search is at a local optima (after K_p Jostle cycles). Likewise, for a selected bin order IJS is performed iteratively until no improvement is found for a pre-defined number of piece kicks (K_b). As the next step of IJRAB, the algorithm performs a perturbation to the current best solution found, instead of starting IJS with a completely new order of bins. In this case, the next bin order is decided by the perturbation applied to the bin order corresponds to the current best solution. We denote this step as the application of a *bin kick*. Once the next bin order is decided, IJS jostles pieces again starting from the piece permutation corresponds to the current best solution. This procedure continues with IJS and the bin kicks. The current best solution can be updated at any point of the algorithm by an improved solution found.

The purpose of implementing bin kicks is twofold. First, it allows to escape from one neighbourhood landscape to another neighbourhood landscape and starts searching in a new area of the search space. According to our experiments, this enhances the search ability of the mechanism and allows us to explore different local optimum solutions in the search space. Second, by changing the bin order, it removes the limitation of that Jostle heuristic being considered only for a selected order of bins. In this implementation, a *bin kick* is performed when the search process is reached to a predefined number of *piece kicks* with no improvement (K_b) in solutions within the selected bin order. In the next set of paragraphs we describe the implementation details of these two kick types.

- *Piece kick:* We perform the *insert move* as described by Bennell and Oliveira (2009) for a single piece (i.e. one-piece insert move) in the piece permutation of the leading solution as follows: First, we select two random positions along the permutation of the pieces and then remove the piece corresponds to the highest position selected. The algorithm is set to insert this piece in front of the piece corresponds to the other position which is selected randomly. Accordingly, the new piece order is derived after ordering the remaining pieces, following the existing permutation.

- *Bin kick:* In order to change the bin order, a random bin is selected out of the used bins of the current best solution found and it is replaced with a random bin selected out of the other bin types. Once this is applied the same corresponding change is applied for the mirror bin position of the bin order as well so that the same bin configuration can be established at both ends when performing Jostle. Application of this bin change is critical for this algorithm to explore the solutions related to different bin configurations.

Algorithm 6 describes execution steps of IJRAB. We use $P^{(L,t)}$ to denote piece order at t^{th} iteration, which packs pieces from left to right. $P^{(R,t)}$ denotes the piece order which packs pieces from right to left at t^{th} iteration. The bin order at t^{th} iteration is denoted as $\tau^{(t)}$. For a given solution $s^{(t)}$, $f(s^{(t)})$ denotes the evaluation value by taking into account the values of U_s and σ_s (see Section 5.3.1). The current best solution of all the executed iterations is denoted by s^{**}. Also, the current best solution of a particular bin order is denoted by s^{*}. T_{max} defines the maximum number of total Jostle cycles that the algorithm which is the termination criterion of the algorithm. Similar to the maximum number of total Jostle cycles, the termination criterion can be set run for a certain amount of time.

5.4.5 Sequential Packing with a Bin Centric Heuristic (SPBCH)

In this approach, we build a feasible solution for a given permutation of the pieces. Our approach is inspired by the idea of improving the utilisation of bins of classical best fit heuristic. Our focus is to find the best bin types configuration that packs pieces for higher utilisation. Following a given arrangement of pieces, the proposed bin-centric heuristic determines an order of bins for the layout. We describe this procedure in section 5.4.5.1 and discuss how solutions can be improved through the local search mechanism implements in Section 5.4.5.2.

5.4.5.1 Constructive Algorithm 2 (CA2)

The constructive algorithm initiates all pieces as an *unpacked piece list* (P_u) and this is the first step of the algorithm. Following an initial piece order, pieces are inserted temporarily to one bin starting from a bin type k. When this progresses, if a certain piece in the order of P_u cannot be placed within the selected bin, the next piece in the order is tried for insertion. The process continues until all the pieces in P_u are tried to fill the bin in a feasible manner. Then the utilisation of the bin is recorded. The algorithm follows the same procedure for one bin from each bin type, and for each different bin, record the utilisation it can have when pieces are inserted starting from the initial unpacked piece list. Then the algorithm selects the highest utilised bin of each bin type and considers that bin as the first used bin of the solution. This first bin carries a certain number of pieces and those pieces are considered as packed pieces. Once this is done, the other remaining pieces are updated as unpacked pieces.

With the updated unpacked list, the algorithm then tries to find the next bin to allocate those some pieces in the unpacked list following the same procedure, by testing pieces with the bin types given. Similarly, the process continues until pieces in the unpack list become empty. Algorithm 7 describes how this procedure works. This algorithm constructs a bin packing solution for a given permutation of pieces and determine the bin order to be set in the solution through a dynamic bin selection mechanism runs inside of it.

Algorithm 6: IJRAB

```
Input  : Input Bin types $k = \left\{1, \ldots, N^{'}\right\}$, Placement Rule $MU$, Piece kick parameter ($K_p$),
         Bin kick parameter ($K_b$)
Output: $s^{**} = \{B_k^{s^{**}} | \; k = 1, \ldots, N'\}$
Set $t = 1$, $D^* = L$, Initialize the count of piece kicks and bin kicks $q_p = 0$, $q_b = 0$ ;
Set $P^{(L,t)}$ as a random order of pieces, Set $\tau^{(t)}$ as a random order of bins;
Set $f(s^*_{D*})) = -\inf$, Set $f(s^{**}_{D**})) = -\inf$;
while $t \leq T_{max}$ do
    Generate solution layout $s^{(L,t)}$ from $P^{(L,t)}$ and $\tau^{(t)}$;
    Evaluate $f(s^{(L,t)})$;
    Derive $P^{(R,t)}$ from the solution;
    Generate solution layout $s^{(R,t)}$ from $P^{(R,t)}$ and $\tau^{(t)}$;
    Evaluate $f(s^{(R,t)})$;
    if $f(s^{(L,t)}) > f(s^{(R,t)})$ then
        if $f(s^*_{D*}) < f(s^{(L,t)})$ then
            Set $s^*_{D*} = s^{(L,t)}$;
            $D^* = L$;
            Reset $q_p = 0$;
        else
            $q_p = q_p + 1$;
        end
    else
        if $f(s^*_{D*}) < f(s^{(R,t)})$ then
            Set $s^*_{D*} = s^{(R,t)}$;
            $D^* = L$;
            Reset $q_p = 0$;
        else
            $q_p = q_p + 1$;
        end
    end
    if $f(s^{**}_{D^{**}}) < f(s^*_{D^*})$ then
        Set $s^{**}_{D^{**}} = s^*_{D^*}$;
        $D^{**} = D^*$;
        Reset $q_p = 0$;
    end
    if $q_b < K_b$ then
        if $q_p > K_p$ then
            Apply piece kick to the piece permutation of $s^*_{D^*}$ and obtain a new piece
              permutation;
            Geneate solution layout from the new piece permutation;
            Derive next corresponding $P^{(L,t)}$ from the solution for jostling;
            $q_b = q_b + 1$;
            Reset $q_p = 0$;
        else
            Derive $P^{(R,t)}$ from the solution;
        end
    else
        Apply bin kick to the bin sequence of $s^{**}_{D^{**}}$ and update $\tau^{(t)}$;
        Obtain the piece permutation of $s^{**}_{D^{**}}$ ;
        Derive next corresponding $P^{(L,t)}$ from the solution for jostling;
        Reset $q_b = 0$, $q_p = 0$;
    end
    $t = t + 1$;
end
return $s^{**}_{D^{**}}$;
```

When packing pieces to a bin, we adapt the features of the constructive method discussed in Chapter 4. Since we pack pieces to only one bin at a time, the constructive method described in Chapter 4 is implemented in a way that the pieces are packed only for one bin space rather considering them as the sequence of bins at once.

Algorithm 7: Constructive algorithm for SPBCH

```
Input  : List of Bin Types $k = 1, ..., N^{'}$, List of Pieces $P$, Placement Rule $MU$
Output: a feasible packing of the pieces $P^*$ into the bins $B^*$ and $U^*$
1  $P_u = P$;
2  Initialize Used bin index $j = 1$;
3  while ($P_u$ is not empty) do
4      for Each bin type $k$ do
5          for Each PIECE in $P_u$ do
6              check for a feasible placement of PIECE in $b_{k,j}$;
7              if PIECE placement is feasible then
8                  pack temporary the PIECE inside $b_{k,j}$;
9              end
10         end
11         Get the bin utilization $u_k$;
12         Record the PIECEs layout
13     end
14     Select the $b_{k,i}$ corresponds for maximum $u_k$;
15     Select $b_{k,j}$ and assign as $b^*_{k,j}$ with PIECEs layout;
16     Update $P_u$ list by removing the packed PIECEs in $b_{k,j}$ from the list;
17     $j = j + 1$;
18 end
19 return packing layout $P^*$ in $B^*$, $U^*$;
```

5.4.5.2 Solution improvement

The solutions generated in section 5.4.5.1 are improved by implementing a local search mechanism. We choose an initial solution S using a random permutation of the pieces and compute the evaluation value of S (denoted as $f(S)$) as described in Section 5.3.1. Initially, we assign $S^* = S$ and $U^* = U$. During the search process, we generate a neighbour solution $S^{'}$ and compute $f(S^{'})$ at each iteration. If the evaluation value of new neighbour is better, then $S^{'}$ and $U^{'}$ replaces the current best S^* and U^*. Otherwise, S^* is retained as the current best solution. The proposed local search improves the current solution until reaching a local optimum when there is no further improvement for a pre-defined number of iterations (c_{max}). In order to generate a neighbour solution, we propose following piece move mechanism.

Since the search occurs over the sequence of the pieces, each iteration generates a neighbour sequence by a *swap move* (Bennell and Oliveira, 2009) of two random pieces of the current best solution's input piece order. The constructive algorithm takes this new sequence of pieces and creates the next layout. At each iteration, only one neighbour is generated. The complete solution is evaluated according to the evaluation criteria discussed in section 5.3.1. At the end of the search, the best-recorded solution is kept.

In order to improve the search process, we also propose *insert piece move* (Bennell and Oliveira, 2009) to disrupt the sequence to a greater extent when there is no improvement in solution by the swap moves for a pre-defined number of iterations (c_{max}). This kick is implemented to move the local optimum to a new area of the solution space. We follow one piece insert move to change the input piece order of the current best solution.

5.5 Results and Analysis

5.5.1 Data instances and implementation

For benchmarking, we could not find any instance of the specific problem discussed in this chapter. To our knowledge, computational results of this specific problem are not available in the literature. Therefore, we used the nesting instances published on ESICUP (EURO Special Interest Group on Cutting and Packing) website for our experiments. In order to evaluate the performance of the proposed algorithms, it is tested with following instances as well as the newly created instances in order to match with the irregular bin packing problem. The algorithms were coded in Visual C++ 2012 as a sequential program. All computations were carried out in a personal computer Intel 2.60 GHz (Sandybridge) processor and 4GB RAM. Algorithms contain no parallel processing and can be run using a single processor.

When selecting instances, we considered the production characteristics of sheet cutting industries. In this case, we excluded the instances from garment industry because they are solved as open dimension problems where the dimensions of the stock sheet; usually the length, are not fixed. Since this problem deals with fixed dimension of stock sheets, the garment instances are not considered in computational experiments. Another reason for this exclusion is that garment instances are mostly used with restricted rotation while our problem mainly considers unrestricted rotation of pieces.

Accordingly, 14 irregular shape instances (in Table 5.1) representing both convex and non-convex polygons were used for experiments with 9 different Bin sizes: $0.5 \times d_{max}$, $0.75 \times d_{max}$ $1 \times d_{max}$, $1.25 \times d_{max}$, $1.5 \times d_{max}$, $1.75 \times d_{max}$, $2 \times d_{max}$, $2.25 \times d_{max}$, $2.5 \times d_{max}$, where d_{max} be the maximum value of length or width of each piece in their initial orientation. Table 5.2 describes four different instances of bin configurations; SB, MB, LB and Mix. In SB, we consider packing pieces with a small set of bin sizes. In comparison, the MB and LB represent bin packing with a large set of bin sizes. In each configuration, five different bin types are considered with a small bin size difference at each configuration. Finally, the Mix option considers nine bin types representing remarkably large bin sizes as well as remarkably small bin sizes in the input bin configuration.

We conducted our tests allowing restricted rotation (RR) and unrestricted orientation (UR) of pieces. Since the problem is new, there are no benchmark results comparing the solutions. However, we evaluated the proposed computational method through following a set of tests. We set up all our experiments for 10 trials since the algorithms contain random components and reported average output of 10 trials.

Operational details and parameters of algorithms:

Table 5.1 and 5.2 details the instances and bin configurations used for the experiments.

Table 5.1: Instances

Instance	No.of Pieces	Instance	No.of Pieces
Shapes2	28	2×Jakobs2	50
3×Dighe1	48	Poly3a	45
3×Dighe2	30	Poly3b	45
Poly4a	60	Poly4b	60
3×Fu	36	Poly5a	75
2×Han	46	Poly5b	75
2×Jakobs1	50	Shapes	43

Table 5.2: Bin Configurations

Configuration	No. of Bin types	Input bin sizes	Bin type ID
SB	5	$0.5d_{max}$, $0.75d_{max}$, $1.0d_{max}$, $1.25d_{max}$,$1.5d_{max}$	1,2,3,4,5
MB	5	$1.0d_{max}$, $1.25d_{max}$, $1.5d_{max}$, $1.75d_{max}$,$2.0d_{max}$	3,4,5,6,7
LB	5	$1.5d_{max}$, $1.75d_{max}$, $2.0d_{max}$, $2.25d_{max}$,$2.5d_{max}$	5,6,7,8,9
Mix	9	$0.5d_{max}$, $0.75d_{max}$, $1.0d_{max}$, $1.25d_{max}$,$1.5d_{max}$	1,2,3,4,5
		$1.75d_{max}$, $2.0d_{max}$, $2.25d_{max}$, $2.5d_{max}$	6,7,8,9

5.5.2 Performance of two constructive procedures (CA1 vs. CA2)

We compare the performance of CA1 and CA2 algorithms presented in Section 5.4.1 and 5.4.5.1 in Table 5.3. For each instance, an average U value was calculated for 10 trials. The values represent average results computed over the 14 instances. In each trial, we ran each algorithm for 800 seconds under following criteria and consider the best-recorded solution.

Rand-CA1:	Running CA1 with different random orders of bins and random orders of pieces and pack pieces
Rand-CA2:	Running CA2 with different random orders of pieces and pack pieces

Table 5.3: Performance of construction algorithms

Bins Config.		Rand-CA1		Rand-CA2	
		RR	UR	RR	UR
Mix	*Avg.U* of 10 trials	0.678	0.682	0.682	0.688
	Avg. Time (seconds) for 100 solutions	*34.66*	*51.23*	*176.52*	*245.54*
LB	*Avg.U* of 10 trials	0.665	0.675	0.668	0.677
	Avg. Time (seconds) for 100 solutions	*37.67*	*56.44*	*141.36*	*188.40*
MB	*Avg.U* of 10 trials	0.663	0.671	0.667	0.674
	Avg. Time (seconds) for 100 solutions	*32.78*	*47.15*	*98.41*	*162.49*
SB	*Avg.U* of 10 trials	0.650	0.657	0.656	0.665
	Avg. Time (seconds) for 100 solutions	*29.52*	*41.53*	*74.68*	*120.04*

Considering the performance of both constructive procedures against the randomly generate permutations, *Rand-CA2* performs slightly better in overall utilisation (U) within 800 seconds for all the data sets. Also, the better solutions are produced when the piece rotation is not restricted for both CA1 and CA2.

In comparison, CA1 and CA2 demonstrate a significant difference in computational effort per solution. *Rand-CA2* takes a longer execution time than *Rand-CA1* to generate a single solution. The bin selection process of CA2 is a time-consuming task as it checks for the highest utilised bin type by placing pieces temporary, for each given bin type. Since CA1 does not execute such bin section for all the given bin types and instead places the pieces in a pre-assigned order of bins. This makes CA1 to test a higher number of solutions than CA2 in a common runtime.

Out of the different configurations of bins, the Mix option recorded the highest utilisation while the LB option recorded the second highest. Therefore, out of the options using a limited number of bin types (LB, MB, SB), the LB being the best performer even though it consumes higher computational time than others. Usually, with a set of larger bin types, the algorithms allocate a higher number of pieces in one bin than choosing a set of small bin types. This leads to an increase in the computational time for generating some no-fit polygons according to the constructive method discussed in Chapter 4 as it needs to consider the relatively large size of merged pieces for some insertion of pieces.

In the preceding sections, we investigate the performance of CA1 and CA2 procedures when they are applicable with search methods.

5.5.2.1 Best performing GA Jostle approach:

We compared the performance of three different approaches of using GA to solve 2D IMBSBPP. For each instance, an average U value was calculated for 10 trials over the 14 instances. In each trial, we ran the GA for 800 seconds.

GA:	Running the GA approach discussed in without improving the offsprings generated at each generation through Jostle
HGA_v1:	Running the GA approach with Jostle improvement of all the offsprings generated at each generation
HGA_v2:	Running the GA approach with iterated Jostle improvement of the best offspring generated at each generation

The parameter settings for our proposed GA approaches were determined through the results of pilot tests conducted. We compared different values for *PopSize*, P_{Cr}, P_{Mu}, K_p, $MaxJS$ and *elitist selection ratio* (see in Table 5.4) and selected the combination of parameter values that provided the best quality solutions in 800 seconds.

Table 5.4: Different parameter settings to tune GA, HGA_v1 and HGA_v2

$PopSize$:	$\{8, 12, 16, 24, 32\}$
P_{Cr}:	$\{0.7, 0.8, 0.9, 1.0\}$
P_{Mu}:	$\{0.0125, 0.025, 0.05, 0.075, 0.1\}$
K_p :	$\{2, 4, 6, 8, 10\}$
$MaxJS$:	$\{4, 8, 12, 16, 20, 24\}$
$P_{elitist\ ratio}$:	$\{1/4, 1/8, 1/12, 1/16\}$
T_s:	$\{2, 3, 4, 5\}$

Our pilot tests recommend following a set of values to run each GA approach discussed in this study.

GA:	$PopSize = 24$, $MaxJS = no\ Joslte\ improvement$, $P_{Mu} = 0.05$ All parent solutions are crossover to generate offspring solutions No. of solutions by elitist selection : 1/4 of *popSize*, $T_s = 2$
HGA_v1:	$PopSize = 12$, $MaxJS = 8$, $P_{Mu} = 0.025$ All parent solutions are crossover to generate offspring solutions No. of solutions by elitist selection : 1/8 of *popSize*, $T_s = 2$
HGA_v2 :	$PopSize = 16$, $K_p = 4$, $MaxJS = 12$, $P_{Mu} = 0.025$ All parent solutions are crossover to generate offspring solutions No. of solutions by elitist selection : 1/8 of *popSize*, $T_s = 2$

Table 5.5: Performance of GA and HGA approaches for 800 seconds

Bins Config.	Restricted Rot. *Avg. U*			Unrestricted Rot. *Avg. U*		
	GA	HGA_v1	HGA_v2	GA	HGA_v1	HGA_v2
Mix.	0.689	0.689	0.693	0.695	0.694	0.699
LB	0.679	0.677	0.686	0.686	0.685	0.690
MB	0.678	0.675	0.681	0.684	0.679	0.687
SB	0.660	0.653	0.659	0.666	0.658	0.666

Out of the three GA-based approaches, the *HGA_v2* performs the best according to the packing utilisation over the 14 instances. Table 5.5 demonstrates this for all four bin configurations in both scenarios where piece rotation is restricted and unrestricted. The GA approach without Jostle is also recorded the second best results by running the algorithm for a higher number of generations within the given 800 seconds. Not surprisingly, the computational cost of HGA_v1 incurs the highest computation cost and thereby operated with a less number of generations within the given 800 seconds. This cause a lower utilisation than other two approaches. In comparison, HGA_v2 solves the problem within a reasonable computation time and achieves a better utilisation.

5.5.2.2 Performances of SPBCH approaches:

Two different search methods described in Section 5.4.5.2 were evaluated for the 14 instances. For each instance, we executed each algorithm for 10 trials where each trial was run for 800 seconds. Using pilot studies, we found $c_{max} = 15$ as a good parameter value to run SPBCH after conducting parameter tuning with different $c_{max} = 5, 10, 15, 20$ values.

LS_CA2:	Local search with swap moves of pieces
ILS_CA2:	Iterated local search with the swap and insert moves of pieces

Table 5.6: Performance of LS_CA2 and ILS_CA2 approaches for 800 seconds

Bins Config.	Restricted Rot. *Avg. U*		Unrestricted Rot. *Avg. U*	
	LS_CA2	ILS_CA2	LS_CA2	ILS_CA2
Mix.	0.685	0.689	0.695	0.701
LB	0.673	0.681	0.675	0.694
MB	0.677	0.679	0.684	0.691
SB	0.660	0.663	0.663	0.671

Table 5.6 illustrates, there are 0.58%, 1.19%, 0.30% and 0.45% improvement in overall utilization (on average) for Mix, LB, MB and SB bin configurations respectively by ILS_CA2 when the piece rotation is restricted. Also, it records 0.86%, 2.81%, 1.02% and 1.21% improvement in overall utilisation for Mix, LB, MB and SB bin configurations respectively by ILS_CA2 when the piece rotation is unrestricted. This indicates that approach ILS_CA2 approach generates better results and also noticed that there is no significant difference in computation time between both approaches. Also, the Mix option records the best utilisation in comparison to other bin configuration options (LB, MB, SB).

5.5.2.3 Best performing IJRAB approach:

Three different versions of *IJRAB* were evaluated for the 14 instances in Table 5.1. For each instance, we ran each approach for 10 trials due to the random components of the algorithm. In each trial, we executed the algorithm for 800 seconds.

In pilot studies, we tested different values of $K_p = \{2, 4, 6, 8, 10\}$, $K_b = \{2, 3, 4, 5, 10\}$ to tune the algorithm to provide a reasonably better performance. The tuning results demonstrated $K_p = 4$, $K_b = 3$ as good settings to run IJRAB and these settings were used to run the algorithms.

The three versions of IJRAB tested in this study:

In comparison, the proposed IJRAB algorithm is faster than the other two computational methods. This creates an opportunity to plug some other post-processing strategies to improve its performance. We tested the applicability of two such post-processing strategies with the proposed IJRAB algorithm. In Algorithm 6, each of these strategies are applying before the *bin-kick* step executes at Line 42.

- Strategy 1 (S1): Try to repack pieces in the least utilised bin into the smallest possible bin:

This is a well-known strategy suggested for rectangular bin packing problem (Friesen and Langston, 1986; Ortmann et al., 2010). The strategy attempts to change the bin configuration of the solution by repacking pieces inside the least utilised bin, to the smallest possible bin type. If this move improves the solution and is better than the leading solution, then the new arrangement of bins and pieces are accepted as the leading solution. Then $s^{**}_{D^{**}}$, $U^{*}*$, and $\sigma^{*}*$ are updated according to the new solution (see Algorithm 6).

- Strategy 2 (S2): Try to repack pieces in each of the used bins into the smallest possible bin:

  Strategy 1 is focused only on changing the least utilised bin in the order. In this case, the same procedure is extended to repack the pieces in all the used bins separately into the smallest possible bin. The strategy attempts to change the bin configuration of this solution by repacking pieces inside each of the used bin separately, to the smallest possible bin type. Each solution is compared against the leading solution and the leading solution is updated if a better solution is found.

When any of these strategies are able to improve the leading solution, then the immediate next step of making a bin kick will not be executed since the bin sequence is changed by the post-processing strategy.

Considering these two strategies, we compare the results of three versions of IJRAB (see Tables 5.7 and 5.8).

- *IJRAB-A1*: Implementation of IJRAB Algorithm as it is.
- *IJRAB-A2*: Implementation of IJRAB Algorithm by applying S1 for the local optimum found in that particular bin configuration.
- *IJRAB-A3*: Implement Algorithm 1 applying S2 for the local optimum found in that particular bin configuration.

Out of three IJRAB versions, both *IJRAB-A2* and *IJRAB-A3* reach better packing utilisation than *IJRAB-A1* when comparing the average U values over the 14 instances (The best average values are denoted in bold font). The *IJRAB-A2* and *IJRAB-A3* algorithms achieve almost similar packing utilization for MB, LB, and Mix options, even though the version *IJRAB-A3* is computationally expensive than *IJRAB-A2*. Also, when running *IJRAB-A3*, out of all the used bins, only the lowest utilised bin was frequently replaced by the possible smallest bin. The improvement gap between IJRAB-A2 and IJRAB-A3 for MB, LB, and Mix reflect this by showing a minor difference. However, there is an exception for smaller bin size configuration (SB) since it demonstrates a significant improvement. In summary, through this experiment, we identify *IJRAB-A2* and *IJRAB-A3* as the most promising versions of IJRAB.

Table 5.7: Performance of the IJRAB algorithms for different bin configurations (with Restricted Rotation of pieces)

Bin Config.		IJRAB-A1	IJRAB-A2	IJRAB-A3
Mix	Avg. U	0.692	**0.700**	**0.703**
	Improvement %		**1.16%**	**1.59%**
	Time (seconds for 100 Jostle cycles)	*66.5*	*69.2*	*78.2*
LB	Avg. U	0.667	**0.687**	**0.689**
	Improvement %		**3.00%**	**3.30%**
	Time (seconds for 100 Jostle cycles)	*71.6*	*74.6*	*76.8*
MB	Avg. U	0.671	**0.684**	**0.686**
	Improvement %		**1.94%**	**2.24%**
	Time (seconds for 100 Jostle cycles)	*62.7*	*65.8*	*68.0*
SB	Avg. U	0.656	**0.662**	**0.677**
	Improvement %		**0.91%**	**3.20%**
	Time (seconds for 100 Jostle cycles)	*50.5*	*53.1*	*58.6*

Table 5.8: Performance of the IJRAB algorithms for different bin configurations (with Unrestricted Rotation of pieces)

Bin Config.		IJRAB-A1	IJRAB-A2	IJRAB-A3
Mix	Avg. U	0.699	**0.706**	**0.708**
	Improvement %		**1.00%**	**1.29%**
	Time (seconds for 100 Jostle cycles)	99.7	104.6	107.5
LB	Avg. U	0.680	**0.695**	**0.696**
	Improvement %		**2.21%**	**2.35%**
	Time (seconds for 100 Jostle cycles)	*111.5*	*115.0*	*121.1*
MB	Avg. U	0.684	**0.692**	**0.693**
	Improvement %		**1.17%**	**1.32%**
	Time (seconds for 100 Jostle cycles)	*98.2*	*101.3*	*103.0*
SB	Avg. U	0.668	**0.675**	**0.685**
	Improvement %		**1.05%**	**2.54%**
	Time (seconds for 100 Jostle cycles)	*83.6*	*88.0*	*92.8*

5.5.3 Comparison among three computational methods

Table 5.9 summarises the average U values of each computational method separately. In order to compare the results, the best algorithm of each method was run for 800 seconds in one trial. The values represent average results obtained over the 14 instances for 10 trials.

Table 5.9: Performance comparison of IJRAB, HGA and SPBCH methods

	Restricted Rot. *Avg. U*					Unrestricted Rot. *Avg. U*				
Bins Config.	Random CA1	Random CA2	IJRAB _A3	HGA _v2	SPBCH ILS_CA2	Random CA1	Random CA2	IJRAB _A3	HGA _v2	SPBCH ILS_CA2
Mix.	0.678	0.682	**0.703**	0.693	0.689	0.682	0.688	**0.708**	0.699	0.701
Impr. %	-	-	3.69%	2.21%	1.03%	-	-	3.81%	2.49%	1.89%
LB	0.665	0.668	**0.689**	0.686	0.681	0.675	0.677	**0.696**	0.690	0.694
Impr. %	-	-	3.61%	3.16%	1.95%	-	-	3.11%	2.22%	2.51%
MB	0.663	0.667	**0.686**	0.681	0.679	0.671	0.674	**0.693**	0.687	0.691
Impr. %	-	-	3.47%	2.71%	1.80%	-	-	3.28%	2.38%	2.52%
SB	0.650	0.656	**0.677**	0.659	0.663	0.657	0.665	**0.685**	0.666	0.671
Impr. %	-	-	4.15%	1.38%	1.07%			4.26%	1.37%	0.90%

In general, there is a significant improvement with all three search methods. The table also reveals best performance results by IJRAB (_A3) than HGA_v2 and ILS_CA2, in all rotation variants and input bin set configurations.

Out of different input bin configurations, a mixed use of bins with a higher number of bin types (Mix) generates significantly better solutions. A greater variety of input bin configuration creates more opportunities in finding highly utilised bins by replacing poorly utilised bins using smaller bin types.

The three input configurations; SB, MB and LB, with an equal number of input bin types, demonstrate a pattern. The large bin set (LB) achieves higher utilisation whereas small bin set (SB) gets lower utilisation. We observe that the irregular pieces placed in large bin types tend to move, rotate and compact more when there is the higher number of pieces inside the bin. We also noticed that the accommodation of smaller pieces in large bin types leads to a higher utilisation of those bins.

Examining the results, IJRAB_A3 and HGA_v2 perform better than the SPBCH method. This is most likely due to the time advantage of CA1 and the effective moves established during the search process. The results also demonstrate that the piece moves from the Jostle structure have significantly contributed to solution improvement in IJRAB and HGAIJ. Within a reasonable computational effort ($\leq$ 800 seconds), the IJRAB_A3 demonstrates significantly better results than HGA_v2. Since HGA_v2 works with a population of solutions to ensure its bin moves, the overall method is more time-consuming than IJRAB_A3. However, in the long run, both approaches demonstrate less significant differences in their results. In comparison, there is only a slight improvement in the solution quality by IJRAB_A3 vs. IJRAB_A2.

Since our goal is to produce a computational method to generate a good solution within a short period of time, the study involves the most efficient method; the IJRAB_A3 for the next set of experiments. For easy reference, here onward, it is simply referred as IJRAB

5.5.4 Performance comparison: 2D IMBSBPP vs. 2D ISBSBPP

We presented Iterated Jostle algorithm in Chapter 4 to solve 2D ISBSBPP with free rotation of pieces, which has better results in a shorter time than the other approaches Lopez-Camacho et al. (2013a) and Martinez-Sykora et al. (2017). We use this algorithm to solve ISBSBPP problem for each bin type separately and compared the results. Table 5.10 and 5.11 present the average U values of 10 trials of algorithm execution. We ran an algorithm for 800 seconds at each trial.

Table 5.10 and 5.11 demonstrate the advantage of solving 2D IMBSBPP as the results of 2D IMBSBPP outperform the results achievable from 2D ISBSBPP.

We also noticed that, with single bin sizes, most frequently there is one bin in the solution recording poor utilization than others, especially when large bins (e.g. $2.25d_{max}, 2.5d_{max}$) are involved. This results reduction in overall utilisation as well. However using different bin sizes maintaining a certain variability in the sizes cause improvement in utilization by replacing such poorly utilized bins in the solution with a smaller bin type. Also, this finding opens another path to investigate. When there is such poorly utilized bin space, or a sheet space in sheet cutting, taking such residual spaces as a separate stock sheet in the next orders would allow to improve the overall utilization. If a company decides to implement such a post production policy to reuse usable residuals, still, the proposed algorithm for 2D IMBSBPP can be applicable. We dicuss more details of this matter in Chapter 6.

5.5.5 Best solutions

Table 5.12 and Table 5.13 present the best U values we obtained across 10 trials of experiments by executing the three main algorithms; IJRAB_A3, HGA_v2 and ILS_CA2, with 9 input bin types (Mix) and 5 input bins types (LB) respectively. The tables present the number of bins used in the solution, percentage of each bin type used and the computation method which derived those solutions. In addition, the tables demonstrate the average utilisation of bins for each bin type used. As an example, in Table 5.12, the instance *Fu* in RR has used six bins altogether where one bin from bin types 2,3,6,9 and two bins from bin type 8 are used. The use of each of those bin types are presented as percentages. The values within brackets below the percentages indicate the average utilisation of each bin type used in the solution.

By examining the table, we noticed that the best results are achieved by IJRAB_A3 approach in most cases except for one case in Table 5.12 and one case in Table 5.13. In some cases, both IJRAB_A3 and HGA_v2 generated the same solution. Also, it demonstrate that IJRAB_A3 reaches good solutions quicker than the other two approaches.

Table 5.10: Performance comparison: 2D IMBSBPP vs.2D ISBSBPP (with restrictions of piece rotation)

Instance	Single bin sizes							Multiple bin sizes (IJRAB_A3)			
	(1.0d_{max}**)**	**(1.25**d_{max}**)**	**(1.5**d_{max}**)**	**(1.75**d_{max}**)**	**(2.0**d_{max}**)**	**(2.25**d_{max}**)**	**(2.5**d_{max}**)**	**SB**	**MB**	**LB**	**MIX**
Shapes2	0.589	0.518	0.576	0.605	0.648	0.640	0.518	0.662	0.671	0.690	0.698
3×Dighe1	0.689	0.630	0.612	0.562	0.574	0.680	0.551	0.689	0.705	0.701	0.713
3×Dighe2	0.680	0.653	0.680	0.666	0.510	0.605	0.490	0.727	0.712	0.737	0.730
3×Fu	0.592	0.589	0.670	0.677	0.691	0.655	0.663	0.772	0.762	0.745	0.788
2×Han	0.621	0.681	0.662	0.608	0.621	0.490	0.397	0.703	0.728	0.730	0.729
2×Jakobs1	0.681	0.713	0.778	0.667	0.613	0.605	0.653	0.799	0.800	0.805	0.811
2×Jakobs2	0.660	0.676	0.586	0.574	0.566	0.521	0.563	0.693	0.699	0.690	0.709
Poly3a	0.607	0.582	0.647	0.594	0.607	0.479	0.582	0.616	0.645	0.660	0.660
Poly3b	0.602	0.578	0.642	0.589	0.602	0.475	0.578	0.625	0.654	0.666	0.674
Poly4a	0.571	0.565	0.616	0.634	0.607	0.639	0.518	0.671	0.656	0.619	0.695
Poly4b	0.610	0.586	0.581	0.598	0.572	0.603	0.488	0.690	0.672	0.639	0.690
Poly5a	0.578	0.597	0.674	0.660	0.607	0.599	0.647	0.602	0.647	0.665	0.661
Poly5b	0.604	0.632	0.690	0.591	0.679	0.536	0.579	0.628	0.665	0.678	0.674
Shapes	0.543	0.521	0.603	0.532	0.509	0.536	0.434	0.609	0.593	0.614	0.608
Avg.	0.616	0.609	0.644	0.611	0.600	0.576	0.547	0.678	0.686	0.689	0.703

Table 5.11: Performance comparison: 2D IMBSBPP vs.2D ISBSBPP (No restriction of piece rotation)

Instance	Single bin sizes							Multiple bin sizes (IJRAB_A3)			
	(1.0d_{max})	**(1.25d_{max})**	**(1.5d_{max})**	**(1.75d_{max})**	**(2.0d_{max})**	**(2.25d_{max})**	**(2.5d_{max})**	**SB**	**MB**	**LB**	**MIX**
Shapes2	0.617	0.518	0.576	0.605	0.648	0.640	0.691	0.656	0.640	0.692	0.699
3×Dighe1	0.689	0.630	0.612	0.562	0.574	0.680	0.551	0.701	0.705	0.703	0.726
3×Dighe2	0.680	0.653	0.680	0.666	0.510	0.605	0.490	0.714	0.718	0.734	0.734
3×Fu	0.663	0.589	0.737	0.773	0.691	0.655	0.663	0.755	0.760	0.772	0.778
2×Han	0.677	0.681	0.662	0.608	0.621	0.490	0.596	0.705	0.734	0.731	0.740
2×Jakobs1	0.681	0.784	0.778	0.800	0.766	0.807	0.653	0.788	0.801	0.805	0.811
2×Jakobs2	0.660	0.676	0.670	0.689	0.660	0.695	0.563	0.679	0.689	0.694	0.706
Poly3a	0.607	0.582	0.647	0.594	0.607	0.479	0.582	0.630	0.665	0.663	0.674
Poly3b	0.602	0.660	0.642	0.589	0.602	0.475	0.578	0.641	0.663	0.681	0.691
Poly4a	0.607	0.621	0.616	0.634	0.607	0.639	0.518	0.686	0.678	0.644	0.689
Poly4b	0.610	0.651	0.581	0.598	0.572	0.603	0.488	0.692	0.684	0.641	0.696
Poly5a	0.607	0.647	0.674	0.660	0.607	0.599	0.647	0.636	0.670	0.684	0.676
Poly5b	0.639	0.632	0.690	0.591	0.679	0.536	0.579	0.690	0.677	0.688	0.683
Shapes	0.543	0.579	0.603	0.532	0.509	0.536	0.434	0.618	0.615	0.607	0.604
Avg.	0.634	0.636	0.655	0.636	0.618	0.603	0.574	0.685	0.693	0.696	0.708

Table 5.12: Best solutions received in 10 trials by running algorithm for 800 seconds -(with Mix set of bins)

	Instance	n	U	Number of bins	Percentage of each type of bin is used									Method
					1	2	3	4	5	6	7	8	9	
RR	Shapes2	28	0.718	5			20% [0.725]				60% [0.723]	20% [0.705]		IJRAB_A3, HGA_v2
	3×Dighe1	48	0.725	3			33.3% [0.718]		33.3% [0.650]				33.3% [0.753]	IJRAB_A3, HGA_v2
	3×Dighe2	30	0.742	3			66.7% [0.734]						33.3% [0.744]	IJRAB_A3, HGA_v2
	3×Fu	36	0.789	6		16.7% [0.782]	16.7% [0.765]			16.6% [0.808]		33.3% [0.812]	16.7% [0.747]	IJRAB_A3
	2×Han	46	0.740	3				33.3% [0.759]	33.3% [0.763]				33.3% [0.727]	IJRAB_A3, HGA_v2
	2×Jakobs1	50	0.838	5		20% [0.780]	20% [0.962]				40% [0.785]	20% [0.904]		IJRAB_A3
	2×Jakobs2	50	0.719	6		33.3% [0.762]			33.3% [0.665]		16.7% [0.730]	16.7% [0.748]		IJRAB_A3
	Poly3a	45	0.669	4		25% [0.650]	25% [0.648]			25% [0.647]			25% [0.690]	IJRAB_A3, HGA_v2
	Poly3b	45	0.688	3	33.3% [0.580]						33.3% [0.684]		33.3% [0.694]	IJRAB_A3, HGA_v2
	Poly5a	75	0.672	4				25% [0.540]			25% [0.717]		50% [0.674]	IJRAB_A3, HGA_v2
	Poly5b	75	0.684	4		25% [0.638]					25% [0.700]	25% [0.658]	25% [0.701]	IJRAB_A3, HGA_v2
	Shapes	43	0.612	3						33.3% [0.486]	33.3% [0.715]		33.3% [0.607]	IJRAB_A3
UR	Shapes2	28	0.718	5			20% [0.717]				60% [0.724]	20% [0.705]		IJRAB_A3
	3×Dighe1	48	0.720	3	33.3% [0.694]					33.3% [0.742]			33.3% [0.709]	HGA_v2
	3×Dighe2	30	0.753	2						50% [0.708]		50% [0.781]		IJRAB_A3, HGA_v2
	3×Fu	36	0.847	14		42.9% [0.892]	35.7% [0.861]			14.3% [0.868]		7.1% [0.779]		IJRAB_A3, HGA_v2, LS-CA2
	2×Han	46	0.740	3				33.3% [0.759]	33.3% [0.763]				33.3% [0.727]	IJRAB_A3, HGA_v2
	2×Jakobs1	50	0.852	3						33.3% [0.853]		33.3% [0.852]	33.3% [0.850]	IJRAB_A3
	2×Jakobs2	50	0.722	5	40% [1.000]						20% [0.730]	40% [0.704]		IJRAB_A3
	Poly3a	45	0.710	2							50% [0.573]		50% [0.798]	IJRAB_A3, HGA_v2
	Poly3b	45	0.704	2							50% [0.688]		50% [0.715]	IJRAB_A3, HGA_v2
	Poly5a	75	0.696	4						50% [0.696]		25% [0.670]	25% [0.725]	IJRAB_A3, HGA_v2
	Poly5b	75	0.709	3							33.3% [0.661]	33.3% [0.717]	33.3% [0.735]	IJRAB_A3, HGA_v2
	Shapes	43	0.612	4			50% [0.602]					25% [0.632]	25% [0.602]	IJRAB_A3

Table 5.13: Best solutions received in 10 trials by running algorithm for 800 seconds -(with LB set of bins)

	Instance	n	U	Number of bins	Percentage of each type of bin is used					Method
					5	6	7	8	9	
RR	Shapes2	28	0.715	4			50% [0.703]	50% [0.724]		IJRAB_A3
RR	3×Dighe1	48	0.740	2		50% [0.706]			50% [0.756]	IJRAB_A3, HGA_v2
RR	3×Dighe2	30	0.753	2		50% [0.708]		50% [0.781]		IJRAB_A3, HGA_v2
RR	3×Fu	36	0.814	4			25% [0.774]	50% [0.833]	25% [0.808]	IJRAB_A3
RR	2×Han	46	0.727	3	33.3% [0.723]		66.7% [0.727]			IJRAB_A3
RR	2×Jakobs1	50	0.831	3	33.3% [0.853]				66.7% [0.824]	IJRAB_A3
RR	2×Jakobs2	50	0.716	3	33.3% [0.604]				66.7% [0.736]	IJRAB_A3
RR	Poly3a	45	0.677	3	33.3% [0.591]				66.7% [0.739]	IJRAB_A3, HGA_v2
RR	Poly3b	45	0.672	3	33.3% [0.629]				66.7% [0.702]	HGA_v2
RR	Poly5a	75	0.681	4	25% [0.665]	25% [0.611]			50% [0.700]	IJRAB_A3, HGA_v2
RR	Poly5b	75	0.674	4		25% [0.674]	50% [0.669]	25% [0.680]		IJRAB_A3, HGA_v2
RR	Shapes	43	0.612	3		33.3% [0.486]	33.3% [0.715]		33.3% [0.607]	IJRAB_A3
UR	Shapes2	28	0.696	4	25% [0.680]		25% [0.690]		50% [0.695]	IJRAB_A3
UR	3×Dighe1	48	0.671	2			50% [0.695]		50% [0.657]	IJRAB_A3, HGA_v2
UR	3×Dighe2	30	0.753	2		50% [0.708]		50% [0.781]		IJRAB_A3, HGA_v2
UR	3×Fu	36	0.794	5	20% [0.698]	40% [0.742]			40% [0.837]	IJRAB_A3
UR	2×Han	46	0.736	2				100% [0.736]		IJRAB_A3, HGA_v2
UR	2×Jakobs1	50	0.852	3		33.3% [0.853]		33.3% [0.852]	33.3% [0.850]	IJRAB_A3
UR	2×Jakobs2	50	0.678	3		33.3% [0.626]			66.7% [0.693]	IJRAB_A3
UR	Poly3a	45	0.710	2			50% [0.573]		50% [0.798]	IJRAB_A3, HGA_v2
UR	Poly3b	45	0.704	2			50% [0.688]		50% [0.715]	IJRAB_A3, HGA_v2
UR	Poly5a	75	0.691	3				33.3% [0.642]	66.7% [0.711]	IJRAB_A3
UR	Poly5b	75	0.709	3			33.3% [0.661]	33.3% [0.717]	33.3% [0.735]	IJRAB_A3, HGA_v2
UR	Shapes	43	0.612	3		33.3% [0.486]	33.3% [0.715]		33.3% [0.607]	IJRAB_A3

5.6 Concluding Remarks

In this chapter, we investigate on developing a new computational method to find good solutions for 2D IMBSBPP in less time. The objective is to maximise the overall utilisation of bins. This problem is practical in several sheet cutting industries which work efficiently with different standard sheet sizes. When considering both constructive algorithms proposed, based on the computational complexity and solution quality, CA1 demonstrate the most efficient choice. Out of three main improvement methods proposed, the IJRAB_A3 performs better with the suggested improvement strategies applied at frequent intervals while Jostle continues. The proposed IJRAB_A3 is a low-cost solution algorithm and does not require any advanced optimisation packages to implement it in real world problems.

The other promising approach HGAIJ also reaches equal performance to IJRAB at the long run. Main reason for its less performance in short computational times is the design structure of the algorithm. Since it implements as a population based metaheuristic, it consumes a significant time to move construct population of solutions. In comparison, the approach also take significant time to move from one search area to the other in the search space. Compared to IJRAB, the time gap between two improvements of is less at the early stages of a HGAIJ run. This causes considering HGAIJ as a lower performing algorithm during short time periods.

In order to highlight the use of IJRAB algorithm, we compare the use of single sheet (i.e. bin) size vs. a mix of sheet sizes when cutting shapes from a given set of bin types. Based on the results, we summarise the following key findings.

1) Use of mix of bin sizes is worth in terms of material saving than using single bin size approaches.

2) Selection of a set of large bin sizes configuration is more advantageous than the selection of a set of smaller bin size configuration in terms of saving material when there is a choice of selecting from a mix of bin sizes.

The algorithm can be further used to calculate the setups based on the number of bins. Likewise, implementation of this computation method is robust not only to evaluate the trim loss but also to implement different objective functions such as minimising material costs, setup costs or a combination of both, and generate solutions accordingly. This will be our next study in this thesis as we use this efficient computational method to investigate those problems in Chapter 6. We suggest extending this computational method to solve the irregular shape bin packing problem with usable leftovers where heterogeneous rectangular bins are possible in the form of residuals, which can be reusable in future cut orders.

Chapter 6

Use and Reuse of Materials in Cutting Industries

Evaluation of operational policies for effective use of resources

6.1 Introduction

In chapters 4 and 5, we discussed solution methods applicable for 2D irregular shape bin packing problems. The solution of the 2D irregular shape bin packing problem generates a cutting plan that contains a set of bins (i.e. stock sheets) packed with irregular pieces. In the case of reusing residuals, it allows the use of leftovers which are reusable as input stock sheets for the next orders. A residual of a packed bin is considered as a usable leftover if the area of it is larger than a certain threshold area and both length and width of the residual area are longer than a certain threshold length. Otherwise, the residual bin space is considered to be trim-loss which becomes a waste. In a regular production, the Usable Leftovers (ULs) are returned to the warehouse to use them in the future cut orders. In the C&P literature, this problem is called as 2D Irregular shape Bin Packing Problem with ULs (2D IBPPUL) so that the cutting production is run with reusing residuals. As reviewed in Chapter 2, Wäscher et al. (2007)'s typology denotes this problem as 2D Irregular Residual Bin Packing Problem (2D IRBPP), in which the objective is to minimise the material waste.

Instead of throwing away the residual stock sheet, reusing them as an input sheet for future production orders may allow reducing the overall consumption of sheet material throughout the sheet cutting production. In view of operations management, reuse is a well-known production policy in manufacturing businesses. The reuse policy is implemented in different ways within a supply chain, either as product returns; also known as *external returns* (e.g. Minner (2001), French and LaForge (2006)), or *internal returns* which is formed a by-product or a production scrap such as material waste generated by under-utilized processes; which are being able to recover fully or partially (e.g. Fleischmann et al. (1997)). However, the reuse practices have also been stressed due to the extensive effort required in handling and controlling inventory (Fleischmann et al., 1997; Minner, 2003). For instance, reuse of materials creates variability in material return, which causes issues in utilising standard sizes of storage shelves and trucks. This variability generates extra effort in handling and storage, hence resulting extra costs. From a management perspective, the importance of having the best procurement, production and inventory policies in place is emphasised due to the costs attached to these additional factors. To our knowledge, the existing approaches of cutting and packing problem with 2D irregular shapes have not been able to investigate different scenarios of the problem considering varying costs related to material, setups and off-cuts handling. Also, as reviewed in Chapter 2, most research studies that have analysed usable leftovers highlight the significance of conducting an economic analysis to compare the cost of handling leftovers, as future research opportunities (Cherri et al., 2014a). As an example, the authors of those studies believed in keeping usable leftovers in the warehouse are economical for 1DCSP when the material cost is dominant, as the savings in the material can compensate the costs of handling, transportation and warehouse (Trkman and Gradisar, 2007; Cherri et al., 2014a). Similarly, the trade-off between material saving and auxiliary costs need to be analysed with respect to the other cost scenarios dominated by setups and dominated by labour, where cutting and packing of 2D irregular shapes is progressed.

6.1.1 Contribution

This research discusses in this chapter is an exploratory study examining the reuse practices of residuals in sheet cutting industries. There is a lack of evidence in existing research to justify the most suitable operational strategy for a given operational scenario, let alone for a set of scenarios. This chapter provides guidance to make more accurate operational decisions on procurement and production stages within the scenarios where material prices, setup costs and handling costs vary with one another.

In achieving this, we evaluate the suitability of four operational policies in nine different operational scenarios. With regards to saving resources, we present the potential relationships between costs and policies at different operational scenarios. In this Chapter, we investigated two models and designed set of experiments to evaluate the effect of different operational policies. The first model was developed with the objective function of minimising the material waste. This is the most popular objective of existing research attempts when dealing with cutting problems especially when ULs are involved. We considered different standard bin sizes individually (Single) and the mix option of standard bin types (Mix). Section 6.4 provides a brief description of this model. Next, we extended our investigation and discussed the second model with the objective of minimising the total cost. The second model allows us to compare the performances recorded by each policy, involving the other cost components such as setups, inventory and handling costs of ULs when deriving solutions. We denote this as *Model 02:cost saving* in Section 6.5.

In order to run multiple orders, we introduce a computation method to solve the 2D IBPPUL problem. In this case, we present a heuristic search mechanism to solve the problem, by modifying the IJRAB algorithm designed in Chapter 5. The problem requires us to consider two procurement options of using standard stock sheets purchased. The first option is to use a single standard bin size where only a selected type of sheet size is purchased from the sheet supplier/s. The second option is to use multiple standard bin sizes separately, so that cut layouts can have different sizes of standard stock sheets. In solving 2D IBPPUL, the propose two models examine the overall performance of fulfilling the cut demands arising in successive periods. These models also consider the fact of using usable leftovers and managing them efficiently throughout the periods without accumulating them in the warehouse. In order to satisfy this requirement, special attention is made in reviewing different inventory and production scenarios in relation to the residual reuse.

6.2 Overview of Using and Reusing of Material in Sheet Cutting Process

The sheet cutting process occurs in many industries such as sheet metal, paper, tools, and foams. During this process, a known demand of pieces is cut using multiple numbers of sheets within a certain time period. In most cases, the process generates residuals where some of them can be reused. If an organisation has the policy to reuse these residuals than discarding them, it will reduce the material waste. Additionally, if the sheet material is expensive, this will give an additional saving in cost in terms of material. These residuals then can be used as an input material in the proceeding cut orders, in addition to the standard sheet material of those orders. This can be continued until all the orders of the same sheet material are completed. Thus the context of reusing residuals expands the scope of a traditional single-order cutting problem to the multiple consecutive periods cutting problem where residuals from one period may be used in the following period or the periods afterwards.

The general sheet cutting problem can be defined as cutting a set of two-dimensional pieces from a set of rectangular sheets. The cut pieces either can be regular or irregular in shape and the rectangular sheets will have multiple sheets from one standard size or from a mix of various sheet sizes. A cut order consists of a number of pieces assigned to be cut within a certain time period and the time period of each order is fixed. In other words, each order will have different cut quantities to be cut within a fixed time period. The cut demand for the next period is unknown while the demand for the current period is being processed and the smallest piece dimensions are assumed to be known in advance. The usable residuals generated from the current order can be used in the next order/s as input material. The usability of the residuals is determined by its size (i.e. length and width) compared to the smallest available cut piece of all orders. If a residual dimensions are larger than the smallest of all the orders, then these residuals are returned to the inventory, in order to be used in proceeding orders. Such residuals in general, are referred as "Usable Leftovers" (ULs) in cutting and packing literature. The residuals smaller than the smallest piece are counted as trim-loss.

Compared to the other studies of irregular packing problems, irregular bin packing with usable leftovers in a new area of research. Considering this in the multi-period scenario is more challenging than solving irregular bin packing problem. In this thesis, we address this problem at a basic level as an initial study. Therefore, we believe assuming future demands are unknown as a considerable assumption which can provide substantial improvement for minimising waste as well as substantial contribution to research in this area. In solving this new problem, we were inspired by work on one-dimensional cutting stock problem with usable leftovers with the multi-period scenario, which are also at very early stage (e.g. Trkman and Gradisar (2007), Cui et al. (2016)). Those papers have also used this assumption and justify that the assumption is reasonable to use.

Research interest in multiple order cutting problems has only gained the attention of a few scholars. The existing methods of material reuse in relation to one-dimensional cutting stock problems are addressed in Gradišar et al. (1999a); Trkman and Gradisar (2007); Cherri et al. (2009); Cui and Yang (2010); Cherri et al. (2013) whereas two-dimensional problems with regular pieces are addressed by Venkateswarlu (2001), Andrade et al. (2014) and Andrade et al. (2016). Trkman and Gradisar (2007) discuss two models for one-dimensional (1D) cutting stock problem where the first model is focused on minimising the trim-loss and second on minimising the cost of material and the cost of returning usable residuals. However, this research has remained focused on operational aspects such as minimising cost exclusively under a selected operational policy. The discussion of this chapter expands the scope of evaluating a set of policies and details of these are presented in section 6.2.1.

Our study begins by identifying operational policies and different scenarios where the sheet cutting process is executed. Policies represent the options which the organisations follow when running the sheet cutting process, whereas scenarios represent the practical situations that organisations face in terms of material, machinery and labour.

6.2.1 Identifying operational policies:

When planning the cutting process, a firm has the option of either purchasing stock sheets from one standard sheet size or as a mix of several standard sheet sizes. The lessons from 1D cases mainly favour mixed sizes with the aim of improving material utilisation. However, our study investigates how the purchasing decision affects the total cost, therefore both the criteria are considered. They are 1) considering a single standard sheet size as the purchasing option 2) considering the mix of standard sheet sizes as the purchasing option. The next factor considered in determining the policy is the decision the *use* or *non-use* of residuals. Combining both factors, we address below four policies during our study:

- Policy 1: Mix bin sizes procurement policy with residual reuse (UL-Mix)
- Policy 2: Mix bin sizes procurement policy with no residual reuse (nonUL-Mix)
- Policy 3: A single bin size procurement policy with no residual reuse (nonUL-Single)
- Policy 4: A single bin size procurement policy with residual reuse (UL-Single)

Based on the operational policy used in the production, the packing problem that will be solved in this study is either a 2D irregular single bin size bin packing problem (2D ISBSBPP) or 2D irregular multiple bin size bin packing problem (2D IMBSBPP) (Wäscher et al., 2007). As an example, Policy 01 involves addressing the multiple bin size problem while Policy 03 addresses the single bin size problem. However, even the Policy 4 uses single bin size, the cutting and packing problem in consideration is a multiple bin size problem because of the use of residuals, which will be in different sizes. A detailed review of these problems can be found in Wäscher et al. (2007). In order to solve these problems, we employ the heuristic search tools developed in this study, which are discussed in Chapter 4 and 5.

6.3 Model for Processing Successive Cut Orders

The problem considers a sequence of equal time periods where each period is assigned with a cut quantity. The model assumes that the cut demand for the next period is unknown, while the demand for the current period is being processed. Therefore, the optimisation of each order is done separately. The usable leftovers generated from the current order are used for the future orders. The cut shapes are heterogeneous and the smallest piece dimensions (w_{min}, l_{min}) are assumed to be known in advance. For the demand quantity of each period, a cutting and packing problem is solved to find a good solution which minimises the trim loss or the total cost. As the input material, either standard stock sheets or existing usable leftovers or both are used to fulfill the demand at each period. In the output of the current order, if the residual material is larger than the smallest piece (evaluated as mentioned in Section 6.3.1), then it is considered as candidate Usable Leftovers which can be returned to the warehouse for future use. However, confirming them as ULs returned to the warehouse is subject to the criterion of managing the ULs in the warehouse. Details of this criterion are described in Section 6.4 and 6.5. The residuals do not return to the warehouse as ULs are considered as trim loss or waste.

6.3.1 Finding candidate usable leftovers

Let l_{min} be the minimum piece length out of all the pieces, and w_{min} be the minimum piece width out of all those pieces. l_{min} and w_{min} are measured when pieces are aligned at their initial rotation of a given instance. Let $l^{(R)}$and $w^{(R)}$ be the length and the width of the residual sheet (bin), after making a horizontal and vertical cut. Notations L_f and W_f denote the length and the width of a standard bin of type f. Similarly, L_g and W_g denote the length and the width of an OldUL bin of type g.

Figure 6.1 illustrates four possible ways of generating usable leftovers from residual space of a packed bin.

We define following three categories of bin types for 2D IBPPUL, adopting the Cui et al. (2016)'s terminology for leftovers defined in solving the 1D Cutting Stock problem with Usable Leftovers (1DCSPUL).

- *Standard bin types*: The stock sheets in standard sizes. During the planned time horizon of completing all order quantities, it is assumed that standard bins are available in unlimited quantity, from each type, in the inventory.
- *Old UL types*: The UL stock sheet types returned to the warehouse by the cut orders processed in earlier periods.
- *New UL types*: The UL return to the warehouse by the cut order processes in the current period.

Within each category, the bins with same dimensions are considered as bins belong to the same type. At the end of each cut order, the *New UL* bins are returned to the inventory and become *Old UL* for the following orders.

The typical constraints of the problem applied to both models discussed in Section 6.4 and Section 6.5 can be outlined as follows. In each period, the quantity to be cut is equivalent to the demand quantity of that period. Also, all the layouts must satisfy the basic overlap and containment constraints of general cutting and packing problem.

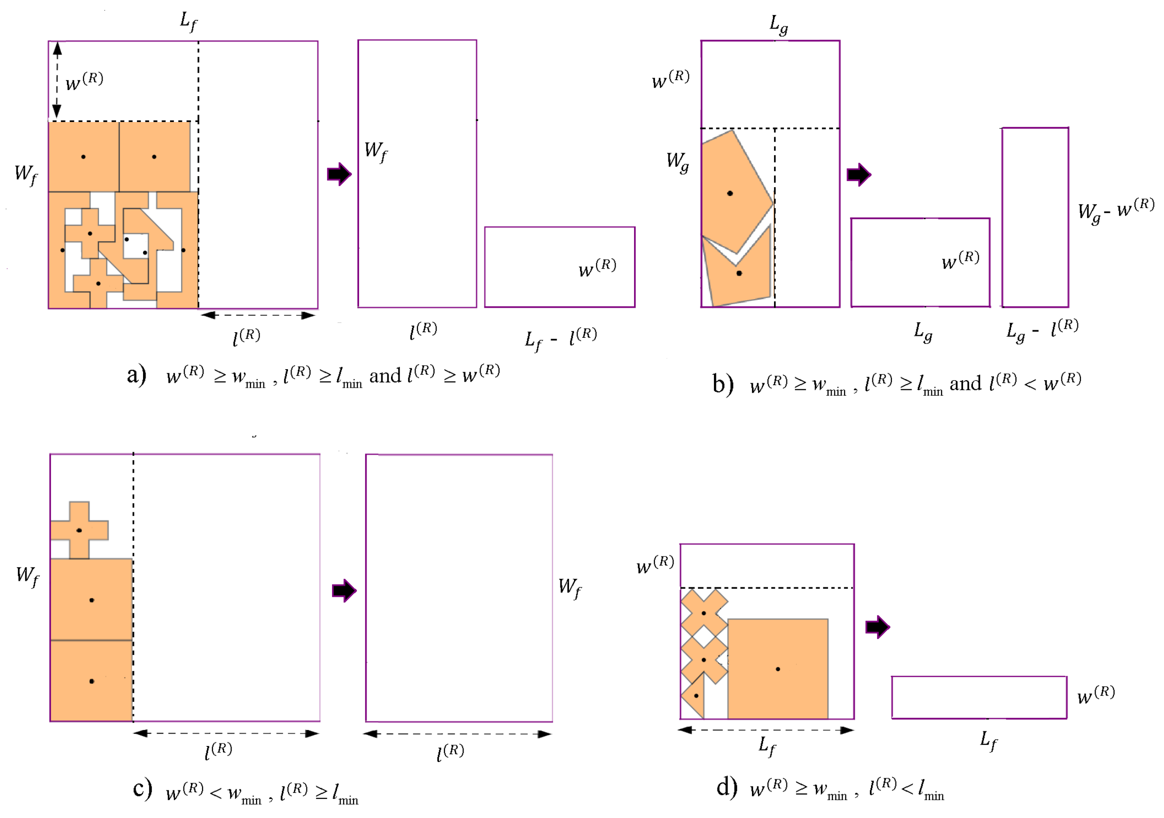

L_f
$w^{(R)}$
W_f
$l^{(R)}$
W_f
$l^{(R)}$
$w^{(R)}$
L_f - $l^{(R)}$
a) $w^{(R)} \geq w_{\min}$, $l^{(R)} \geq l_{\min}$ and $l^{(R)} \geq w^{(R)}$
L_g
$w^{(R)}$
W_g
$w^{(R)}$
W_g - $w^{(R)}$
L_g
L_g - $l^{(R)}$
b) $w^{(R)} \geq w_{\min}$, $l^{(R)} \geq l_{\min}$ and $l^{(R)} < w^{(R)}$
W_f
W_f
$l^{(R)}$
$l^{(R)}$
c) $w^{(R)} < w_{\min}$, $l^{(R)} \geq l_{\min}$
$w^{(R)}$
$w^{(R)}$
L_f
L_f
d) $w^{(R)} \geq w_{\min}$, $l^{(R)} < l_{\min}$

Figure 6.1: Defining usable leftovers

In order to define each model, we use following notations.

6.3.2 Notation

T: Number of time periods considered where t denotes a certain time period so that $t = 1, ..., T$.

Therefore the symbols corresponding to the order processed at time period t are denoted with subscript t:

Inputs when start processing an order quantity n at period t.

P: Pieces where $P = \{p_1, p_2, ...p_n\}$

$B_{f,t}$: Set of Standard bins of bin type f

S'_t : Number of Standard bin types

$N_{f,t}$: Number of Standard bins of type f in the warehouse at time period t

$B_{g,t}$: Set of OldUL bins of type g

$Q_t^{'}$: Number of OldUL bin types in the warehouse at the start of time period t

$N_{g,t}$: Number of OldUL bins of type g in the warehouse at time period t

$N_t^{(Ow)}$: Total number of *Old-UL* bins in the warehouse at the beginning of time period t

Outputs after processing an order quantity n at period t.

$B_{h,t}$: Set of New UL bins of type h

$R_t^{'}$: Number of NewUL bin types return to the warehouse at period t.

$N_{h,t}$: Number of NewUL bins of type h return to the warehouse at time period t

$Q_t^{(O)}$: Number of *Old-UL* bin types used in time period t

$N_t^{(S)}$: Total number of standard bins used in the order processed at time period t.

$N_t^{(O)}$: Total number of *Old-UL* bins used at period t

$N_t^{(R)}$: Total number of *New-ULs* bins return to the warehouse at time period t

Notations used to define the objective functions to solve the problem at period t.

$Area_t^{(Pcs)}$: Total area of demanding pieces in the order processed at time period t

$Area_t^{(S)}$: Total area of standard bins used in the order processed at time period t

$Area_t^{(S)} = \sum_{f=1}^{S_t^{'}} \sum_{j=1}^{N_{f,t}} (c_{f,j,t}.W_f.L_f)$

$Area_t^{(O)}$: Total area of *Old-UL* bins used in the order processed at time period t

$Area_t^{(O)} = \sum_{g=1}^{O_t^{'}} \sum_{j=1}^{N_{g,t}} (d_{g,j,t}.W_g.L_g)$

$Area_t^{(R)}$: Total area of *New-ULs* bins return to the warehouse at the order processed at time period t

$Area_t^{(R)} = \sum_{h=1}^{R_t^{'}} \sum_{j=1}^{N_{h,t}} (W_h.L_h)$

$Trimloss_t$: Trim loss caused in the order processed at time period t

J_{max}: The maximum number of *Old-UL* types

where; $b_{f,j,t}$ and $b_{g,j,t}$ denote the j^{th} bin of standard bin type f and OldUL bin of type g respectively at time period t, and

$$c_{f,j,t} = \begin{cases} 1, & \text{if bin } b_{f,j,t} \text{ is occupied} \\ 0, & \text{otherwise} \end{cases}$$

$$d_{g,j,t} = \begin{cases} 1, & \text{if bin } b_{g,j,t} \text{ is occupied} \\ 0, & \text{otherwise} \end{cases}$$

6.3.3 Problem to be solved at each order:

Since the demand for the future periods is unknown, we do not solve an optimisation problem over the planned time horizon. Instead, the scope of the optimisation problem is limited only for the cut demands to be fulfilled within a period. Therefore, we solve a cutting and packing problem individually for each period, to find a good solution within each period for a given quantity of polygonal shapes (i.e. a heterogeneous set of pieces). However, in both models, the ULs connects each periods by considering the up-to-date ULs stock at each period. At any period, it is possible to use Old UL bin type in the warehouse.

Let $S^{'}$ be a given number of rectangular standard bin types and B_k be a set of bins of standard bin type k where $k = 1, ..., S^{'}$. j denotes the index of a bin of type k so that a bin of type $k : b_{k,j} \in B_k$. Width and length of a bin of type k are denoted by W_k and L_k. Similarly, let $O^{'}$ be a number of rectangular shape leftover bin types made by previous orders. We denote B_l be a set of bins of type l where $l = 1, ..., O^{'}$. j denotes the index of a bin of type l so that a bin of type $l : b_{l,j} \in B_l$. Width and length of a bin of type l are denoted W_l and L_l. The newly generated number of rectangular shape leftover bin types is denoted with $R^{'}$ and B_m is a set of bins of type m where $m = 1, ..., R^{'}$. j denotes the index of a bin of type m so that a bin of type $m : b_{m,j} \in B_m$. Width and length of a bin of type m are denoted W_m and L_m. The problem considers a scenario where the number of bins from each type k is extremely large, number of bins for each type l is finite. The input pieces consist of a set of n polygons $P = p_1, ..., p_n$ equivalent to the demand quantity of the current order.

In each order, an input minimization problem is solved with the objective of minimising trim-loss if Model 01 is used or minimising total cost if Model 02 is used.

A feasible solution can be denoted as a set of occupied standard bins, a set of occupied OldULs, a set of newly generated ULs and the packed pieces. All the pieces in the order are packed exactly to the quantity demanded in the order. Each packed piece p_i, has an orientation angle $o_i \in [0, 360^0)$ and a position in its allocated bin relative to the piece's origin of (x_i, y_i). We considered the reference point of each piece as the bottom-left corner point of the enclosing rectangle for a given angle of rotation and defined the origin as the bottom left corner of the first bin.

6.3.4 Modifications proposed for IJRAB to solve 2D IRBPP

At each period, our approach requires solving an irregular shape bin packing problem with a heterogeneous set of bins in most cases and a set of homogeneous bins otherwise. The input bins can be either standard bins or UL bins which represent the standard sheets or usable leftovers.

When the problem is irregular shape packing with heterogeneous bins, we check the possibility of using IJRAB algorithm discussed in Chapter 5. The original IJRAB solves 2D IMBSBPP when the number of bins from each bin type is not limited. However, with ULs, the number of input bins from each *Old-UL* is finite. Therefore, we proposed the following modifications to the IJRAB algorithm.

Modifying IJRAB to work with a finite number of bins from each bin type:

- The original method initially arranges an arbitrary order of bins to generate a packing layout from left to right along the strip. However, the modified algorithm decides the bin order considering the number of bins available in each type. As an example, if a certain bin type, for instance a bin type 7, has only one bin (an *UL bin* type), then only one bin is allocated from type 7 in to the order. The number of input bins from each *Standard bin* type is assumed to be a large number since we assume no stock-outs of standard bins for each type during the planned time horizon.

- The improvement mechanism suggested in the original work involves two types of kicks during the search process. Out of those two, the *bin kick* is proposed to alter the bin order, in which case, a used bin is replaced with any other randomly selected bin type. In the original algorithm, any bin type other than the selected used bin type can be used as the replacement since the number of bins for each type is not limited. However, in the modified version, the used bin is replaced by another bin type if the new bin type has enough number of bins to use. In fact, this is applied to the UL bin types as their availability is finite at each time period. Therefore, the overall computational method maintains a record of the number of bins available for each type, at each time period.

- The original study suggests a set of post-processing strategies which replace the under-utilised bins of the leading solutions by another bin type focusing an improvement in overall utilisation. Referring to the results in Chapter 5, both IJRAB_A2 and IJRAB_A3 versions performed better with no significant difference in overall utilisations than the other approaches. We decided to involve IJRAB_A2 to solve the problems in this study since IJRAB_A2 is more computationally efficient than the other.

6.4 Model 01: Minimizing the Trim Loss

The trim-loss is the objective measurement to evaluate the quality of a solution and it is evaluated by the equation 6.1.

$$Trimloss_t = Area_t^{(S)} + Area_t^{(O)} - Area_t^{(Pcs)} - Area_t^{(R)} \tag{6.1}$$

The objective is to reduce the trim-loss of the individual orders. The ties in the minimum trim-loss are broken by the solution with the maximum sum of areas of the *New ULs* and the minimum number of different *New UL* types.

When considering usable leftovers, handling a large number of heterogeneous *UL* bins in the warehouse causes additional effort for a company. This imposes managing some practical constraints when handling leftovers. Heterogeneous ULs cause extra effort in managing the inventory level of those leftovers since they can be accumulated in the warehouse when time progresses.

In this model, we use the constraint: $Q_t' \leq J_{max}\ \forall t$. Once New ULs are generated, out of which, the ULs return to the warehouse is determined according to the upper bound J_{max} . At the end of a time period, if the number of New UL types is $\leq$ ($J_{max}-$ Number of OldUL types at the end of the period), then all those generated New UL types return to the warehouse. Otherwise, the number of New ULs is determined based on the sizes of ULs where largest ULs cuts are returned to the warehouse. In this case, we select the largest UL types out of the newly created bin types and the Old UL bin types without violating the constraint J_{max}. Accordingly we set the model to progress with the largest leftovers in the warehouse.

6.4.1 Computational experiments

The model presented in Section 6.4 was tested with a series of experiments to analyse different situations arise in sheet cutting production with irregular-shaped items. All experiments were run by an Intel 2.60 GHz processor and 4GB RAM. In the following sub-sections, we introduce parameter values and sets of benchmark instances used to evaluate the results generated by different production policies.

In terms of computational experiments, each instance represents a certain demand of pieces in one period and there are 30 periods in one trial considering different demand values. Also, for each instance with a demand, in order to find a solution, we ran 300 jostle cycles to minimise the trim loss. Since algorithm contains randomness, the packing algorithm was executed for 10 repetitions to get an average value for each period. Once a trial is finished after 30 periods, we summed up the result to get the total trim loss. Since the demand values of each period are uncertain we conducted 30 trials of 30 periods executions and finalised the average values. This effort was tested for two operational policies; using leftovers and non-using leftovers, for different levels of J_{max}, hence a significant number of experiments were conducted.

Instances: We used the nesting instances published on ESICUP (EURO Special Interest Group on Cutting and Packing) website for our experiments as follows. Fu 60 pieces: (5 x 12 pcs), Shapes2: 56 pieces (2x28 pcs), Jakobs1: 50 pieces (2x25 pcs), Poly: 60 (poly1a 15x4) pieces and Shapes: 86 pieces (43x2).

5 Bin sizes: $1.25 \times m$, $1.5 \times m$, $1.75 \times m$, $2 \times m$, $2.25 \times m$, where m be the maximum value of length or width of pieces of the instance when each piece in their initial orientation.

6.4.1.1 Simulation framework

In order to find the effect on each policy, we designed an empirical simulation framework which connects inputs and outputs of each period within the planned time horizon. The concept of using this framework was adopted by the approaches proposed by Cherri et al. (2013) and Cui et al. (2016), to process successive orders when implementing the residual reuse policy. Cui et al. (2016) solved each order separately since the demand information of the next orders is not known when the current order is being processed. In our simulation framework, we follow the same methodology to solve two-dimensional irregular bin packing problems. Accordingly, at each period, a specific type of irregular shape bin packing problem is solved based on the operational policy as follows.

nonUL-Single :- 2D ISBSBPP

nonUL-Mix :- 2D IMBSBPP

UL-Single :- 2D ISBSBPP at the initial period and then 2D IMBSBPP when ULs are involved

UL-Mix :- 2D IMBSBPP

A certain experiment based on one particular policy was tested in 30 trials. In a given trial, the test was based on a processing cut demands in successive periods to consider the effects in short and long time horizons respectively. Random demand quantities were processed at each time period.

We analysed the effect of using both standard and UL bin types vs. non-use of leftovers when the production is solely focusing on minimising the material used to fulfil the demand quantities. Each of these scenarios was evaluated based on two procurement policies; i) using a single type of standard bins (*Single_S*) or ii) using a mix of standard bin types (*Mix_S*). The "_S" denotes the Standard bins.

One practical issue of using leftover bin types is the inconvenience of handling different types of bins. Due to this reason, we evaluate how different upper bound levels of a number of UL bin types operate with UL-Mix, UL-Single policies.

6.4.1.2 Multi-period run of orders: Period wise investigation

In this test, we consider the performance of nonUL-Mix, UL-Mix, nonUL-Single and UL-Single policies at each time period. The upper bound J_{max} limits the number of residual types returned to the warehouse.

In the analysis, we were interested in the evaluation of the following criteria.

- The variability of total utilisation of material at the end of each period. We use the measurement; cumulative usage of standard material: $U_t = \sum_{t'=1}^{t} Area_{t'}^{(Pcs)} / \sum_{t'=1}^{t} Area_{t'}^{(S)}$. This represents the utilisation of standard material occupied for the sequence of periods. In other words, this explains the amount of material that has been utilised against the amount that the company has paid for at the end of each time period.
- The effects of procurement decisions in the context where the sole objective is to save material as much as possible.

The experiment settings:

We designed a set of experiments to execute the effect of each policy using the instances mentioned in Section 6.4.1. We selected 16 consecutive periods as the number of time periods. In each period, a random demand quantity within the range of [10-50] was taken as the number of pieces to be cut, so that the shapes to be cut are chosen randomly according to the required demand quantity from the instances.

Since we tested the effect of J_{max} values, for the same data instances and the order quantities, we run the tests for different levels of J_{max}= {0 (with no use of ULs), 4, 16, 32}.

At each period, a cut order problem was solved using the modified IJRAB algorithm which terminated after 300 Jostle cycles.

The summarized results are demonstrated in Figure 6.2 and details are given in Tables A.1, A.2, A.3, A.4, A.5, and A.6. In each table, the first data column shows the periods from 1 to 16, second to fifth columns show the U_t values at the end of each t period when $J_{max} = 32, 16, 4$ and 0 (i.e. there is no allowance to use ULs).

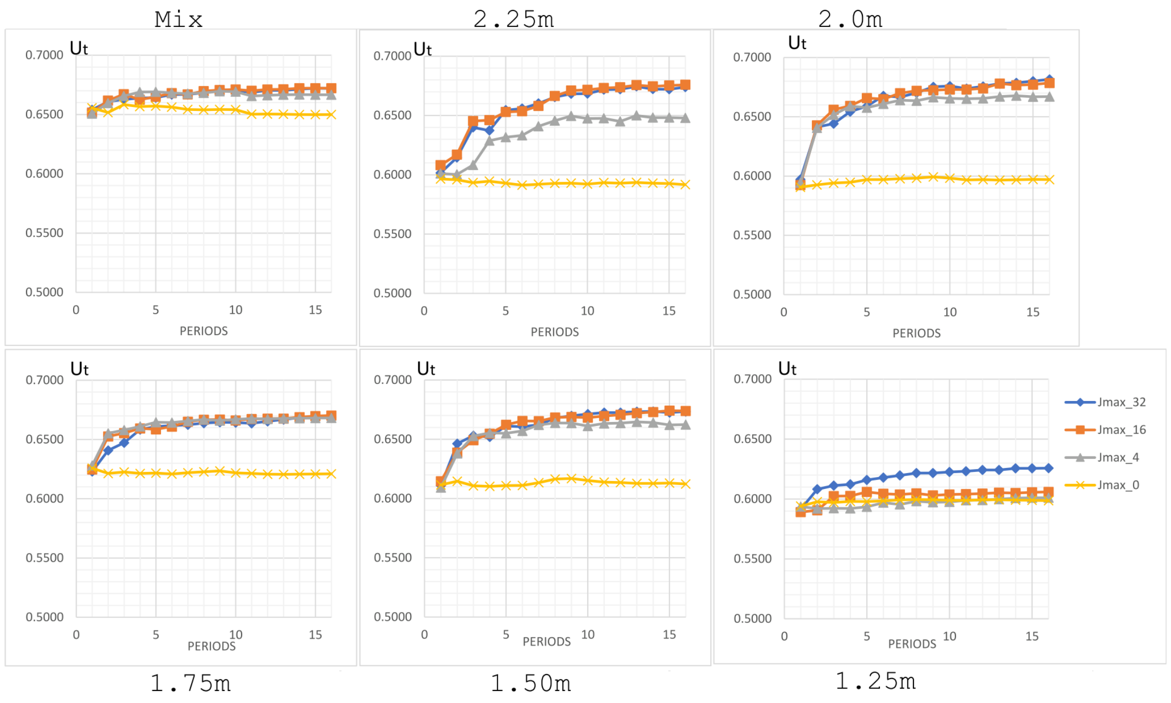


Figure 6.2: Variation of U_t with periods

6.4.1.3 Assessing the results

As illustrated in Figure 6.2, in period wise, the overall utilization till period t (U_t) has neither significant improvement nor drop for nonUL policies (see the Yellow lines correspond for Jmax_0 in each graph of Figure 6.2). The Mix_S option maintains the highest values for utilisation of material than the other Single_S (2.25m to 1.25m) options when no ULs are allowed (Jmax_0); throughout the periods.

However, when UL policies are implemented, the U_t increases when time progresses, especially for the cases starting with Single_S with large bin options (see the Blue, Red, Grey lines correspond for Jmax_32, Jmax_16, Jmax_4 in 2.25m and 2.0m graphs of the Figure 6.2). Higher the area of the starting single bin type, steeper the improvement of U_t gets. When starting with Mix standard bin sizes, comparatively only a little improvement has been achieved with UL policy as the periods progresses. As a special note, interestingly, after a certain period (in this case after the 7th period); the option which starts with large bin sizes reaches higher U_t values than the U_t values generated by Mix bin sizes; when the UL policy is adopted. Also, starting with small bin sizes does not demonstrate promising improvement in U_t as starting with large bin size options or mix bin size options.

The graphs in Figure 6.2 demonstrate reaching to a steady state after a certain period. When analysing this at different J_{max} values, it demonstrates that the higher the value of J_{max}, the period they reach to the steady states becomes longer. Therefore, when deriving conclusions, it is necessary to check whether the true effects of ULs are taken into account properly after it reaches to the steady state. As an example, a short time horizon would not favour UL-Single or UL-Mix policies since the true advantage of using UL for saving material can only be demonstrated after reaching the steady state and this requires a considerable number of periods. When there is a substantial time horizon to reach the steady state, comparatively then the UL-Single policy option starting with the largest single standard bin type demonstrated more advantage in utilising input material by using residuals more effectively than other policies.

6.4.1.4 Multi-period run of orders: Investigation for whole time horizon

Next, we describe the overall results (i.e. The overall performance of running multi-period cutting during the planned time horizon) for the *Model 01* by processing demands for consecutive periods. We conducted the tests for two planned time horizons separately; first for 10 time-periods and then for 30 time-periods.

Table 6.1 and 6.2 demonstrate total usage of purchasing bins ($\sum Area^{(S)}$), the total space of remaining ULs at the end of the time horizon ($RemArea^{(UL)}$) and total trim-loss from each period ($\sum Trimloss$). In this study, the scope of our analysis was based on a certain time horizon and we did not consider the continuity of operations and material for the next periods in the next time horizon. Therefore the UL material remains at the end of the time horizon, was assumed as a waste of material in our analysis. This implies that the total waste is $\sum Trimloss + RemArea^{(UL)}$ in the considered time horizon.

In the tables, the results correspond to the nonUL-Mix and nonUL-Single policies are denoted when $j_{max} = 0$. For UL-Mix and UL-Single policies, the tables show the total area of ULs return to the warehouse (in $\sum Area_t^{(R)}$) and used the area of those returned ULs ($\sum Area_t^{(O)}$) for different $j_{max} = 2, 4, 8, 16, 32$ values. The column headers $N^{(S)}, N^{(O)}$ and $N^{(R)}$ denote the total number of bins of standard bins used, Old UL bins used and New UL bins returned to the warehouse respectively. The overall material utilisation is calculated for each instance as;

$$Utilization = (\sum Area^{(S)} - (\sum trimloss + RemArea^{(UL)}))/\sum Area^{(S)}.$$

The experiment settings:

We designed a set of experiments to execute the effect of each policy using the instances mentioned in Section 6.4.1 and followed the simulation framework discussed in Section 6.4.1.1. As the number of time periods, we ran the tests for two sets of consecutive periods; 10 and 30.

In a certain trial of the multi-period run, a random demand quantity within the range of [10-50] was taken as the number of pieces to be cut at each period, so that the shapes to be cut are chosen randomly according to the required demand quantity from the instances. The outcomes provided by the different J_{max} values are tested for the same data instances and the order quantities. In this set of tests we considered $J_{max} = \{0$ (with no use of ULs), 2, 4, 8, 16, $32\}$. Similar to the previous tests, at each period, an irregular bin problem was solved using the modified IJRAB algorithm which terminates after 300 Jostle cycles.

The summarized results are shown in Tables 6.1 and 6.2.

Table 6.1: Material Saving Scheme: Use of leftovers with different J_{max} values for 10 periods

J_{max}	Purchase	$\sum Area^{(S)}$	$\sum Area^{(O)}$	$\sum Area^{(R)}$	$\sum Trimloss$	$Rem.Area^{(UL)}$	$N^{(S)}$	$N^{(O)}$	$N^{(R)}$	$Utilization\%$	$\sum Area^{(R)}/\sum Area^{(S)}$	$\sum Area^{(O)}/\sum Area^{(R)}$
	Decision	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg %.	Avg %.
0	Mix_ S	21943.1	0.0	0.0	7122.1	0.0	59.5	0.0	0.0	66.86%	0.00%	0.00%
	sgl_S2.25m	23853.6	0.0	0.0	9032.6	0.0	39.7	0.0	0.0	61.28%	0.00%	0.00%
	sgl_S2.00m	23214.6	0.0	0.0	8393.6	0.0	49.5	0.0	0.0	62.66%	0.00%	0.00%
	sgl_S1.75m	22974.8	0.0	0.0	8153.9	0.0	64.1	0.0	0.0	63.32%	0.00%	0.00%
	sgl_S1.50m	23360.8	0.0	0.0	8539.8	0.0	91.6	0.0	0.0	61.84%	0.00%	0.00%
	sgl_S1.25m	25602.7	0.0	0.0	10781.8	0.0	146.7	0.0	0.0	58.25%	0.00%	0.00%
2	Mix_ S	22012.4	243.9	555.5	6879.9	311.5	61.4	2.3	6.8	66.65%	2.83%	42.00%
	sgl_S2.25m	21974.3	946.9	1324.7	6775.6	377.8	37.6	4.4	7.5	66.58%	6.81%	68.04%
	sgl_S2.00m	22159.3	944.7	1407.6	6875.4	462.9	47.2	4.8	7.5	66.28%	6.82%	60.65%
	sgl_S1.75m	21726.3	898.5	1209.1	6594.8	310.5	61.5	5.4	8.2	66.75%	5.99%	69.69%
	sgl_S1.50m	22733.1	487.5	1067.5	7332.2	579.9	89.7	3.6	11.1	64.09%	4.97%	45.79%
	sgl_S1.25m	25291.0	426.8	758.5	10138.3	331.7	145.4	4.2	11.2	59.37%	3.30%	49.65%
4	Mix_ S	21897.9	408.8	757.8	6728.0	348.9	60.9	4.1	9.6	66.98%	3.81%	51.95%
	sgl_S2.25m	21836.3	1376.5	1719.2	6672.6	342.7	37.3	7.3	11.6	67.03%	8.57%	74.21%
	sgl_S2.00m	22036.0	1348.6	1874.1	6689.5	525.5	47.0	6.9	10.5	66.53%	9.52%	66.44%
	sgl_S1.75m	21603.6	1236.9	1563.2	6456.3	326.3	61.5	8.1	12.6	67.03%	7.87%	72.41%
	sgl_S1.50m	22669.3	651.2	1380.2	7119.3	729.0	89.6	5.6	17.4	64.34%	6.52%	49.74%
	sgl_S1.25m	25611.1	527.6	1259.2	10058.5	731.6	146.6	5.4	19.7	59.04%	4.94%	44.74%
8	Mix_ S	21921.0	474.4	918.0	6656.4	443.6	61.6	4.8	12.5	67.00%	4.95%	51.05%
	sgl_S2.25m	21812.2	1522.3	1924.8	6588.6	402.6	37.3	8.2	13.9	67.36%	9.76%	76.47%
	sgl_S2.00m	22012.9	1450.1	2039.8	6602.2	589.8	46.7	8.6	13.2	66.64%	9.87%	66.67%
	sgl_S1.75m	21607.2	1266.9	1637.2	6415.9	370.3	61.8	8.8	14.6	67.03%	8.57%	70.96%
	sgl_S1.50m	22700.1	779.1	1669.4	6988.8	890.3	89.9	6.6	22.8	64.32%	8.05%	51.28%
	sgl_S1.25m	25831.8	511.4	1582.8	9939.4	1071.4	147.2	5.9	30.8	58.88%	6.65%	44.18%
16	Mix_ S	21847.8	588.9	1023.3	6592.4	434.4	61.0	5.7	14.3	67.11%	5.54%	54.99%
	sgl_S2.25m	21794.2	1613.7	1963.4	6623.5	349.7	37.1	8.9	15.1	67.33%	9.99%	78.77%
	sgl_S2.00m	21907.4	1489.5	2072.4	6503.5	582.9	46.8	9.1	13.8	67.09%	9.94%	64.03%
	sgl_S1.75m	21661.5	1314.4	1760.5	6394.5	446.0	62.0	9.5	17.3	66.95%	9.71%	69.27%
	sgl_S1.50m	22758.9	849.1	1877.6	6909.4	1028.5	90.2	7.4	25.3	64.15%	8.52%	52.21%
	sgl_S1.25m	26019.8	602.2	2227.1	9573.8	1625.0	147.9	6.9	50.4	58.70%	9.40%	44.30%
32	Mix_ S	21862.1	603.6	1112.0	6532.6	508.5	61.3	5.9	15.5	67.13%	6.18%	54.06%
	sgl_S2.25m	21773.7	1632.1	1993.1	6591.7	361.0	37.2	8.3	14.5	67.52%	10.01%	79.05%
	sgl_S2.00m	21972.1	1479.5	2073.0	6557.6	593.5	46.6	8.9	13.3	67.06%	10.22%	68.81%
	sgl_S1.75m	21713.6	1228.7	1717.9	6403.4	489.2	62.1	8.7	17.1	66.63%	9.29%	65.93%
	sgl_S1.50m	22816.7	855.8	2056.5	6795.0	1200.7	90.7	7.2	30.6	63.96%	9.43%	51.40%
	sgl_S1.25m	26162.7	768.0	2912.9	9196.9	2144.9	148.5	8.3	66.0	58.65%	11.33%	46.37%

6.4.1.5 Assessing the results for the planned time horizon

Implementation of non-UL policies

For both time horizons as illustrated in Figure 6.3, the nonUL-Mix recorded the highest utilisation of material than any of nonUL-Single when ULs are not allowed ($J_{max} = 0$). In this case, the implementation of nonUL-Single has done in five ways correspond to the five different standard bin sizes considered. However, in terms of the number of bins used, the nonUL-Mix has used a considerably higher number of bins than the nonUL-Single policy with the two largest bin-sizes (see the number of Standard bins used when $J_{max} = 0$ in Tables 6.1 and 6.2).

Since a higher number of occupied bins leads to a higher number of setups with an order, the advantage gained from material saving in Mix bin policy may have been traded-off

Table 6.2: Material Saving Scheme: Use of leftovers with different J_{max} values for 30 periods

J_{max}	Purchase Decision	$\sum Area^{(S)}$ Avg.	$\sum Area^{(O)}$ Avg.	$\sum Area^{(R)}$ Avg.	$\sum Trimloss$ Avg.	$Rem.Area^{(UL)}$ Avg.	$N^{(S)}$ Avg.	$N^{(O)}$ Avg.	$N^{(R)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(R)}/\sum Area^{(S)}$ Avg %.	$\sum Area^{(O)}/\sum Area^{(R)}$ Avg %.
0	Mix_ S	65374.3	0.0	0.0	21283.4	0.0	177.1	0.0	0.0	66.62%	0.00%	0.00%
	sgl_S2.25m	71188.8	0.0	0.0	27097.9	0.0	118.2	0.0	0.0	61.23%	0.00%	0.00%
	sgl_S2.00m	69011.8	0.0	0.0	24920.9	0.0	147.0	0.0	0.0	62.68%	0.00%	0.00%
	sgl_S1.75m	68099.5	0.0	0.0	24008.6	0.0	190.3	0.0	0.0	63.44%	0.00%	0.00%
	sgl_S1.50m	69430.7	0.0	0.0	25339.8	0.0	272.2	0.0	0.0	61.94%	0.00%	0.00%
	sgl_S1.25m	76169.6	0.0	0.0	32078.7	0.0	436.1	0.0	0.0	58.23%	0.00%	0.00%
2	Mix_ S	65344.4	826.1	1566.9	20512.6	740.8	181.6	7.5	18.7	66.66%	2.67%	47.23%
	sgl_S2.25m	65208.9	3178.2	4008.2	20288.0	830.0	112.0	13.6	22.2	66.53%	6.92%	70.72%
	sgl_S2.00m	65990.2	3232.1	4636.7	20494.6	1404.6	140.7	14.9	22.8	65.92%	7.57%	58.69%
	sgl_S1.75m	64675.2	2554.2	3505.9	19632.6	951.6	183.5	15.9	24.6	66.61%	5.91%	67.52%
	sgl_S1.50m	67731.5	1319.9	3234.2	21726.3	1914.3	267.5	10.1	33.9	63.89%	4.99%	41.09%
	sgl_S1.25m	75103.0	1104.7	2266.7	29850.1	1162.0	432.1	10.9	34.1	59.40%	3.24%	47.22%
4	Mix_ S	65141.9	1302.0	1994.3	20358.6	692.3	182.0	13.0	25.5	66.94%	3.52%	59.10%
	sgl_S2.25m	64629.9	4492.2	5160.2	19870.9	668.0	111.2	21.5	31.8	67.32%	8.62%	79.85%
	sgl_S2.00m	64846.1	4670.7	5453.3	19972.6	782.6	139.1	24.1	32.2	67.30%	8.95%	77.04%
	sgl_S1.75m	64172.4	3611.7	4374.9	19318.3	763.2	183.2	24.3	35.4	66.95%	7.43%	73.88%
	sgl_S1.50m	67586.6	1864.1	4133.3	21226.4	2269.2	267.0	16.1	50.6	64.12%	6.60%	46.23%
	sgl_S1.25m	75917.5	1654.1	3560.6	29920.1	1906.5	434.9	17.2	58.7	59.11%	4.97%	48.93%
8	Mix_ S	64800.5	1780.0	2487.6	20002.0	707.6	182.1	17.3	33.3	67.27%	4.43%	66.32%
	sgl_S2.25m	64282.6	4859.9	5435.6	19616.1	575.7	110.8	26.9	39.3	67.79%	9.23%	81.88%
	sgl_S2.00m	64334.4	5268.0	5851.7	19659.9	583.6	138.5	28.0	37.0	67.77%	9.48%	78.53%
	sgl_S1.75m	64000.6	3951.8	4652.0	19209.5	700.3	183.2	28.1	42.3	67.15%	8.31%	75.61%
	sgl_S1.50m	67338.4	2317.1	4634.3	20930.3	2317.2	267.2	21.4	67.0	64.33%	7.71%	53.97%
	sgl_S1.25m	76397.5	1744.4	4183.0	29867.9	2438.6	436.2	19.1	86.5	59.07%	6.35%	50.37%
16	Mix_ S	64659.0	1897.2	2718.6	19746.7	821.4	183.3	19.6	40.0	67.51%	5.09%	68.51%
	sgl_S2.25m	64119.1	4914.4	5416.9	19525.7	502.5	109.9	29.0	41.8	68.22%	9.28%	86.15%
	sgl_S2.00m	64344.2	5242.8	5974.7	19521.4	731.9	137.7	30.4	39.6	67.96%	9.74%	82.85%
	sgl_S1.75m	63929.2	4026.0	4704.8	19159.5	678.8	183.6	28.7	45.8	67.30%	8.47%	77.31%
	sgl_S1.50m	67487.2	2454.1	5056.1	20794.2	2602.0	267.6	23.1	70.5	64.47%	7.91%	57.87%
	sgl_S1.25m	76952.9	1900.6	5974.6	28788.0	4074.0	437.7	22.0	143.0	59.11%	8.86%	51.35%
32	Mix_ S	64581.6	2056.2	2865.5	19681.4	809.3	182.0	21.2	43.3	67.52%	5.46%	69.71%
	sgl_S2.25m	64082.2	4898.7	5354.2	19535.8	455.5	109.0	31.2	43.3	68.27%	9.21%	88.23%
	sgl_S2.00m	64225.5	4576.5	5429.5	19281.7	852.9	137.3	30.1	38.3	67.98%	9.24%	78.17%
	sgl_S1.75m	63892.4	3937.7	4681.6	19057.5	743.9	183.6	29.3	48.7	67.40%	8.61%	77.26%
	sgl_S1.50m	67510.0	2593.1	5655.9	20356.2	3062.8	267.6	23.4	82.3	64.47%	8.64%	58.50%
	sgl_S1.25m	77438.8	2374.4	7982.6	27739.6	5608.2	439.1	27.2	185.7	59.16%	10.69%	54.23%

by the additional setups required. Hence analysing the nonUL-Mix policy and nonUL-Single policy with large bin sizes would be beneficial in terms of material and setup cost. On the other hand, nonUL-Single with small bin sizes neither provide better material utilisation nor use less number of bins, hence smaller bin sizes demonstrate no advantage in terms of saving material or reducing the number of setups.

Implementation of UL policies

Compared to the nonUL policies, the implementation of UL policies has resulted in an improvement in material utilisation for both time horizons as illustrated in Figure 6.3. The results reveal a steeper improvement by each of UL-Single policies than the UL-Mix policy. Increasing J_{max} has improved the overall material utilisation strongly at lower values of it. This is clearly demonstrated by the rapid improvement from $J_{max} = 0$ to 8. However, increasing J_{max} at higher values do not demonstrate significant advantages (see $J_{max} = 8$ to 32) since there is a sufficient level of heterogeneous bin sizes to reach a good solution with higher utilisation.

According to Figure 6.3, it is also noticeable that the substantial improvement of material utilisation in the UL-Single policy with the largest bin size surpasses the utilisation recorded by Mix policy at the higher values of J_{max}. The Mix policies record the best utilisation only when J_{max} (< 4). Therefore, it is reasonable to say that using a UL-Single policy with large bin sizes leads to the least usage of material and the minimum trim loss.

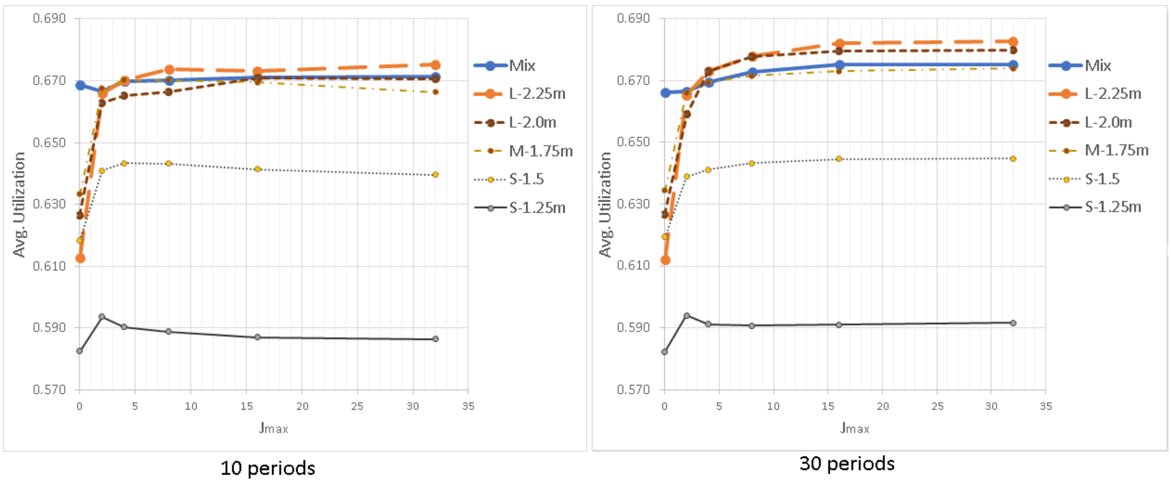


Figure 6.3: Model 01: Use of leftovers with different J_{max} values

With respect to the number of bins occupied, the Tables 6.1 and 6.2 reveal that usable leftovers result in an increase in the total number of bins used ($\sum N_t^{(S)} + \sum N_t^{(O)}$) than in non_UL. In addition to the increase of setups, this causes an extra effort related to managing the leftovers. Comparatively, for every J_{max}, both UL Single policy with large bin sizes (sgl_S2.25m and sgl_S2.00m) and UL Mix policies provide the minimum number of ULs return to the warehouse. Similar to nonUL, the UL Single policy with the largest bin size occupied the minimum number of bins.

Increasing J_{max} allows the reuse of different types of bins at each period. Therefore, we further investigate how the increase in leftovers affect material saving. The ratio $\sum Area_t^{(R)} / \sum Area_t^{(S)}$ is used in measuring the degree of ULs return to the warehouse and the ratio $\sum Area_t^{(O)} / \sum Area_t^{(R)}$ is used in measuring the degree of use of those returned ULs. The results of these measurements are tabulated in the last two columns of Table 6.1 and 6.2.

Not surprisingly, the higher the J_{max}, the higher the ratios for each policy in the results. The UL-Single policy with the largest bin size recorded the highest $\sum Area_t^{(O)} / \sum Area_t^{(R)}$ ratio at higher J_{max} values. Also, expanding the time horizon encourages waste reduction (i.e. Improved utilisation for 30 periods time horizon than 10 periods). The graphs in Figure 6.4 illustrate how the ratio $\sum Area_t^{(O)} / \sum Area_t^{(R)}$, $N^{(R)}$, $N^{(O)}$ of the Mix, largest (2.25m) and smallest (1.25m) standard bin options vary at each J_{max} value. The set of graphs in column one represent the results related to 10 periods whereas the other column represents the same for 30 periods. The results demonstrate a higher ratio of $\sum Area_t^{(O)} / \sum Area_t^{(R)}$ for 30 periods and this provide evidence for increasing opportunities of reusing residuals at longer time horizons. In the case of UL-Single largest (2.25m), it has demonstrated the highest number in utilising OldUL bins in the warehouse. Also, the results demonstrate the UL-Single smallest (1.25m) option records the

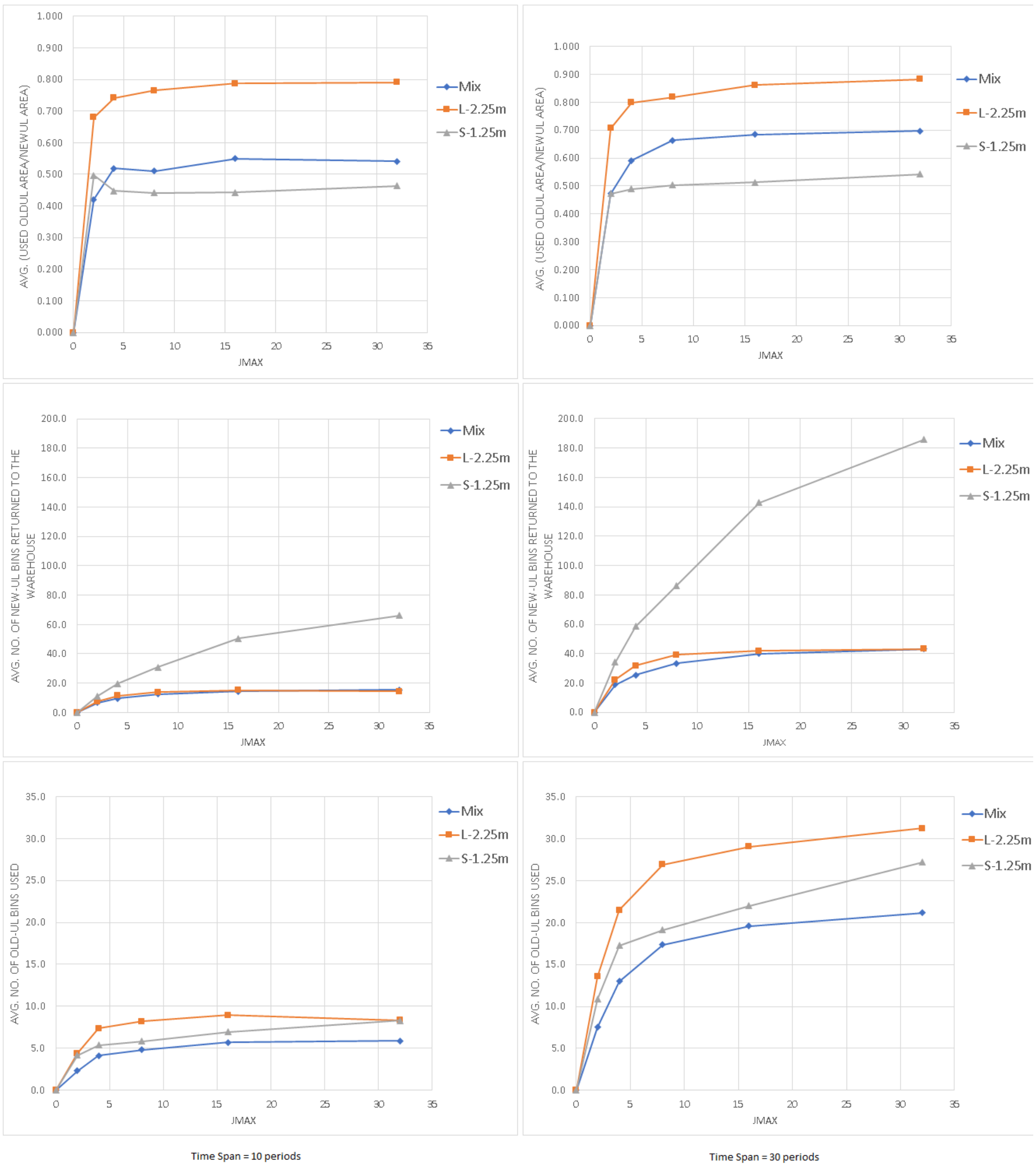


Figure 6.4: Model 01: Use of leftovers with different J_{max} values

highest number of ULs return to the warehouse than the UL-Mix and any other UL-Single bin size options. However, the results demonstrate a lower performance by the UL-Single smallest (1.25m) option in terms of utilising those ULs.

6.4.2 Findings and discussion

The overall results of Model 01 summaries, that both nonUL-Mix and UL-Single with large standard bin size demonstrate significant improvements in terms of material utilisation than other options. While the Mix policy provides the best material saving when there is no use of ULs, the single standard bin policy with large bin sizes provides the best utilisation with ULs. Therefore, when the objective is to save material, the mix sizes policy is most favourable if the production policy is set to discourage using ULs. Otherwise, if the operational policy is based on using ULs and there is a substantial allowance for managing enough number of UL types, then proceeding with the largest standard bin size with ULs is most favourable in order to achieve high utilisation of material. In this case, it is important to set an adequate upper bound on the number of UL bin types and proceeds the multi-period run for the substantial amount of periods so that using ULs can be reached to a steady state level which effectively utilises the returned ULs in the subsequent periods.

While analysing the number of bins used in each case, when the production is not supported by the use of ULs, the single large standard bin size (nonUL-Single Large) records the minimum number of used bins than any other option even in the context of saving material as much as possible. Similarly, when the production is supported by the use of ULs, the single large standard bin size (UL-Single Large) records the minimum number of used bins than other options.

Analysis related to the number of used bins is important in terms of cost of setups which has a significant effect on the economic production as the cost material is not always being the dominant factor in the total cost of sheet cutting production. Therefore, we propose our next model to further analyse the cost trade-off of material and setups arise when implementing those most influential policies; nonUL-Mix, UL-Mix, nonUL-Single with standard large bins and UL-Single with large bins. We also consider the labour commitment to managing ULs since it adds a significant contribution to the total cost in an expensive labour scenario.

6.5 Model 02: Minimizing the Cost

The findings in Section 6.4, motivate us to extend the analysis of using operational policies considering minimising the total cost if manufacturers focus is on the economic benefit than just saving material exclusively. In this study, we further investigate how nonUL-Mix, UL-Mix, nonUL-Single and UL-Single policies perform to minimise the total cost of sheet cutting.

In our investigation, we tried to identify several practical scenarios of the sheet cutting process. The cost trade-offs have a significant impact on the economic value on deciding which procurement or production policy to implement. Yet, this has been given less attention in the literature and we conduct an empirical analysis of the suitability of each operational policy using Model 2. As per practical situations, we identify that there are three factors which have the biggest impact on sheet cutting; material, machinery and labour. These factors contribute to four key cost factors; unit cost of material (c), setup cost per sheet (s), holding cost per off-cut per period (h) and transportation and handling cost per off-cut (r). The first cost component is related to material, the second is related to machinery and the last two are related to labour. According to the cost structure which determines the total cost of sheet cutting, a certain situation of sheet cutting process can be dominated by either material (i.e. material dominant), setups (i.e. setup dominant), labours (i.e. labour dominant) or balanced. Based on these situations, we outline nine operational scenarios considering the combinations of each cost factor which are represented in Table 6.3 and Figure 6.5. Please note that in Figure 6.5 the scenarios are numbered in an order which supports to identify a pattern in the performance characteristics of each operational policy.

Table 6.3: Characteristics of scenarios

Scenario	Cost structure	
Scenario 1	c >>> s	At high level of h and r
Scenario 2	c <<< s	At high level of h and r
Scenario 3	A balanced level of c & s	At high level of h and r
Scenario 4	c >>> s	At medium level of h and r
Scenario 5	c <<< s	At medium level of h and r
Scenario 6	A balanced level of c & s	At medium level of h and r
Scenario 7	c >>> s	At low level of h and r
Scenario 8	c <<< s	At low level of h and r
Scenario 9	A balanced level of c & s	At low level of h and r

Each of these scenarios is tested for the four policies in order to determine the best policy for each scenario.

Cost function:

The total cost function includes four cost parameters: i) Material cost of occupied bins, ii) Setup cost proportional to the number of bins occupied with the solution, iii) Holding cost of *Old-UL* bins and iv) Warehouse return cost of *New-UL*s.

$$Cost_t = c(Area_t^{(S)} + Area_t^{(O)} - Area_t^{(R)}) + s(N_t^{(S)} + N_t^{(O)}) + hN_t^{(Ow)} + rN_t^{(R)} \quad (6.2)$$

c : material cost per unit area

s : set up cost per bin

h : holding cost per *Old-UL* bin

r : inventory return cost per *New-UL* bin

Instead of setting an upper bound as for *Model 01*, the generation of UL types is controlled through the return and holding costs assigned to the NewULs in the objective function. As an example, a significant contribution of return cost can influence to reduce *New UL* bins and therefore discourage ULs to store in the warehouse. At the

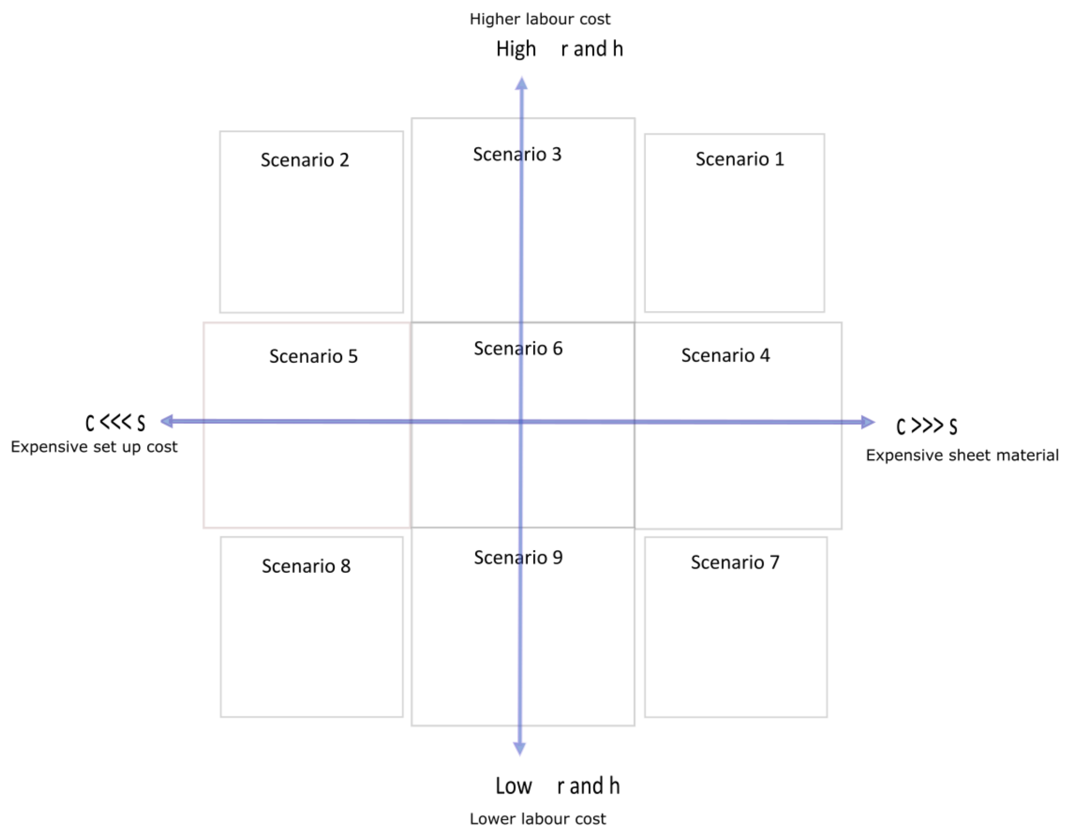


Figure 6.5: Cost Scenarios

comparison stage, we take $c \times \sum Area_t^{(S)}$ for all t values to find the total material cost incurred for the planned time horizon. In addition, $\sum s(N_t^{(S)} + N_t^{(O)})$, $\sum hN_t^{(Ow)}$ and $\sum rN_t^{(R)}$ denote total setups cost, total holding cost and total warehouse return cost respectively. Finally, the total cost within the planned time horizon is calculated by $\sum Cost_t^{(S)} + Cost\ of\ Old\ ULs\ remaining\ at\ the\ end\ of\ time\ horizon.$

6.5.1 Computational experiments

The experiment settings:

Similar to Model 01, we designed a set of experiments to execute the effect of each policy using the instances mentioned in Section 6.4.1. We selected two time horizons; 10 and 30 consecutive periods where at each period, a random demand quantity within the range of [10-50] was taken as the number of pieces to be cut. Since we investigated 4 operational policies in 9 different scenarios considering multi-period runs, the computation effort on this is significant.

Next, we set cost parameter values to analyse the variability of each paradigm not only for the usual practical cases but also for the extreme cases, by aiming at analysing how performances are varied from one extreme scenario to other, for each policy. As the cost parameters, we use different values for $h = \{1, 10, 100, 1000, 10000\}$ and $r = \{2, 20, 200, 2000, 20000\}$ where $r = 2h$ and $\alpha = \{0.05, 0.1, 0.5, 1, 1.5, 5, 10\}$ where $c = \alpha s$. The selected parameter values supported to conduct an empirical analysis by changing the relative contribution of cost components in the total cost function which was set to be minimised at each period. The parameter α is set to change the relative contribution of material and setup costs where higher alpha values make the cost function to be dominated by the material cost and lower alpha values make the cost function to be dominated by the setup cost. Similarly, higher r and h values make the cost function to be dominated by the cost of returning and holding ULs, which act as a restriction to return and use of ULs.

At each period, an irregular bin packing problem was solved using the modified IJRAB algorithm which terminates after 300 Jostle cycles.

Tables 6.4, 6.5, 6.6, and 6.7 summarize the results for four policies. The same results for each scenario are graphically represented in Figure 6.6 and 6.7 for 10 periods and 30 periods.

Table 6.4: Model 02: NonUL-Mix vs. UL-Mix results for with different α and h values (simulation results for 10 periods)

h	α	nonUL-Mix				UL-Mix					
		$\sum Area^{(S)}$ Avg.	$\sum Trimloss$ Avg.	$Cost^{(Total)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(S)}$ Avg.	$\sum Trimloss$ Avg.	$Cost^{(Total)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(R)}/\sum Area^{(S)}$ Avg.	$\sum Area^{(O)}/\sum Area^{(R)}$ Avg.
1	0.05	22220.2	7399.3	221041.1	0.661	22141.6	7059.9	226967.5	0.662	1.76%	33.10%
	0.1	22149.8	7328.8	122129.8	0.663	22137.5	7078.1	122322.7	0.662	1.65%	34.66%
	0.5	22118.3	7297.3	41962.5	0.662	22135.5	7079.8	42674.0	0.662	1.68%	36.98%
	1	22127.2	7306.2	32033.0	0.662	22074.9	6992.2	32373.5	0.663	1.83%	35.10%
	1.5	22094.0	7273.0	28774.2	0.664	22037.9	6932.8	29038.1	0.664	1.97%	34.55%
	5	22000.6	7179.6	24112.2	0.667	21909.7	6699.9	24185.1	0.669	3.37%	47.26%
	10	21952.3	7131.3	23026.6	0.669	21913.3	6660.4	23131.7	0.668	3.76%	47.58%
10	0.05	22092.2	7271.2	217788.0	0.664	22066.7	7071.6	223082.1	0.664	1.36%	41.93%
	0.1	22097.7	7276.8	121186.9	0.664	22135.4	7093.2	123318.1	0.663	1.50%	33.22%
	0.5	22097.0	7276.0	41899.6	0.663	22064.5	7048.2	42448.7	0.664	1.37%	35.56%
	1	22122.8	7301.9	32050.8	0.664	22154.8	7127.7	32663.6	0.663	1.43%	34.91%
	1.5	22088.9	7268.0	28860.8	0.664	22023.2	7034.6	29073.9	0.666	1.50%	49.13%
	5	21976.1	7155.1	24081.3	0.668	21849.1	6803.8	24277.6	0.671	2.12%	51.47%
	10	21941.5	7120.5	23031.6	0.669	21790.7	6741.6	23201.7	0.674	2.32%	54.87%
100	0.05	22108.3	7287.3	219271.6	0.663	22278.7	7402.8	224856.1	0.657	0.35%	29.11%
	0.1	22113.3	7292.3	120600.1	0.664	22170.6	7270.6	122409.0	0.660	0.49%	27.73%
	0.5	22142.1	7321.1	41729.2	0.662	22188.7	7331.8	42315.5	0.659	0.31%	48.45%
	1	22057.5	7236.5	31994.8	0.665	22134.6	7277.5	32244.2	0.660	0.29%	44.51%
	1.5	22059.4	7238.4	28732.1	0.664	22146.1	7292.4	29154.6	0.660	0.30%	50.11%
	5	22025.2	7204.2	24115.7	0.667	22019.3	7187.0	24267.4	0.664	0.17%	69.18%
	10	21924.9	7103.9	22991.0	0.669	21954.5	7115.5	23144.4	0.666	0.14%	42.54%
1000	0.05	22146.4	7325.4	218608.5	0.662	22258.1	7429.1	220208.9	0.657	0.05%	34.34%
	0.1	22163.1	7342.2	119783.6	0.662	22312.2	7484.6	122557.5	0.657	0.04%	21.47%
	0.5	22166.1	7345.1	42009.8	0.661	22271.3	7442.9	43420.5	0.657	0.04%	NA
	1	22083.2	7262.2	31891.8	0.664	22225.5	7401.5	33306.6	0.659	0.03%	NA
	1.5	22033.2	7212.3	28702.2	0.665	22160.6	7337.6	29865.8	0.660	0.03%	NA
	5	22012.2	7191.2	24112.5	0.668	22080.7	7258.4	24477.5	0.664	0.01%	NA
	10	21974.3	7153.3	23053.5	0.668	22081.3	7258.9	23535.4	0.664	0.01%	NA
10000	0.05	22126.5	7305.5	220029.4	0.662	22249.6	7417.7	229760.8	0.659	0.06%	12.01%
	0.1	22151.1	7330.1	120906.5	0.663	22293.7	7470.7	129015.1	0.656	0.03%	NA
	0.5	22131.9	7311.0	41901.7	0.662	22282.2	7455.9	51153.5	0.658	0.06%	59.39%
	1	22070.1	7249.1	32107.5	0.664	22251.9	7424.5	41778.5	0.658	0.03%	NA
	1.5	22049.4	7228.4	28773.2	0.665	22197.7	7370.2	34873.3	0.660	0.03%	NA
	5	21972.0	7151.0	24093.0	0.668	22110.9	7289.5	27415.2	0.663	0.01%	NA
	10	21954.8	7133.8	23021.7	0.669	22062.7	7240.3	28593.1	0.664	0.02%	NA

Best performing policy at each scenario

Comparison between Scenario 1 (Expensive material with higher labour cost of managing ULs) and Scenario 7 (Expensive material with lower labour cost of managing ULs):

Table 6.5: Model 02: NonUL-Single L vs. UL-Single L results for with different α and h values (simulation results for 10 periods)

h	α	nonUL-Single Large				UL-Single Large					
		$\sum Area^{(S)}$ Avg.	$\sum Trimloss$ Avg.	$Cost^{(Total)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(S)}$ Avg.	$\sum Trimloss$ Avg.	$Cost^{(Total)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(R)}/\sum Area^{(S)}$ Avg.	$\sum Area^{(O)}/\sum Area^{(R)}$ Avg.
1	0.05	23791.1	8970.1	170649.4	0.614	22429.1	6944.6	171799.9	0.654	6.22%	52.42%
	0.1	23848.0	9027.0	97452.7	0.614	22381.8	6926.6	96730.9	0.654	6.15%	53.90%
	0.5	23846.8	9025.8	38567.1	0.613	22333.7	6896.2	37198.3	0.655	5.87%	52.95%
	1	23811.3	8990.3	31160.5	0.614	22374.9	6911.7	29852.3	0.655	6.22%	53.86%
	1.5	23785.6	8964.6	28679.7	0.614	22293.9	6845.3	27311.5	0.658	6.37%	55.79%
	5	23828.9	9007.9	25299.8	0.613	22049.5	6660.1	23672.3	0.665	8.60%	70.03%
	10	23934.9	9113.9	24673.6	0.612	21938.0	6567.9	22760.1	0.669	9.54%	73.75%
10	0.05	23792.3	8971.4	170658.6	0.615	22344.5	6891.2	170871.5	0.656	5.94%	52.33%
	0.1	23708.7	8887.7	96883.7	0.616	22393.3	6899.9	97144.5	0.655	6.27%	52.08%
	0.5	23879.9	9059.0	38620.7	0.612	22336.6	6928.0	37499.9	0.657	5.65%	53.46%
	1	23841.5	9020.5	31200.0	0.613	22400.4	6948.7	30256.8	0.655	6.10%	53.86%
	1.5	23868.6	9047.6	28779.8	0.610	22240.7	6861.2	27589.4	0.659	6.16%	59.21%
	5	23896.5	9075.5	25371.6	0.612	22100.0	6770.7	24026.2	0.664	7.47%	69.19%
	10	23838.8	9017.8	24574.5	0.614	21958.8	6659.6	23142.9	0.668	8.74%	75.08%
100	0.05	23827.1	9006.1	170907.9	0.613	22305.3	7183.4	171564.1	0.653	5.17%	73.89%
	0.1	24040.2	9219.2	98238.3	0.610	22400.3	7240.8	97532.7	0.651	4.94%	69.43%
	0.5	23787.3	8966.3	38470.8	0.614	22355.1	7216.2	38579.9	0.652	4.97%	71.36%
	1	23792.3	8971.4	31135.6	0.615	22380.6	7237.7	31042.6	0.652	5.02%	71.36%
	1.5	23971.6	9150.6	28904.0	0.611	22326.0	7190.1	28657.4	0.654	5.16%	72.64%
	5	23818.7	8997.7	25288.9	0.614	22215.3	7174.6	24777.9	0.656	5.42%	81.75%
	10	23847.7	9026.7	24583.7	0.614	22271.8	7209.8	24120.5	0.654	5.35%	79.76%
1000	0.05	23858.2	9037.2	171131.1	0.613	23675.3	8845.3	173038.2	0.620	1.05%	96.38%
	0.1	23859.0	9038.0	97497.7	0.613	23670.9	8841.7	98667.6	0.620	0.92%	96.20%
	0.5	23769.1	8948.1	38441.4	0.615	23632.6	8803.0	39237.1	0.619	0.75%	95.12%
	1	23872.9	9051.9	31241.0	0.613	23641.3	8802.3	32139.4	0.621	0.92%	91.78%
	1.5	23830.6	9009.6	28734.0	0.613	23543.9	8706.5	29636.9	0.622	1.01%	93.04%
	5	23829.5	9008.5	25300.4	0.614	23632.6	8800.9	25917.4	0.619	0.71%	93.63%
	10	23860.2	9039.2	24596.6	0.613	23490.3	8661.7	25304.1	0.623	0.94%	96.50%
10000	0.05	23802.5	8981.5	170731.2	0.614	23537.1	8702.9	184809.2	0.622	1.06%	94.67%
	0.1	23840.6	9019.6	97422.7	0.614	23637.4	8816.4	110684.9	0.620	1.00%	100.00%
	0.5	23921.4	9100.4	38687.7	0.612	23538.0	8708.9	50401.9	0.622	1.09%	96.84%
	1	23908.0	9087.1	31287.0	0.612	23594.8	8765.5	40277.9	0.621	0.74%	95.19%
	1.5	23871.3	9050.3	28783.1	0.613	23537.6	8708.0	40309.9	0.622	0.95%	96.14%
	5	23823.6	9002.6	25294.2	0.613	23663.8	8842.8	35407.5	0.620	0.84%	100.00%
	10	23893.2	9072.3	24630.7	0.612	23558.1	8728.2	35482.1	0.622	0.90%	95.80%

Table 6.6: Model 02: NonUL-Mix vs. UL-Mix results for with different α and h values (simulation results for 30 periods)

h	α	nonUL-Mix				UL-Mix					
		$\sum Area^{(S)}$ Avg.	$\sum Trimloss$ Avg.	$Cost^{(Total)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(S)}$ Avg.	$\sum Trimloss$ Avg.	$Cost^{(Total)}$ Avg.	$Utilization\%$ Avg.	$\sum Area^{(R)}/\sum Area^{(S)}$ Avg.	$\sum Area^{(O)}/\sum Area^{(R)}$ Avg.
1	0.05	65911.5	21820.6	653298.2	0.662	65782.8	21050.6	684220.2	0.663	1.79%	45.59%
	0.1	65740.5	21649.6	358053.8	0.664	65679.8	21064.2	373612.5	0.665	1.62%	50.76%
	0.5	65766.3	21675.4	124576.1	0.663	65728.5	21054.7	128274.2	0.664	1.81%	51.01%
	1	65702.1	21611.2	95437.4	0.664	65624.5	20928.8	96871.7	0.664	1.82%	49.26%
	1.5	65654.2	21563.3	85716.5	0.665	65299.9	20650.3	86682.2	0.667	1.88%	54.53%
	5	65435.0	21344.1	71740.9	0.668	64895.2	20032.2	71958.7	0.672	2.98%	60.04%
	10	65375.7	21284.8	68575.9	0.669	64727.7	19844.1	68571.5	0.673	3.68%	66.74%
10	0.05	65853.5	21762.6	654636.0	0.662	65652.6	21047.1	689980.7	0.664	1.59%	50.79%
	0.1	65937.4	21846.5	357939.5	0.662	65590.7	21069.0	374576.8	0.665	1.41%	53.57%
	0.5	65832.5	21741.6	124116.0	0.663	65543.2	21012.2	128554.1	0.665	1.50%	55.18%
	1	65740.7	21649.8	95241.3	0.663	65513.9	20997.1	98234.8	0.666	1.42%	54.25%
	1.5	65610.6	21519.6	85494.5	0.664	65306.8	20800.0	87915.9	0.668	1.55%	59.02%
	5	65351.5	21260.6	71613.1	0.668	64847.9	20230.4	73318.5	0.673	2.25%	63.98%
	10	65321.8	21230.9	68521.6	0.668	64783.7	20217.7	70044.1	0.674	2.36%	68.91%
100	0.05	65958.2	21867.2	648241.5	0.662	66037.3	21840.5	667451.9	0.660	0.43%	62.63%
	0.1	65869.2	21778.3	358044.7	0.662	65973.1	21801.8	366609.2	0.660	0.35%	65.35%
	0.5	65773.0	21682.0	124026.6	0.663	66040.7	21849.0	127864.6	0.659	0.41%	62.73%
	1	65781.4	21690.5	95399.9	0.663	65874.0	21719.3	97597.5	0.660	0.31%	68.37%
	1.5	65687.1	21596.2	85664.5	0.665	65706.0	21556.2	87060.2	0.662	0.29%	68.65%
	5	65349.7	21258.8	71617.2	0.668	65425.9	21301.1	72391.4	0.665	0.22%	76.17%
	10	65281.2	21190.3	68484.1	0.669	65279.6	21163.3	69177.6	0.666	0.25%	84.67%
1000	0.05	65953.8	21862.9	654711.7	0.662	66182.9	22076.1	663699.5	0.658	0.08%	69.73%
	0.1	65838.3	21747.4	359385.8	0.663	66286.7	22171.9	370571.1	0.657	0.07%	49.07%
	0.5	65884.5	21793.6	124604.6	0.662	66294.6	22187.2	131664.6	0.657	0.05%	44.92%
	1	65716.2	21625.3	95324.8	0.664	66134.8	22034.6	101748.1	0.658	0.04%	67.24%
	1.5	65580.2	21489.2	85476.2	0.665	65930.4	21833.5	90779.7	0.661	0.03%	NA
	5	65335.1	21244.2	71551.0	0.668	65670.2	21575.7	75948.8	0.663	0.02%	NA
	10	65247.7	21156.7	68450.6	0.669	65665.0	21571.1	71076.4	0.664	0.02%	NA
10000	0.05	65916.3	21825.4	652014.7	0.662	66227.1	22120.8	727967.8	0.658	0.04%	43.52%
	0.1	65930.6	21839.7	357314.7	0.662	66344.8	22246.6	408833.8	0.657	0.03%	NA
	0.5	65765.9	21675.0	124235.9	0.663	66290.2	22185.0	185529.1	0.657	0.03%	NA
	1	65833.3	21742.4	95278.2	0.663	66209.7	22112.8	171594.1	0.658	0.04%	77.91%
	1.5	65623.7	21532.8	85540.5	0.665	66053.3	21956.9	124502.1	0.660	0.02%	NA
	5	65414.6	21323.7	71727.2	0.668	65785.9	21693.0	102210.1	0.662	0.02%	NA
	10	65368.4	21277.4	68583.1	0.668	65693.5	21600.8	101311.5	0.664	0.01%	NA

In Scenario 1, the nonUL-Mix has demonstrated the lowest cost as illustrated in the top right graphs of Figures 6.6 and 6.7. In this scenario selecting a nonUL option is cost beneficial than UL options since there is a high cost associated with managing leftovers

Table 6.7: Model 02: NonUL-Single L vs. UL-Single L results for with different α and h values (simulation results for 30 periods)

h	α	nonUL-Single Large				UL-Single Large					
		$\sum Area^{(S)}$	$\sum Trimloss$	$Cost^{(Total)}$	$Utilization\%$	$\sum Area^{(S)}$	$\sum Trimloss$	$Cost^{(Total)}$	$Utilization\%$	$\sum Area^{(R)}/\sum Area^{(S)}$	$\sum Area^{(O)}/\sum Area^{(R)}$
		Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.
	0.05	71288.4	27197.5	511340.5	0.612	66020.3	20699.1	510074.1	0.660	5.27%	64.66%
	0.1	71032.8	26941.9	290269.9	0.613	66044.3	20724.9	288580.1	0.660	5.24%	64.50%
	0.5	71225.6	27134.7	115192.0	0.612	65998.1	20716.5	110741.2	0.660	5.32%	66.08%
1	1	71136.6	27045.7	93092.4	0.612	65961.6	20738.4	88530.2	0.660	5.36%	67.99%
	1.5	71149.8	27058.9	85789.7	0.612	65654.5	20514.4	80848.6	0.664	5.59%	71.42%
	5	70988.6	26897.7	75370.6	0.613	64839.0	19897.9	69792.2	0.675	7.87%	83.33%
	10	71208.3	27117.4	73406.1	0.612	64725.4	19743.3	67352.4	0.676	8.79%	84.34%
	0.05	71218.3	27127.4	510837.5	0.612	66020.2	20739.9	511910.0	0.660	5.08%	64.57%
	0.1	71052.7	26961.8	290351.3	0.613	65984.5	20718.4	289590.1	0.661	5.04%	64.63%
	0.5	71208.5	27117.5	115164.3	0.612	65959.2	20836.3	112577.7	0.661	5.10%	69.34%
10	1	71182.0	27091.1	93151.7	0.612	65844.4	20683.7	90327.4	0.662	5.28%	69.21%
	1.5	71055.4	26964.5	85675.9	0.613	65674.9	20555.0	82746.6	0.664	5.54%	71.73%
	5	71191.5	27100.6	75586.0	0.612	64969.0	20126.3	71537.4	0.671	6.90%	83.22%
	10	71280.4	27189.5	73480.4	0.612	64669.4	19900.0	68698.9	0.675	7.77%	86.49%
	0.05	71174.5	27083.6	510523.2	0.612	66228.4	21601.5	513315.8	0.654	4.50%	82.00%
	0.1	71154.0	27063.1	290765.0	0.612	66337.2	21634.2	294034.8	0.653	4.50%	79.50%
	0.5	71119.1	27028.2	115019.7	0.612	66303.6	21678.2	116485.9	0.653	4.65%	82.68%
100	1	71055.9	26965.0	92986.7	0.613	66079.7	21531.7	93514.2	0.655	4.48%	84.56%
	1.5	71191.2	27100.3	85839.6	0.612	66000.7	21547.9	85462.3	0.655	4.60%	88.08%
	5	71054.1	26963.2	75440.1	0.613	65883.0	21473.3	74313.4	0.656	5.17%	90.64%
	10	71087.4	26996.4	73281.4	0.613	65801.5	21433.0	71684.7	0.656	5.10%	91.74%
	0.05	71091.9	27001.0	509931.1	0.613	70420.4	26326.3	513893.1	0.620	0.87%	99.49%
	0.1	71168.8	27077.9	290825.7	0.612	70399.7	26299.8	293564.0	0.620	0.92%	98.62%
	0.5	70870.7	26779.8	114618.0	0.614	70233.6	26131.5	117822.5	0.621	0.96%	98.34%
1000	1	71187.9	27097.0	93159.5	0.612	70400.6	26300.7	95962.2	0.620	0.95%	98.66%
	1.5	71112.2	27021.3	85744.3	0.613	70281.7	26165.0	88496.2	0.621	0.94%	96.11%
	5	71200.8	27109.8	75595.9	0.612	70342.8	26243.2	77960.8	0.620	0.86%	98.57%
	10	71110.6	27019.7	73305.4	0.613	70238.5	26127.5	75849.5	0.621	0.90%	96.83%
	0.05	71171.0	27080.1	510498.1	0.612	70441.1	26341.3	541109.1	0.619	0.84%	98.50%
	0.1	71303.1	27212.2	291374.3	0.611	70386.2	26285.4	324609.6	0.620	0.88%	98.40%
	0.5	71273.9	27182.9	115270.1	0.612	70429.7	26329.5	147876.7	0.620	0.91%	98.54%
10000	1	71094.2	27003.3	93036.9	0.613	70424.8	26322.6	122525.0	0.619	0.86%	98.12%
	1.5	71164.2	27073.2	85807.0	0.612	70456.8	26357.2	120329.6	0.620	0.91%	98.65%
	5	71313.0	27222.1	75715.0	0.612	70316.8	26216.8	108960.3	0.620	0.94%	98.63%
	10	71229.6	27138.7	73428.1	0.612	70472.6	26364.1	102128.9	0.619	0.83%	97.00%

(r). Therefore, at each period, the algorithm selects the solutions with less number of NewULs, in order to minimise the total cost. In this scenario, the choice of selecting single bin size vs. mix bin size policy leads us to choose the mix bin size policy due to its ability to save material through a variety of bin sizes and thereby reducing the total cost. As the scenario is significantly dominated by material cost compared to the setup cost, the impact of setup cost by using a higher number of bins with a mix of bin sizes is negligible. In summary, within the context of implementing nonUL, the results reveal that the mix of bin sizes is able to save expensive material better than any of the single bin size options when setup cost is not making significant effect against the cost saving achieved by the material.

We noticed a similar outcome in Model 1, where the mixed bin sizes provide the highest saving than any single bin size if the production policy is based on no use of ULs. In fact, the context of not using ULs in Model 1 (i.e. $J_{max} = 0$) emulates the features of Scenario 1 in Model 2, by discouraging the occurrence of ULs. Additionally, in Scenario 1 in Model 2, since materials are expensive, it also encourages saving material as much as possible as in Model 1.

The results related to the Scenario 7 illustrated by the bottom right graphs in Figures 6.6 and 6.7, shows the lowest total cost when the UL-Large policy is imposed. Scenario 7 motivates minimising material usage due to the dominance of the material cost in the cost function. The scenario also encourages more returns of ULs to the warehouse due to the low labour cost of managing ULs, while encouraging saving material as much

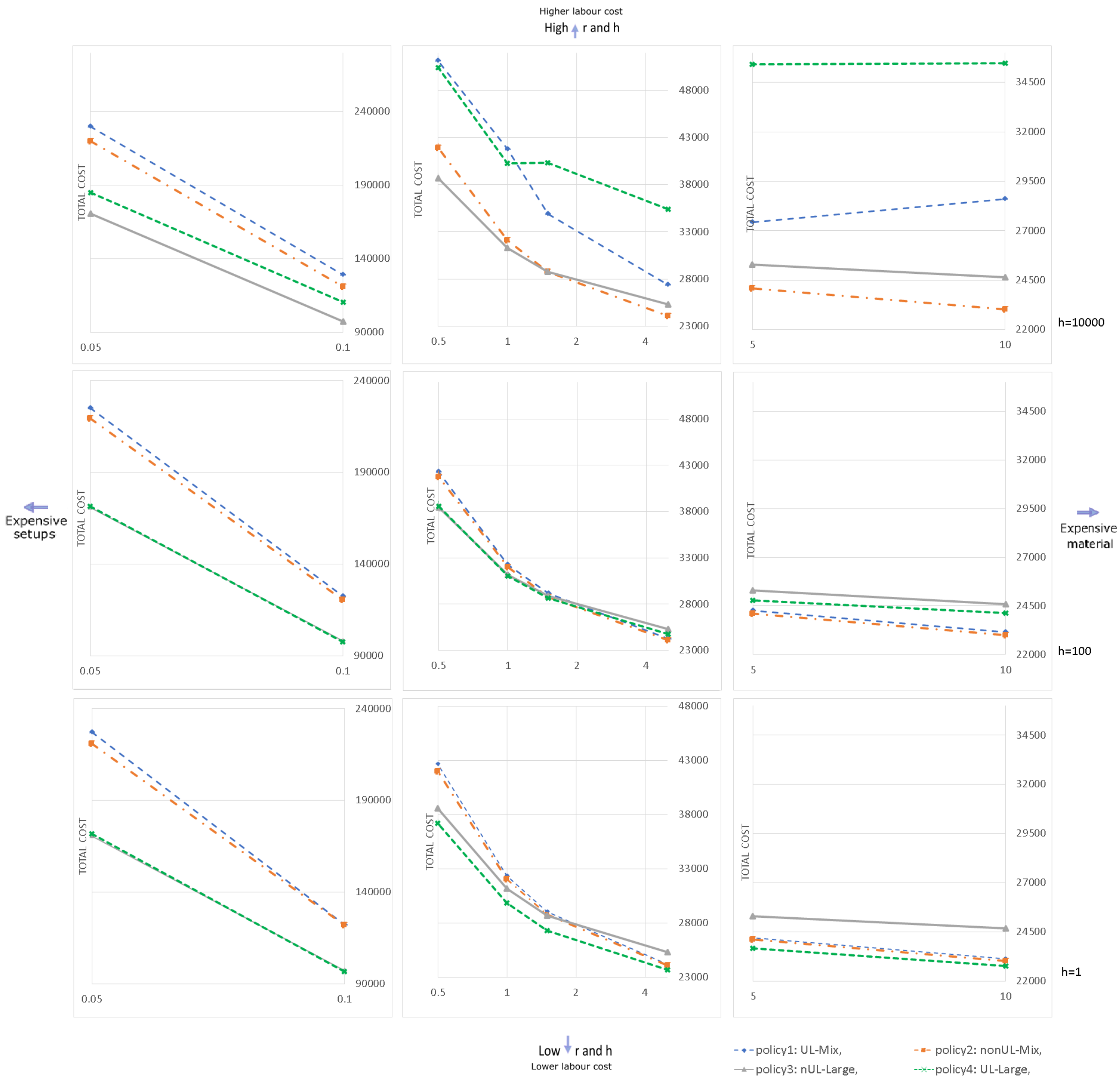


Figure 6.6: Model 02: Total cost graphs by running simulation for 10 periods for each policy

as possible. According to Tables 6.4, 6.5, 6.6, and 6.7, it is evident that the UL-Large policy results an increased number of leftovers when h and r are lower and α is higher. In order to reduce the total cost in Scenario 7, the results demonstrate more returns of ULs and using them, to save more material since the material cost dominates the total cost. We have observed similar outcome when assessing results of Model 1 in which the highest usage of ULs was recorded in UL single bin size policy with larger bin size than any other policy.

Comparison between Scenario 2 (Expensive setups with higher labour cost of managing ULs) and Scenario 8 (Expensive setups with lower labour cost of managing ULs):

In Scenario 2 (see the graphs on the top left in Figures 6.6 and 6.7), the cost environment

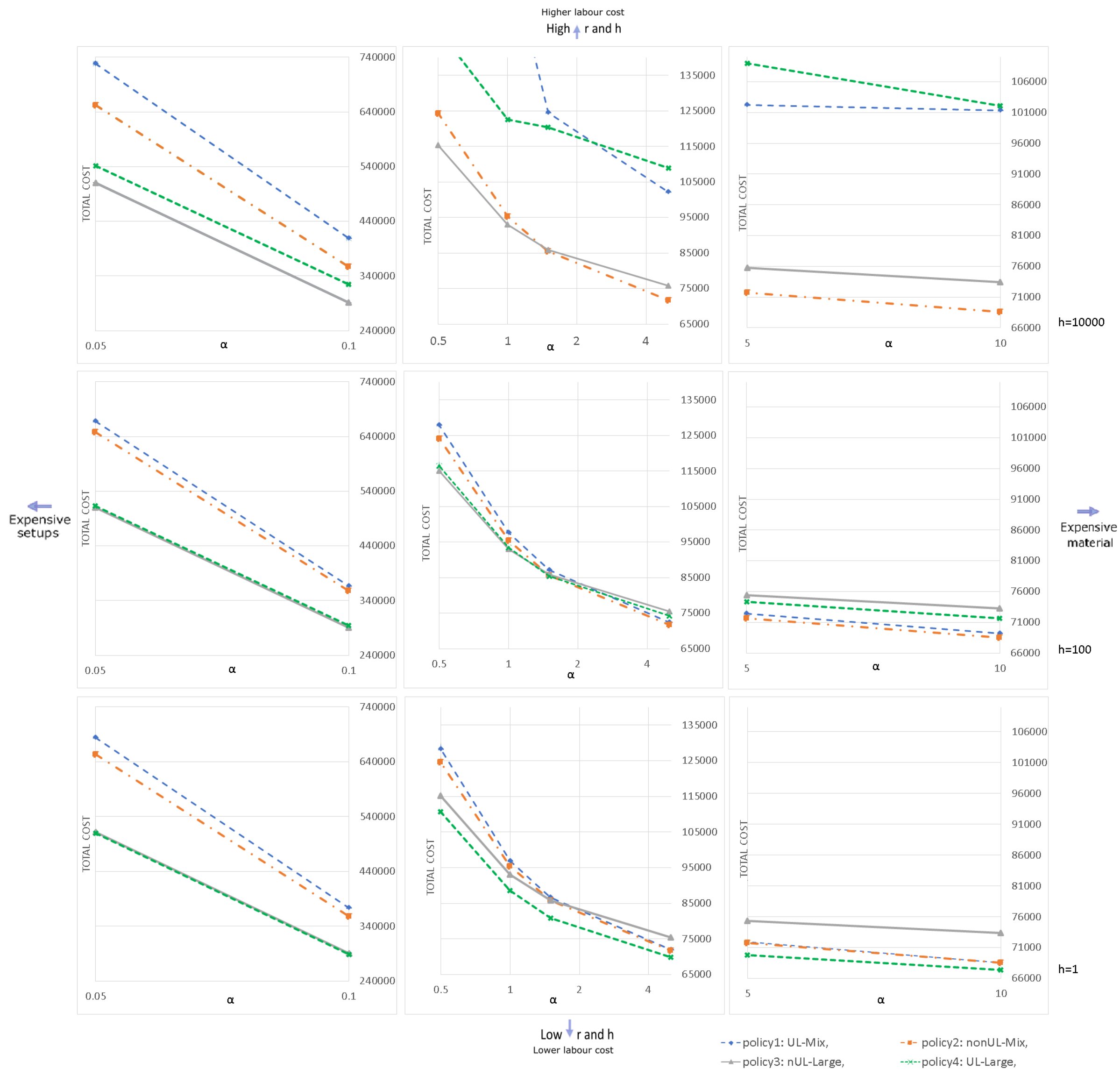


Figure 6.7: Model 02: Total cost graphs by running simulation for 30 periods for each policy

is dominated by setup cost and labour cost, therefore results demonstrate solutions with less number of bins by encouraging minimising the number of used bins and the returning of ULs to the warehouse. Therefore, the policies with single standard bins with smaller sizes and mix sizes of bins are not favoured in this case since those options result in a higher number of bins. Instead, using large bin sizes which use a lower number of setups are preferred in this scenario. This situation is also supported by the experiment results of Model 1 in the case of nonUL which records the lowest number of used bins with largest bin sizes.

However, the scenario simultaneously motivates using ULs due to the low cost of labour in managing ULs. The simulation results for Scenario 8 show almost equal minimum costs for nonUL-Single Large and UL-Single Large policies than the other options (see bottom left graphs of Figures 6.6 and 6.7). As in Scenario 2, Scenario 8 motivates solutions with a lower number of bins (i.e. setups). Set-up cost dominates the cost function and therefore supports the use of large bin sizes which reduces the number of used bins. Even though there is an allowance for ULs, the higher dominating power of setup cost in this scenario forces to use less number of bins from standard bins, UL bins or both. The UL-Single Large shows improvement in saving material than nonUL-Single Large though it doesn't demonstrate a significant advantage in the view of total cost since both policies tend to have no significant difference in amount of setups used. Also, in Scenario 8, achieving low-cost solutions with smaller standard bin sizes with or without using ULs is not likely due to the limitation in their sizes (i.e. smaller standard bins and smaller ULs making a higher number of used bins in the solutions which are not advantageous in terms of setup costs).

Scenario 4 (Expensive material with medium level of labour cost for managing ULs)

Scenario 4 is an intermediate situation between Scenario 1 and Scenario 7. Scenario 4 motivates minimising material usage as material cost dominates the total cost function, however, the impact of the labour cost of managing ULs can be varied from high to low. As a transitional scenario, it is worth discussing how the appropriate policies are determined by the variability in labour costs in the total cost function.

The three graphs on the right side of the Figures 6.6 and 6.7 demonstrate how the variability in labour costs affects the total cost when the material is expensive. When moving from Scenario 7 to 1 (from low to high of r and h), the best applicable policy is changed from UL-Large to nonUL-Mix. During this transition, the total cost of the UL-Large policy with respect to the other three policies is increased. On the other hand, the total cost of the nonUL-Mix policy with respect to the other three policies is decreased. This implies that the appropriate policy for Scenario 4 depends on the relative bias towards Scenario 7 or Scenario 1. However, the applicable policies in this scenario where the expensive material is being used can only be UL-Large or nonUL-Mix.

In summary, it is evident that both UL-Large policy and nonUL-Mix policies are more sensitive to variations in labour costs of managing ULs when expensive sheet materials are used. It also proved that when unit costs of h and r are moving from low to high, the best applicable policies are shifting from UL Large policy to nonUL-Mix policy.

Results of Scenario 5 (Expensive setups with medium level of labour cost for managing ULs)

Scenario 5 is the intermediate situation between Scenario 2 and Scenario 8. Scenario 5 motivates minimising number of used bins as setup cost dominates the total cost function, while the impact of the labour cost of managing ULs varies from high to low. As a transitional scenario, we discuss the appropriate policy alignment for the variability in labour cost in the total cost function.

The three graphs in the left side of the Figures 6.6 and 6.7 demonstrate how the variability in labour costs (h and r) affects to the total cost when the setup cost is high. When moving from Scenario 8 to 2 (from low to high of r and h), the best applicable policy is changed from UL-Large to nonUL-Large. During this transition, the total cost of the UL-Large policy with respect to the other three policies is increased. In contrast, the total cost of nonUL-Large policy with respect to the other three policies is decreased. This implies that the appropriate policy for Scenario 5 depends on the relative bias towards Scenario 8 or Scenario 2. However, the applicable policies in this scenario where expensive setups are being used can only be UL-Large or nonUL-Large.

In summary, this demonstrates that both UL-Large policy and nonUL-Large policies are more sensitive to variations in labour costs of managing ULs when expensive setups occur. It is also evident that when unit costs of h and r are moving from low to high, the best applicable policies are shifting from UL Large policy to nonUL-large policy.

Results of Scenario 3 (Balanced cost level of material and setup at high labour cost factors for managing ULs)

Scenario 3 is an intermediate situation between Scenario 2 and Scenario 1. Scenario 3 discourages using ULs due to the high cost of labour. On the other hand, the total cost is mostly determined by the variability of the relative ratio between setup cost and material cost.

The test results illustrated in top three graphs of the Figures 6.6 and 6.7 demonstrate how the variability in the ratio between unit setup and unit material cost affects the total cost in Scenarios 2, 3 and 1. When moving from Scenario 2 to 1 (from relatively lower material cost to higher material cost and from relatively higher setup cost to lower setup cost), the best applicable policy is changed from nonUL-Large to nonUL-Mix. During this transition, the total cost of nonUL-Large policy relative to nonUL-Mix policy is increased. In contrast, the total cost of nonUL-Mix policy relative to nonUL-Large policy is decreased. This implies that the appropriate policy for Scenario 3 depends on the relative bias towards Scenario 2 or Scenario 1. However, the applicable policies swing between nonUL-Large or nonUL-Mix.

Results of Scenario 9 (Balanced cost level of material and setups at low labour cost factors for managing ULs)

Scenario 9 is an intermediate situation between Scenario 8 and Scenario 7. Scenario 9 motivates using ULs to save material due to the low cost of labour, however, the total cost is determined by the variability of the relative ratio between setups cost and material cost.

The test results illustrated in bottom three Figures 6.6 and 6.7 demonstrate how the variations in c and s affect the total cost in Scenarios 8, 9 and 7. When moving from Scenario 8 to 7 (from relatively lower material cost to higher material cost ad from relatively higher setup cost from lower setup cost), surprisingly, the best applicable policy remains unchanged in UL-Large, due to the dominance of leftover generation motivated by the low cost of h and r. Therefore, it is interesting to analyse the change that would happen when leftover generation is limited by increasing h and r.

Results of Scenario 6 (Balanced cost level of material and setups at medium level of h and r)

Scenario 6 demonstrates a balanced state of all other eight scenarios. Along the horizontal level of Scenario Matrix illustrated in Figure 6.5, material cost and setup cost dominates the decision of selecting the appropriate policy (see the variation of three cost graphs from left to right; in the middle of the Figures 6.6 and 6.7). In other words, the results correspond to this scenario demonstrate minimum cost by using UL-Large or nonUL large policies when setup cost dominates the cost function. On the other hand, when the material cost dominates the cost function, the nonUL-Mix and UL-Mix provides the minimum cost.

Along the vertical level of the Matrix, variability in labour costs of managing ULs affects the total cost. In this case, Scenario 6 acts as a transitional stage between Scenario 3 and Scenario 9. When unit costs of h and r are moving from low to high (see the variation of three cost graphs from bottom to top; in the middle of the Figures 6.6 and 6.7, the best applicable policies shift from UL-Large policy to nonUL-Large policy when setup cost is dominated or if not, to nonUL-Mix policy when material cost is dominated.

6.5.2 Findings and discussion

Summarising the results of the direct cost model, we develop a supportive tool which companies can use as a guide in deciding the appropriate policy for their cost scenario. The graphical illustration of the framework is shown in Figure 6.8.

If a company follows one of the four extreme scenarios; 1,2,7 and 8, this framework provides direct guidance in choosing the appropriate policy. For example, the framework guides a company to use UL-Large policy if their practical situation is similar to Scenario 7. One of the key findings of the data analysis of Model 2 is that when the sheet cutting process acts on a transitional scenario, then there is an intersection of policies that can be used in that scenario. For example, in Scenario 5, there is a shift in policy which depends on the variability of the labour cost. Similarly, in Scenario 3, the policy shift can be identified based on the variability of the ratio between material cost and setup cost. This is clearly represented in the supportive tool by showing the intersections of policies in transitional scenarios.

By reviewing existing literature, we found two main practices that companies believe when dealing with residuals. First one is that they believe that material waste can

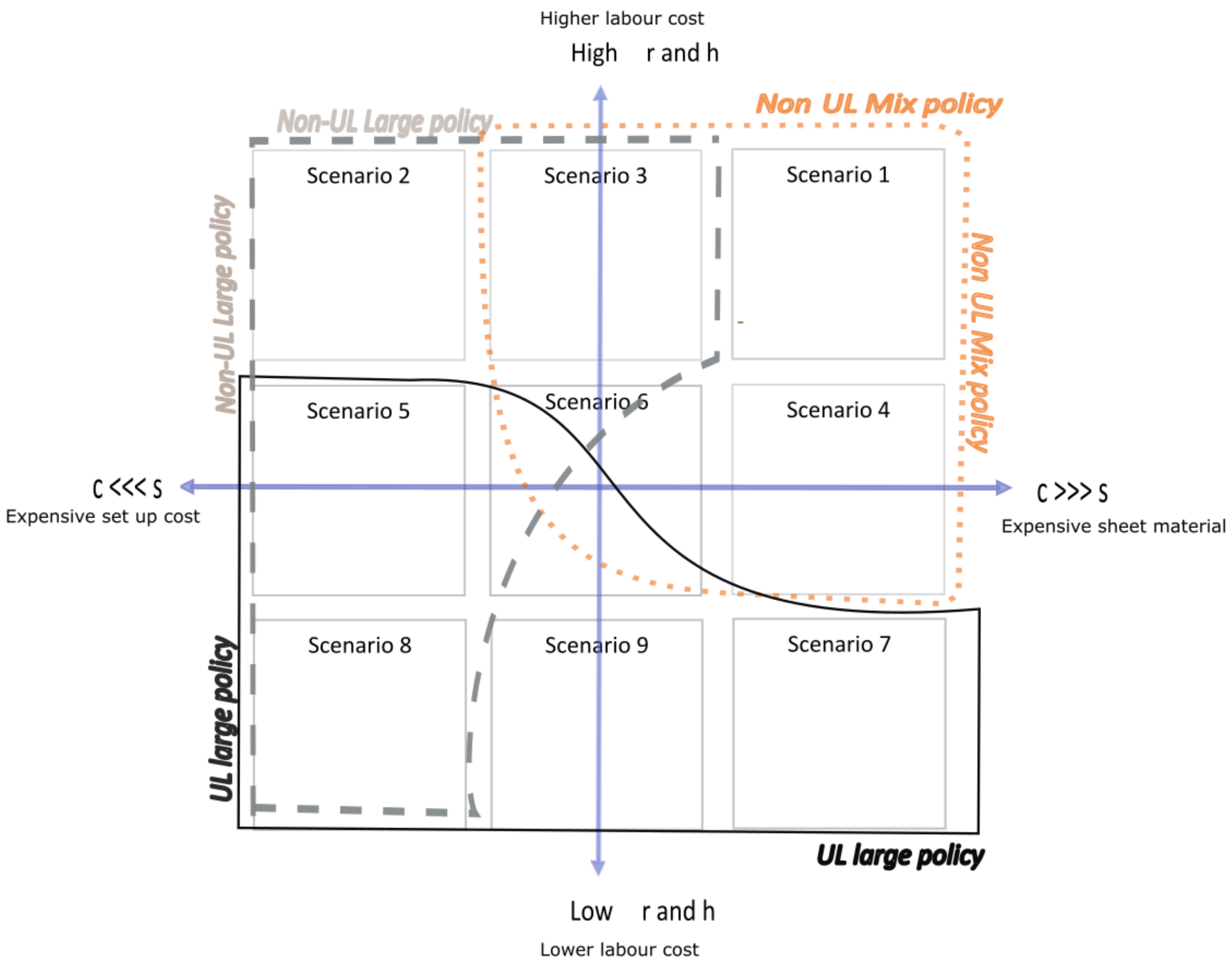


Figure 6.8: Supportive tool for policy selection

be minimised by reusing residuals. Cui et al. (2016) claim that multiple bar types in one-dimensional cutting are useful for improving material utilisation as they expand the solution space. Furthermore, they claim that different sizes of residuals can improve utilisation when there is a small number of standard bar types to purchase. The findings of this study support this argument, even for the 2D irregular sheet cutting process, by providing economic outputs in Scenarios 7,8 and 9. However, the other scenarios failed to provide economical solutions mainly due to the domination of other cost factors.

Manufacturers also prefer reducing the number of setups, due to the assumption of high setup cost in manufacturing compared to material cost. According to our analysis, we found that this is highly applicable in Scenarios 2,5 and 8, but is not applicable or Scenarios 1,4 and 7.

This sums up that practical situations of using residuals depend on different cost factors. Therefore, it is important to evaluate those different factors in deciding the appropriate operational policy, to achieve cost-effective sheet cutting process with residuals.

6.6 Concluding Remarks

For practical applications, it is believed that the use of a combination of sheet sizes can reduce the overall consumption of material. In order to achieve this, a manufacturer tends to follow one of four operational policies: 1) use a mix of standard sheet sizes with no use of residuals, 2) use a mix of standard sheet sizes with the use of residuals, 3) use a single size standard stock sheet with no use of residuals or 4) use single size standard stock sheets with the use of residuals. In this chapter, we investigate the outcome of those four operational policies. The study explicitly considers managing the maximum number of UL-types if a production policy is based on using residuals; by considering the handling and storage limitation of ULs arise in a production environment. In comparison, the policies based on ULs often lead to the significant reduction in the trim-loss and improve the overall utilisation of material. In this study, we demonstrate that the largest standard bin sizes perform well in saving material than other mentioned policies, by allowing an adequate upper bound on number of UL bin types for an adequate number of periods, so that multi-period run can be reached to a steady state level which effectively returns a substantial amount of ULs to the warehouse and effectively utilizes those ULs in subsequent periods. Otherwise, the study recommends the NonUL Mix policy to operate in order to achieve the highest advantage in saving material.

Next, we extend the multi-period analysis to consider cost of material as well as setup cost and cost of managing ULs related to the sheet cutting process. In this case, we produce results to provide useful insights for manufacturers to tune their production and procurement policies using the cost results generated. Our investigation is based on a series of multi-period runs, which apply the above-mentioned four policies on nine different cost scenarios which are illustrated in Table 6.3 and Figure 6.5. The newly proposed cost model fills the gap in the literature by simulating multi-period cut order processing with different operational policies, particularly related to the 2D irregular sheet cutting problems. The study addresses the issues in cost trade-off and provides economical recommendations applicable in different scenarios.

By analysing the results, we summarise the outcomes of each policy at different cost scenarios and present them as a supportive tool which is presented in Figure 6.8. This shows that using Standard Single Large bins with ULs (UL-Single Large) is economical in low labour cost scenarios of managing ULs. If the cost of managing ULs is expensive, our findings recommend to use NonUL-Mix policy if material cost is dominant than setup cost; or use Non-UL Single policy with large bin sizes if setup cost is dominant than the material cost in the total cost function.

The findings in Figure 6.8 support operations managers to identify the practical scenario of a given sheet cutting process by assessing the characteristics such as the price of sheet material, the cost of machinery and the cost of labour, and then to select the most suitable policy. Furthermore, the tool provides an in-depth analysis of the transitional scenarios which explains the variability of cost factors affect to the total cost. We believe that the overall methodology described in this chapter can be applied for other cutting and packing problems to analyse the same behavioural aspects and decide the appropriate policies for different practical situations.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The main objective of this study is to develop efficient computational methods to solve three types of irregular shape bin packing problem. Based on the findings of this study, the concluding remarks are summarised as follows.

First, in Chapter 4 we consider the two-dimensional irregular shape single bin size bin packing problem, without restricting the final orientation angle of pieces at the point of placement. The newly developed constructive method embedded within the Jostle heuristic, finds competitively local optimal solutions within reasonable computation time compared to the other computational methods in the literature. As a search method which is applicable for irregular bin packing problems, we demonstrate that Jostle has the ability to self-governing the neighbourhood structure that modifies the solution. The proposed angle tuning mechanism can find promising placement-orientation angles on a continuous scale. We also propose an iterated jostle procedure to explore the solution space more broadly rather than being stuck at one local optimum. We show that the Iterated Jostling procedure enhances the searchability and improves its efficiency in finding good solutions.

In comparison, the tests conducted based on different pre-ordering, such as sorting pieces by decreasing area, did not reach the solution quality obtained by the local search methods. In addition, the generated layouts were significantly better than a random search process. After implementing and testing several different placement techniques we realised that the Minimum length and Maximum utilisation placement rules are better than the traditional Bottom Left rule for irregular bin packing with unrestricted rotation. We also test the results generated for unrestricted rotation of pieces vs. restricted rotation of pieces. Not surprisingly, packing pieces with no rotation restriction outperforms the results generated through restricted rotation of pieces apart from the jigsaw instances. Overall, the proposed algorithms have generated better results within less computation time than the results published in the literature.

In Chapter 5, we consider the two-dimensional irregular shape multiple bin size bin packing problem with the unrestricted rotation of pieces. The problem is new to the literature and is a more general version of the 2D irregular bin packing problem. In this study, we introduce two constructive algorithms to generate feasible solutions for the problem. The first algorithm; denoted as CA1, is an extended method of the constructive method discussed in Chapter 4 which is supported by the Jostle heuristic in finding the input permutation of pieces. The second constructive algorithm; denoted as CA2, generates a feasible solution for a given order of pieces and uses a traditional bin selection heuristic in allocating pieces to the available bin types. Three main solution methodologies; Hybrid method of Genetic Algorithm/Jostle (HGA), Iterated Jostle with Random Assignment of Bins (IJRAB) and Sequential Packing with Bin Centric Heuristic (SPBCH), are designed to search for a solution. Each method consists of a unique way of changing the permutation of bins during the solution searching stage so that overall utilisation of bins is maximised. Out of the two constructive algorithms, CA1 supported by the Jostle heuristic demonstrates better results in terms of solution quality and computational complexity. Out of three main computational methods, HGA and IJRAB generate the better results and both used the approach of jostling pieces within a certain configuration of bins arranged in a certain sequence. The two methods follow different techniques to change the bin configuration, hence change the permutation pieces as well.

We also test different search improvement strategies and identify one version of IJRAB (IJRAB_A3) outperforms all the other search techniques in terms of finding quality solutions in quicker time. The IJRAB implements kicks to change the bin order and piece permutation when the search process becomes stuck at a local optimum. By running IJRAB with the proposed post-processing strategy; repacking of pieces in each of the occupied bins into the possible smallest bin, makes the search process more efficient. We also test different bin size configurations to test our algorithms and identify that a mix of bin sizes including very small bin sizes and very large bin sizes with moderate sized bins, leads to the best results in utilising the overall usage of bin area. The major findings of this study are applicable when a company requires making procurement decisions in a situation when there is a set of standard sheets available in multiple sheet sizes for the sheet cutting process. The study also emphasises its use when there is an option to use off-cuts to cut irregular shapes in the cutting process. In both situations, the cut order is processed considering a heterogeneous set of input bin types. According to the findings in Chapter 5, the use of a mix of bin sizes is worthwhile in saving material rather than using a single bin size. Moreover, given a range of input bin types, selecting large bin sizes is advantageous than smaller bin sizes in terms of saving material. Clearly, having large bins in the pool of input bin types is also advantageous for irregular shape bin packing, not only to improve the utilisation but also to reduce the number of bins occupied in the final solution. In Chapter 6, the newly proposed IJRAB algorithm is used to minimise the bin waste and to minimise the total cost of setups, sheet material usage and off-cuts handling and storage costs, when residuals are considered for reusing.

In Chapter 6, our investigation is extended toward the implementation of the above algorithms in practical cases. Usually, the first problem addressed here is how a manufacturer can manage combinations of sheet sizes to reduce overall consumption of materials. In general, this can be achieved by adopting one of the following four operational policies; 1) use a mix of standard sheet sizes (nonUL-Mix), 2) use a single standard sheet size (nonUL-Single), 3) use a single standard sheet size with residual reuse (UL-Single) and 4) use a mix of standard sizes of sheets with residual reuse (UL-Mix). We experiment with sequences of cut orders occurring in consecutive time periods as data instances, to evaluate the outcomes of the above four operational policies. The general problem addressed here is the multi-period irregular bin packing problem. Each policy specifies the specific problem type; whether it is a multi-period irregular shape single bin size bin packing problem, multi-period irregular shape multiple bin size bin packing problem or multi-period irregular shape multiple bin size bin packing problem with usable leftovers. We design a computational method to solve these problems considering the multi-period sheet cutting optimisation with unknown future demands. The developed algorithmic tools are based on the best performing Iterated Jostle heuristic and IJRAB search algorithm described in chapters 4 and 5. When the objective is to minimise the trim loss, in comparison, the policies based on reusing residuals (i.e. off-cuts) lead to a significant reduction in trim-loss. The study of reusing residuals explicitly considers the constraint on the number of Usable Leftover (UL) types since it requires significant effort to handle and store the usable off-cuts which are heterogeneous in sizes. If the UL policy is imposed with a reasonable upper bound on the number of UL bin types, using only the largest standard bin sizes, performs best in terms of saving material. This is true when the considered time span has higher number of time periods (i.e.>9), since it requires a certain amount of time periods to be progressed to have a substantial mix of heterogeneous set of ULs to achieve better utilization than from a single standard bin type. Otherwise, for the situations where shorter time spans are considered, using a mix of standard bin types is advantageous. Even if the UL policy is imposed, in a multi-period run, it requires a substantial number of periods, a substantial number of ULs and their size heterogeneity to reach the steady state level which effectively utilises returned ULs to the warehouse. Otherwise, the Non-UL Mix policy achieves the best advantage in terms of saving sheet material.

Next, we extend this multi-period analysis by considering other cost factors involved, to analyse the effect of the same four different operational policies on nine possible operational cost scenarios. The nine scenarios are based on the relative values of cost variables, so that four scenarios represent the situations where the value of one cost variable is significantly higher than values of the other cost variables. Also, five intermediate scenarios were defined considering the cost sensitivity and trade-off arising in different economic situations. This covers material cost, setups cost and handling and storage cost of off-cuts, which affect the total cost of the process. Figure 6.8 summarises the findings of this study which works as a guide to select the best performing policy suitable for each scenario. Selecting the most appropriate policy in an intermediate scenario is critical since there is an intersection of policies. A shift of policies from one to another was noted when one cost paradigm is varying from a lower value to a higher value. Therefore, findings recommend selecting the most appropriate policy in such cases by analysing the cost variability.

In summary, the thesis has achieved its three main objectives in solving the target problems; 2D ISBSBPP, 2D IMBSBPP and 2D IBPP with ULs. The algorithms developed in this thesis have demonstrated their applicability in those problems by finding reasonably good solutions in quicker time. Using the algorithms developed, we have presented how solving multi-period irregular shape bin packing problem with usable leftovers can be applied to identify the most suitable operational policy for a given sheet cutting process. While contributing to the C&P field with a new set of solution methods, we fulfill the main purpose of this study, which is to develop computational methods for a set of practical problems arise in sheet cutting industries.

As key contributions, all the computational methods in this study facilitate packing irregular pieces by identifying the promising rotation of pieces when they are being placed in the bins rather than restricting the piece rotation to a predefined finite set of orientation angles. Compared to the other approaches, the approach proposed in the study does this more efficiently on a continuous scale of angles range from 0 to 360 degrees. Second, the application of Jostle heuristic in bin packing problems (2D ISBSBPP and 2D IMBSBPP) is new and includes additional features such as unrestricted rotation of pieces which was not available in the previous implementation of Jostle algorithms. We introduce this method as a fast heuristic for solving irregular shape bin packing problems. Compared to some computational techniques applied in the same context, the proposed computational methods does not require expensive software such as CPLEX to implement these algorithms. Third, the study has presented a computational approach for another variant of IBPP which has not been considered in the literature. While solving this problem, we achieve one of our aims on evaluating the reusing practices in material waste arise in sheet cutting processes. Also, we discussed the cost effectiveness of a set of well-known operational policies within nine different operational cost scenarios.

7.2 Future Work

There are several improvement areas related to the scope of this study that can be investigated as future studies.

The computational methods introduced in this study are based on the search frameworks designed to find good solutions. The proposed Jostle, Iterated Jostle and the other meta-heuristic approaches are subject to problem-specific parameter values. Even though we present better results compared to the results in the existing literature, still there is room for fine-tuning the parameters or altering the moves to enhance the search procedures.

One significant part of all these algorithms is a generation of no-fit polygons. Throughout the experiments, we realised, approximately 70% of the computational time is consumed for no-fit polygon generation. Therefore, it is always better to improve procedures of generating no-fit-polygons in future research.

Due to the use of geometric representation, the proposed methods always find an accurate representation and placement of the pieces. However, one limitation of this study is that every piece needs to be a polygonal shape. In future research, we recommend extending this to the pieces with curved edges that can enhance the scope of applicability to cases in sheet cutting industries where pieces have curved shapes.

Another important component of the developed algorithms is the generation of the initial solution. In this study, we mainly relied on random sequence of pieces as the initial permutation of pieces. However, further investigation of different pre-processing mechanisms such as clustering pieces, or an investigation of new rules to order pieces may be able to find efficient methods to improve the solution approach. As one of the matured research areas, the algorithms developed for rectangular shapes multiple bin size bin packing problems provide greater intuition for the irregular bin packing problem. We believe investigation on those computational techniques or their solution improvement strategies would help to enhance the solution methods developed for irregular bin packing.

Another main investigation of this study is on managing the reuse of residuals. In our study, we considered differentiating waste from usable leftovers if leftover size is larger than the smallest piece type. However, it would be better if further research is carried out to define an economical size. Considering leftovers which are smaller than this *economical* size, would become a cost burden to the company.

Sheet cutting process was the main common application area of the problems we considered in this study. Based on cut orders processed in different periods, we develop our analysis so that the outcome of this analysis can be incorporated into decision support tool to determine the most appropriate operational policy. In this study, our analysis was limited to major policies such as reusing residuals, purchasing multiple sheet sizes;

which are common to most of the sheet cutting industries. However, this effort can be further extended by being biased towards the organization-specific constraints. For example, manufacturers who follow philosophies such as lean manufacturing would impose different constraints to the storage and work-in-progress stocks of sheet material and residuals so that the overall multi-period model can be changed.

As another future area of research, it would be interesting to consider other variants of multi-period irregular bin packing problems since it has potential applications in manufacturing industries. We considered particularly the variant where demand cut quantities of a future period are unknown. This allowed us to solve irregular bin packing problem individually for each period since the future qualities to be cut is unknown. However, there are situations where manufacturers know the future cut quantities in advance or at least predicts order demand for future time periods. In such cases, the solutions have to be searched considering the optimisation throughout the time span rather than optimising within individual periods.

7.3 Summary

This thesis has studied the irregular shape bin packing problem for homogeneous and heterogeneous bin types. Several algorithms have been designed and tested. These have produced competitive results and as a result, we have been able to draw insight about the problem.

Appendix A

Period wise Results for Model 01

Table A.1: Multi period cutting with residual use (starting with Single standard 2.25m bins)

Period	U_t			
	$\mathrm{J}_{max} = 32$	$J_{max} = 16$	$J_{max} = 4$	J_{max}=0
1	0.6015	0.6080	0.6010	0.5965
2	0.6145	0.6170	0.6002	0.5958
3	0.6399	0.6453	0.6082	0.5934
4	0.6374	0.6461	0.6288	0.5946
5	0.6545	0.6530	0.6317	0.5929
6	0.6558	0.6535	0.6332	0.5913
7	0.6602	0.6580	0.6409	0.5921
8	0.6655	0.6664	0.6456	0.5927
9	0.6682	0.6710	0.6496	0.5930
10	0.6685	0.6716	0.6474	0.5922
11	0.6718	0.6732	0.6475	0.5934
12	0.6721	0.6738	0.6451	0.5930
13	0.6745	0.6756	0.6500	0.5936
14	0.6723	0.6745	0.6482	0.5929
15	0.6720	0.6753	0.6482	0.5926
16	0.6739	0.6759	0.6479	0.5917

Table A.2: Multi period cutting with residual use (starting with Single standard 2.00m bins)

Period	U_t			
	J$_{max}$ = 32	J_{max} = 16	J_{max} = 4	J_{max}=0
1	0.5972	0.5925	0.5944	0.5907
2	0.6414	0.6428	0.6406	0.5925
3	0.6442	0.6560	0.6517	0.5942
4	0.6543	0.6592	0.6590	0.5947
5	0.6590	0.6658	0.6577	0.5970
6	0.6676	0.6648	0.6610	0.5970
7	0.6666	0.6700	0.6640	0.5978
8	0.6702	0.6719	0.6635	0.5983
9	0.6753	0.6723	0.6665	0.5993
10	0.6759	0.6730	0.6656	0.5982
11	0.6738	0.6731	0.6652	0.5968
12	0.6756	0.6739	0.6656	0.5970
13	0.6781	0.6781	0.6671	0.5966
14	0.6786	0.6767	0.6677	0.5969
15	0.6800	0.6771	0.6669	0.5971
16	0.6816	0.6786	0.6670	0.5970

Table A.3: Multi period cutting with residual use (starting with Single standard 1.75m bins)

Period	U_t			
	J$_{max}$ = 32	J_{max} = 16	J_{max} = 4	J_{max}=0
1	0.6109	0.6145	0.6092	0.6115
2	0.6463	0.6389	0.6381	0.6145
3	0.6531	0.6492	0.6527	0.6108
4	0.6520	0.6548	0.6553	0.6102
5	0.6612	0.6622	0.6551	0.6109
6	0.6605	0.6655	0.6571	0.6110
7	0.6622	0.6654	0.6622	0.6133
8	0.6680	0.6684	0.6638	0.6164
9	0.6698	0.6689	0.6636	0.6169
10	0.6713	0.6683	0.6611	0.6152
11	0.6726	0.6696	0.6633	0.6139
12	0.6725	0.6706	0.6636	0.6136
13	0.6733	0.6721	0.6646	0.6128
14	0.6735	0.6730	0.6640	0.6128
15	0.6725	0.6741	0.6620	0.6131
16	0.6734	0.6739	0.6624	0.6122

Table A.4: Multi period cutting with residual use (starting with Single standard 1.50m bins)

Period	U_t			
	J$_{max}$ = 32	J_{max} = 16	J_{max} = 4	J_{max}=0
1	0.6231	0.6248	0.6281	0.6257
2	0.6408	0.6527	0.6552	0.6214
3	0.6473	0.6553	0.6580	0.6225
4	0.6584	0.6594	0.6611	0.6214
5	0.6609	0.6586	0.6646	0.6216
6	0.6616	0.6608	0.6643	0.6209
7	0.6624	0.6651	0.6657	0.6220
8	0.6638	0.6666	0.6668	0.6227
9	0.6646	0.6668	0.6660	0.6235
10	0.6643	0.6662	0.6670	0.6219
11	0.6636	0.6674	0.6676	0.6213
12	0.6653	0.6676	0.6674	0.6207
13	0.6669	0.6676	0.6677	0.6206
14	0.6684	0.6689	0.6679	0.6207
15	0.6687	0.6698	0.6681	0.6209
16	0.6687	0.6703	0.6682	0.6211

Table A.5: Multi period cutting with residual use (starting with Single standard 1.25m bins)

Period	U_t			
	J$_{max}$ = 32	J_{max} = 16	J_{max} = 4	J_{max}=0
1	0.5917	0.5891	0.5936	0.5941
2	0.6081	0.5906	0.5920	0.5977
3	0.6111	0.6023	0.5922	0.5970
4	0.6123	0.6025	0.5921	0.5980
5	0.6158	0.6058	0.5934	0.5979
6	0.6178	0.6044	0.5970	0.5984
7	0.6197	0.6038	0.5954	0.5992
8	0.6216	0.6047	0.5981	0.5996
9	0.6217	0.6030	0.5972	0.5994
10	0.6225	0.6038	0.5976	0.5989
11	0.6231	0.6040	0.5989	0.5987
12	0.6241	0.6046	0.5991	0.5993
13	0.6243	0.6052	0.5998	0.5994
14	0.6255	0.6049	0.6008	0.5990
15	0.6255	0.6056	0.6010	0.5989
16	0.6257	0.6059	0.6010	0.5986

Table A.6: Multi period cutting with residual use (starting with Mix of standard sizes)

Period	U_t			
	J$_{max}$ = 32	J_{max} = 16	J_{max} = 4	J_{max}=0
1	0.6539	0.6516	0.6507	0.6557
2	0.6608	0.6617	0.6596	0.6517
3	0.6626	0.6671	0.6656	0.6584
4	0.6639	0.6625	0.6690	0.6567
5	0.6642	0.6650	0.6691	0.6572
6	0.6669	0.6681	0.6682	0.6563
7	0.6669	0.6670	0.6676	0.6543
8	0.6684	0.6697	0.6682	0.6540
9	0.6707	0.6707	0.6696	0.6543
10	0.6704	0.6712	0.6692	0.6541
11	0.6686	0.6700	0.6656	0.6503
12	0.6703	0.6710	0.6661	0.6503
13	0.6703	0.6713	0.6667	0.6503
14	0.6714	0.6721	0.6668	0.6500
15	0.6722	0.6722	0.6667	0.6499
16	0.6721	0.6721	0.6667	0.6499

References

Abuabara, A. and Morabito, R. (2009). Cutting optimization of structural tubes to build agricultural light aircrafts. *Annals of Operations Research*, 169(1):149–165.

Adamowicz, M. and Albano, A. (1972). A two-stage solution of the cutting stock problem. *Information Processing*, 71:1086–1091.

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

Agarwal, P., Flato, E., and Halperin, D. (2002). Polygon decomposition for efficient construction of minkowski sums. *Computational Geometry Theory and Applications*, 21:39–61.

Albano, A. and Sapuppo, G. (1980). Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *IEEE Transactions on Systems, Man and Cybernetics*, 10(5):242–248.

Alvarez-Valdes, R., Martinez, A., and Tamarit, J. M. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, 145(2):463–477.

Andrade, R., Birgin, E. G., and Morabito, R. (2016). Two-stage two-dimensional guillotine cutting stock problems with usable leftover. *International Transactions in Operational Research*, 23(1-2):121–145.

Andrade, R., Birgin, E. G., Morabito, R., and Ronconi, D. P. (2014). Mixed integer programming models for two-dimensional non-guillotine cutting problems with usable leftovers. *Journal of the Operational Research Society*, 65(11):1649–1663.

Arbib, C. and Marinelli, F. (2005). Integrating process optimization and inventory planning in cutting-stock with skiving option: An optimization model and its application. *European Journal of Operational Research*, 163(3):617–630.

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

Babu, A. R. and Babu, N. R. (1999). Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. *International Journal of Production Research*, 37(7):1625–1643.

Babu, A. R. and Babu, N. R. (2001). A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. *Computer-Aided Design*, 33(12):879–891.

Balaprakash, P., Birattari, M., and Stützle, T. (2007). *Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement*, volume 4771 of *Lecture Notes in Computer Science*, pages 108–122. Springer.

Baldacci, R., Boschetti, M. A., Ganovelli, M., and Maniezzo, V. (2014). Algorithms for nesting with defects. *Discrete Applied Mathematics*, 163, Part 1:17–33.

Baum, E. (1986). Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, Caltech, Pasadena, CA.

Baum, E. B. (1987). Towards practical 'neural' computation for combinatorial optimization problems. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 53–58, Woodbury, NY, USA. American Institute of Physics Inc.

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

Bennell, J. A. and Dowsland, K. A. (1999). A tabu thresholding implementation for the irregular stock cutting problem. *International Journal of Production Research*, 37(18):4259–4275.

Bennell, J. A. and Dowsland, K. A. (2001). Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science*, 47(8):1160–1172.

Bennell, J. A., Dowsland, K. A., and Dowsland, W. B. (2001). The irregular cutting-stock problem : a new procedure for deriving the no-fit polygon. *Computers & Operations Research*, 28(3):271–287.

Bennell, J. A. and Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184(2):397–415. 00128.

Bennell, J. A. and Oliveira, J. F. (2009). A tutorial in irregular shape packing problems. *The Journal of the Operational Research Society*, 60:s93–s105.

Bennell, J. A., Scheithauer, G., Stoyan, Y., and Romanova, T. (2010). Tools of mathematical modelling of arbitrary object packing problems. *Annals of Operations Research*, 179(1):343–368.

Bennell, J. A. and Song, X. (2008). A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. *Computers & Operations Research*, 35(1):267–281.

Bennell, J. A. and Song, X. (2010). A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 16(2):167–188. 00037.

Bertrand, J. W. M. and Fransoo, J. C. (2002). Operations management research methodologies using quantitative modeling. *International Journal of Operations & Production Management*, 22(2):241–264.

Birattari, M. (2009). *Tuning Metaheuristics: A Machine Learning Perspective*. Springer Science & Business Media.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A Racing Algorithm for Configuring Metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 02, pages 11–18, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010). F-Race and Iterated F-Race: An Overview. In *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer, Berlin, Heidelberg.

Błażewicz, J., Hawryluk, P., and Walkowiak, R. (1993). Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, 41(4):313–325. 00109.

Bohme, D. and Graham, A. (1979). Practical experiences with semiautomatic and automatic part nesting methods. *Computer Applications in Automation of Shipyard Operations and Ship Design III*, pages 213–220.

Bounsaythip, C. and Maouche, S. (1997). Irregular shape nesting and placing with evolutionary approach. In *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3425–3430.

Brooks, R. L., Smith, C. A. B., Stone, A. H., and Tutte, W. T. (1940). The dissection of rectangles into squares. *Duke mathematical Journal*, 7:312–340.

Brown, A. R. (1971). *Optimum packing and depletion: The computer in space - and resource-usage problems*. American Elsevier, London, New York.

Burke, E., Hellier, R. S. R., Kendall, G., and Whitwell, G. (2006). A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, 54(3):587–601.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

Burke, E. K., Hellier, R. S. R., Kendall, G., and Whitwell, G. (2007). Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 179(1):27–49.

Burke, E. K., Hellier, R. S. R., Kendall, G., and Whitwell, G. (2010). Irregular packing using the line and arc no-fit polygon. *Operations Research*, 58(4-Part-1):948–970.

Carravilla, M. A., Ribeiro, C., and Oliveira, J. F. (2003). Solving nesting problems with non-convex polygons by constraint logic programming. *International Transactions in Operational Research*, 10(6):651–663.

Charalambous, C. and Fleszar, K. (2011). A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. *Computers and Operational Research*, 38:1443–1451.

Chernov, N., Stoyan, Y., Romanova, T., Pankratov, A., Chernov, N., Stoyan, Y., Romanova, T., and Pankratov, A. (2012). Phi-Functions for 2D Objects Formed by Line Segments and Circular Arcs, Phi-Functions for 2D Objects Formed by Line Segments and Circular Arcs. *Advances in Operations Research, Advances in Operations Research*, 2012, 2012:e346358.

Cherri, A. C., Arenales, M. N., and Yanasse, H. H. (2009). The one-dimensional cutting stock problem with usable leftover—a heuristic approach. *European Journal of Operational Research*, 196(3):897–908.

Cherri, A. C., Arenales, M. N., and Yanasse, H. H. (2013). The usable leftover one-dimensional cutting stock problem—a priority-in-use heuristic. *International Transactions in Operational Research*, 20(2):189–199.

Cherri, A. C., Arenales, M. N., Yanasse, H. H., Poldi, K. C., and Gonçalves Vianna, A. C. (2014a). The one-dimensional cutting stock problem with usable leftovers – A survey. *European Journal of Operational Research*, 236(2):395–402.

Cherri, A. C., Arenales, M. N., Yanasse, H. H., Poldi, K. C., and Gonçalves Vianna, A. C. (2014b). The one-dimensional cutting stock problem with usable leftovers – A survey. *European Journal of Operational Research*, 236(2):395–402.

Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M. B., Oliveira, J. F., and Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, 253(3):570–583.

Cui, Y., Song, X., Chen, Y., and Cui, Y.-P. (2016). New model and heuristic solution approach for one-dimensional cutting stock problem with usable leftovers:model and heuristic for 1D cutting stock problem with leftovers. *Journal of the Operational Research Society*.

Cui, Y., Yang, L., Zhao, Z., Tang, T., and Yin, M. (2013). Sequential grouping heuristic for the two-dimensional cutting stock problem with pattern reduction. *International Journal of Production Economics*, 144(2):432–439.

Cui, Y. and Yang, Y. (2010). A heuristic for the one-dimensional cutting stock problem with usable leftover. *European Journal of Operational Research*, 204(2):245–250.

Cuninghame-Green, R. (1989). Geometry, shoemaking and the milk tray problem |new scientist.

Dagli, C. H. and Hajakbari, A. (1990). Simulated annealing approach for solving stock cutting problem. In *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*, pages 221–223.

Dagli, C. H. and Tatoglu, M. Y. (1987). An approach to two-dimensional cutting stock problems. *International Journal of Production Research*, 25(2):175–190.

Dighe, R. and Jakiela, M. J. (1995). Solving Pattern Nesting Problems with Genetic Algorithms Employing Task Decomposition and Contact Detection. *Evolutionary Computation*, 3(3):239–266.

Dowsland, K. A. and Dowsland, W. B. (1995). Solution approaches to irregular nesting problems. *European Journal of Operational Research*, 84(3):506–521.

Dowsland, K. A., Dowsland, W. B., and Bennell, J. A. (1998). Jostling for position: local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, 49(6):647–658. 00051.

Dowsland, K. A., Vaid, S., and Dowsland, W. B. (2002). An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research*, 141(2):371–381.

Dréo, J. (2006). *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer Science & Business Media.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159.

Egeblad, J., Nielsen, B. K., and Odgaard, A. (2007). Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, 183(3):1249–1266. 00097.

Eiben, A. E. and Smith, J. E. (2013). *Introduction to Evolutionary Computing*. Springer Science & Business Media.

Elkeran, A. (2013). A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research*, 231(3):757–769. 00019.

Erjavec, J., Gradišar, M., and Trkman, P. (2012). Assessment of stock size to minimize cutting stock production costs. *International Journal of Production Economics*, 135(1):170–176.

Falkenauer, E. (1999). The worth of the uniform [uniform crossover]. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, page 782 Vol. 1.

Feo, T. A. and Resende, M. G. C. (1989). A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Operational Research Letters*, 8(2):67–81.

Fischetti, M. and Luzzi, I. (2008). Mixed-integer programming models for nesting problems. *Journal of Heuristics*, 15(3):201–226.

Fleischmann, M., Bloemhof-Ruwaard, J. M., Dekker, R., van der Laan, E., van Nunen, J. A. E. E., and Van Wassenhove, L. N. (1997). Quantitative models for reverse logistics: A review. *European Journal of Operational Research*, 103(1):1–17.

Foerster, H. and Wäscher, G. (2000). Pattern reduction in one-dimensional cutting stock problems. *International Journal of Production Research*, 38(7):1657–1676.

Freeman, H. and Shapira, R. (1975). Determining the Minimum-area Encasing Rectangle for an Arbitrary Closed Curve. *Commun. ACM*, 18(7):409–413.

French, M. L. and LaForge, R. L. (2006). Closed-loop supply chains in process industries: An empirical study of producer re-use issues. *Journal of Operations Management*, 24(3):271–286.

Friesen, D. K. and Langston, M. A. (1986). Variable-sized bin packing. *Computers & Operations Research*, 15(1):222–230.

Fujita, K., Akagi, S., and Hirokawa, N. (1993). Hybrid approach for optimal nesting using a genetic algorithm and a local minimization algorithm. In *in Proc. of the 19th Annual ASME Design Automation Conference, part 1*, pages 477–484.

Gendreau, M. and Potvin, J., editors (2010). *Handbook of Metaheuristics*. Springer, New York, 2nd edition.

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

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

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Gomes, A. M. (2013). Irregular Packing Problems: Industrial Applications and New Directions Using Computational Geometry. In *11th IFAC Workshop on Intelligent Manufacturing Systems*, pages 378–383.

Gomes, A. M. and Oliveira, J. F. (2001). Automatic marker making. In *4th Metaheuristics International Conference*, pages 47–52.

Gomes, A. M. and Oliveira, J. F. (2002). A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, 141(2):359–370.

Gomes, A. M. and Oliveira, J. F. (2006). Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3):811–829.

Gonçalves, J. F. and Resende, M. G. C. (2013). A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, 145:500–510.

Gonzalez, T. F. (2007). *Handbook of Approximation Algorithms and Metaheuristics.* CRC Press.

Gradišar, M., Jesenko, J., and Resinovič, G. (1997). Optimization of roll cutting in clothing industry. *Computers & Operations Research*, 24(10):945–953.

Gradišar, M., Kljajić, M., Resinovič, G., and Jesenko, J. (1999a). A sequential heuristic procedure for one-dimensional cutting. *European Journal of Operational Research*, 114(3):557–568.

Gradišar, M., Resinovič, G., and Kljajić, M. (1999b). A hybrid approach for optimization of one-dimensional cutting. *European Journal of Operational Research*, 119(3):719–728.

Gradisar, M. and Trkman, P. (2005). A combined approach to the solution to the general one-dimensional cutting stock problem. *Computers & Operations Research*, 32(7):1793–1807.

Grinde, R. B. and Cavalier, T. M. (1995). A new algorithm for the minimal-area convex enclosure problem. *European Journal of Operational Research*, 84(3):522–538.

Guide, J. V. D. R. (2000). Production planning and control for remanufacturing: industry practice and research needs. *Journal of Operations Management*, 18(4):467–483.

Guo, B., Peng, Q., Cheng, X., and Dai, N. (2015). Free-form contour packing based on material grid approximation and lowest-gravity-center methods. *Expert Systems with Applications*, 42(4):1864–1871.

Han, W., Bennell, J. A., Zhao, X., and Song, X. (2013). Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints. *European Journal of Operational Research*, 230(3):495–504.

Heckmann, R. and Lengauer, T. (1995). A simulated annealing approach to the nesting problem in the textile manufacturing industry. *Annals of Operations Research*, 57(1):103–133.

Henn, S. and Wäscher, G. (2013). Extensions of cutting problems: setups. *Pesquisa Operacional*, 33(2):133–162.

Hifi, M. and M'Hallah, R. (2003). A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes. *International Transactions in Operational Research*, 10(3):195–216.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, Mass, new ed edition edition.

Hong, S., Zhang, D., Lau, H. C., Zeng, X., and Si, Y.-W. (2014). A hybrid heuristic algorithm for the 2d variable-sized bin packing problem. *European Journal of Operational Research*, 238(1):95–103.

Hopper, E. (2000). *Algorithms and other Meta-Heuristic Methods*. PhD thesis, University of Wales, UK.

Hopper, E. and Turton, B. C. H. (2001). A Review of the Application of Meta-Heuristic Algorithms to 2D Strip Packing Problems. *Artificial Intelligence Review*, 16(4):257–300.

Hu, Y., Hashimoto, H., Imahori, S., and Yagiura, M. (2015). Efficient implementations of construction heuristics for the rectilinear block packing problem. *Computers & Operations Research*, 53:206–222.

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Stützle, T. (2009). Paramils: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research*, 36:267–306.

Huyao, L., Yuanjun, H., and Bennell, J. A. (2007). The Irregular Nesting Problem: A New Approach for Nofit Polygon Calculation. *The Journal of the Operational Research Society*, 58(9):1235–1245.

Imamichi, T., Yagiura, M., and Nagamochi, H. (2009). An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6(4):345–361.

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1):165–181.

Junior, B. A., Pinheiro, P. R., and Saraiva, R. D. (2013). A Hybrid Methodology for Nesting Irregular Shapes: Case Study on a Textile Industry. *IFAC Proceedings Volumes*, 46(24):15–20.

Kallrath, J. (2008). Cutting circles and polygons from area-minimizing rectangles. *Journal of Global Optimization*, 43(2-3):299–328.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Journal of Management Science*, 6:366–422.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680.

Koch, S., König, S., and Wäscher, G. (2009). Integer linear programming for a cutting problem in the wood-processing industry: a case study. *International Transactions in Operational Research*, 16(6):715–726.

Konopasek, M. (1981). Mathematical treatments of some apparel marking and cutting problems. *U.S. Department of Commerce Report*, 99-26-90857-10.

Leao, A. A. S., Toledo, F. M. B., Oliveira, J. F., and Carravilla, M. A. (2016). A semi-continuous MIP model for the irregular strip packing problem. *International Journal of Production Research*, 54(3):712–721.

Leung, S. C. H., Lin, Y., and Zhang, D. (2012). Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, 39(3):678–686.

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

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

Liu, H. and He, Y. (2006). Algorithm for 2D irregular-shaped nesting problem based on the NFp algorithm and lowest-gravity-center principle. *Journal of Zhejiang University SCIENCE A*, 7(4):570–576.

Lopez-Camacho, E., Ochoa, G., Terashima-Marín, H., and Burke, Edmund, K. (2013a). An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, 206(1):241–264.

Lopez-Camacho, E., Terashima-Marín, H., Ochoa, G., and Conant-Pablos, S. E. (2013b). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics*, 145(1):488–499.

Lopez-Camacho, E., Terashima-Marin, H., Ross, P., and Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated Local Search: Framework and Applications. In Gendreau, M. and Potvin, J., editors, *Handbook of Metaheuristics*, number 146 in International Series in Operations Research & Management Science, pages 363–397. Springer US. DOI: 10.1007/978-1-4419-1665-5_12.

Luke, S. (2013). *Essentials of Metaheuristics.* lulu.com.

Mahadevan, A. (1984). *Optimization in Computer-aided Pattern Packing (Marking, Envelopes).* PhD thesis, North Carolina State University.

Martin, R. R. and Stephenson, P. C. (1988). Putting objects into boxes. *Computer-Aided Design*, 20(9):506–514.

Martinez-Sykora, A. (2013). *Nesting Problems: Exact and Heuristic Algorithms.* PhD thesis, University of Valencia, Spain.

Martinez-Sykora, A., Alvarez-Valdes, R., Bennell, J., and Tamarit, J. M. (2015). Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts. *Omega*, 52:15–32.

Martinez-Sykora, A., Alvarez-Valdes, R., Bennell, J. A., Ruiz, R., and Tamarit, J. M. (2017). Matheuristics for the irregular bin packing problem with free rotations. *European Journal of Operational Research*, 258(2):440–455.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Milenkovic, V., Daniels, K., and Li, Z. (1991). Automatic marker making. In *Third Canadian Conference on Computational Geometry*, pages 243–246.

Minner, S. (2001). Strategic safety stocks in reverse logistics supply chains. *International Journal of Production Economics*, 71:417–428.

Minner, S. (2003). Multiple-supplier inventory models in supply chain management: A review. *International Journal of Production Economics*, 81-82:265–279.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Morabito, R. and Belluzzo, L. (2007). Optimising the cutting of wood fibre plates in the hardboard industry. *European Journal of Operational Research*, 183(3):1405–1420.

Mutingi, M. and Mbohwa, C. (2016). *Grouping Genetic Algorithms: Advances and Applications*. Springer.

Oliveira, J. F. and Ferreira, J. A. S. (1993). *Algorithms for Nesting Problems*, pages 255–273. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg.

Oliveira, J. F., Gomes, A. M., and Ferreira, J. S. (2000). TOPOS-A new constructive algorithm for nesting problems. *OR-Spektrum*, 22(2):263–284.

Ortmann, F. G., Ntene, N., and van Vuuren, J. H. (2010). New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. *European Journal of Operational Research*, 203(2):306–315.

Peng-Yeng, Y. (2012). *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*. IGI Global.

Pisinger, D. and Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154–167.

Qu, W. and Sanders, J. L. (1987). A nesting algorithm for irregular parts and factors affecting trim losses. *International Journal of Production Research*, 25(3):381–397.

Resende, M. G. C. and Ribeiro, C. C. (2010). Greedy Randomized Adaptive Search Procedures. In *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, pages 219–249. Springer, Boston, MA.

Resende, M. G. C. and Ribeiro, C. C. (2016). GRASP: The basic heuristic. In *Optimization by GRASP*, pages 95–112. Springer New York.

Ribeiro, C., Carravilla, M. A., and Oliveira, J. (1999). Applying constraint logic programming to the resolution of nesting problems. *Pesquisa Operacional*, 19(2):239–247.

Rocha, P. (2014). *Geometrical models and algorithms for irregular shapes placement problems*. PhD thesis, INESC,Faculty of Engineering, University of Porto, Portugal.

Rocha, P., Rodrigues, R., Toledo, F. M. B., and Gomes, A. M. (2013). Circle Covering using Medial Axis. *IFAC Proceedings Volumes*, 46(7):402–407.

Rocha, P., Rui, R., Gomes, A. M., Andretta, M., and Toledo, F. M. B. (2014). Circle covering representation for nesting problems with continuous rotations. In *Proceedings of the 19th World Congress. The International Federation of Automatic Control*, pages 5235–5240, Cape Town, South Africa.

Roodman, G. M. (1986). Near-optimal solutions to one-dimensional cutting stock problems. *Computers & Operations Research*, 13(6):713–719.

Scheithauer, G. (1991). A note on handling residual lengths. *Optimization*, 22(3):461–466.

Segenreich, S. A. and Braga, L. M. P. F. (1986). Optimal nesting of general plane figures: A monte carlo heuristical approach. *Computer Graphics*, 10(3):229–237.

Sinuany-Stern, Z. and Weiner, I. (1994). The one dimensional cutting stock problem using two objectives. *The Journal of the Operational Research Society*, 45(2):231–236.

Song, X. and Bennell, J. A. (2014). Column generation and sequential heuristic procedure for solving an irregular shape cutting stock problem. *Journal of the Operational Research Society*, 65(7):1037–1052.

Stoyan, Y., Pankratov, A., and Romanova, T. (2015a). Cutting and packing problems for irregular objects with continuous rotations: mathematical modelling and non-linear optimization. *Journal of the Operational Research Society.*

Stoyan, Y., Pankratov, A., and Romanova, T. (2015b). Quasi-phi-functions and optimal packing of ellipses. *Journal of Global Optimization*, pages 1–25.

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

Stoyan, Y., Scheithauer, G., Gil, N., and Romanova, T. (2004). Phi-functions for complex 2D-objects. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(1):69–84.

Sweeney, P. E. and Paternoster, E. R. (1992). Cutting and Packing Problems: A Categorized, Application-Orientated Research Bibliography. *The Journal of the Operational Research Society*, 43(7):691–706.

Takahara, S., Kusumoto, Y., and Miyamoto, S. (2003). Solution for textile nesting problems using adaptive meta-heuristics and grouping. *Soft Computing*, 7(3):154–159.

Talbi, E. (2009). *Metaheuristics: From Design to Implementation.* John Wiley & Sons, Inc.

Terashima-Marín, H., Ross, P., Farías-Zárate, C. J., López-Camacho, E., and Valenzuela-Rendón, M. (2010). Generalized hyper-heuristics for solving 2D regular and irregular packing problems. *Annals of Operations Research*, 179(1):369–392.

Toledo, F. M. B., Carravilla, M. A., Ribeiro, C., Oliveira, J. F., and Gomes, A. M. (2013). The dotted-board model: A new MIP model for nesting irregular shapes. *International Journal of Production Economics*, 145(2):478–487.

Trkman, P. and Gradisar, M. (2007). One-dimensional cutting stock optimization in consecutive time periods. *European Journal of Operational Research*, 179(2):291–301.

Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K., and Ibaraki, T. (2009). Solving the irregular strip packing problem via guided local search for overlap minimization. *International Transactions in Operational Research*, 16(6):661–683.

Venkateswarlu, P. (2001). The trim-loss problem in a wooden container manufacturing company. *Journal of Manufacturing Systems*, 20(3):166–176.

Wang, F. K. and Liu, F. T. (2014). A new decision model for reducing trim loss and inventory in the paper industry. *Journal of Applied Mathematics*, 2014(Article ID 987054).

Wäscher, G. (2012). Cutting and packing typology: part1, part2, part3. EURO Summer Institute on Cutting and Packing (ESI-CP), Porto, Portugal.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130.

Watson, P. D. and Tobias, A. (1999). An efficient algorithm for the regular w1 packing of polygons in the infinite plane. *Journal of the Operational Research Society*, 50:1054–1062.

Xu, Y. (2016). An Efficient Heuristic Approach for Irregular Cutting Stock Problem in Ship Building Industry. *Mathematical Problems in Engineering*, 2016:e8703782.

Xu, Y., Yang, G., and Pan, C.-c. (2013). An Approach Based on Evaluation Particle Swarm Optimization Algorithm for 2D Irregular Cutting Stock Problem. In Tan, Y., Shi, Y., and Mo, H., editors, *Advances in Swarm Intelligence*, number 7928 in Lecture Notes in Computer Science, pages 168–175. Springer Berlin Heidelberg.