D 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

GEOMETRICAL MODELS AND ALGORITHMS FOR IRREGULAR SHAPES PLACEMENT PROBLEMS

PEDRO FILIPE MONTEIRO ROCHA

TESE DE DOUTORAMENTO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
ENGENHARIA INFORMÁTICA

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Geometrical Models and Algorithms for Irregular Shapes Placement Problems

Pedro Filipe Monteiro Rocha

Doctoral Program in Informatics Engineering

Supervisor: A. Miguel Gomes

Co-Supervisor: Rui Rodrigues

Julho de 2014

Geometrical Models and Algorithms for Irregular Shapes Placement Problems

Pedro Filipe Monteiro Rocha

Doctoral Program in Informatics Engineering

Julho de 2014

Abstract

This thesis addresses the Irregular Piece Placement problem, also known as Nesting problem, while focusing on the resolution of real world instances with continuous rotations. The real world instances are considered very large instances containing many pieces with very complex outlines, where continuous rotations may be desired. The Nesting problem is presented, with a description of its characteristics, identifying the main challenges, and related problems. This is done through a literature review, considering the geometric representations and common solution approaches that are normally used, in order to identify possible paths to explore, and confirm its inherent difficulty in being efficiently tackled, specially when dealing with continuous rotations.

The geometric component of the Nesting problems is addressed by using a novel algorithm that generates Circle Covering representations based on the Medial Axis skeleton of a piece. This iterative algorithm enables control over the approximation error, which allows managing the trade-off between the quality of circle covering representation and the total number of circles. This algorithm allows producing coverings with different levels of quality, and with different types of covering, depending on the characteristics of the Nesting problem where they will be used, which have a significant impact on the feasibility of the final solution and in the tightness of the layout.

The solution approach is based on Non-Linear Programming models, from which several formulations were made, taking into account the size of the problem being addressed. These models fully support continuous rotations, and can produce feasible solutions with an acceptable computational cost. In order to tackle large instances, extensions to the NLP models are done, by aggregating constraints by type, and implementing other tweaks to reduce computational cost.

In order to address issues with high computational cost, layout solutions with insufficient quality, and very large instances, three approaches are proposed. The first uses a two-step compaction process, where the NLP model uses low resolution in the first step, and high resolution in the second. The second approach uses a two-phase compaction process, where in the first phase big pieces are compacted, considering certain parameters, and holes are created between the pieces, and in the second phase, the small pieces remaining are assigned and placed in the holes created in the first phase, and all compacted together. The last approach uses a multi-step approach, where pieces are separated in groups, and compacted into the layout in a sequential order, forming layers of pieces. These approaches show very promising results, being able to address the Nesting problem with continuous rotations, considering their purpose. If these approaches are combined they can be used to improve results of currently existing approaches.

Resumo

Esta tese aborda o problema de Posicionamento de Peças Irregulares, também conhecido como problema de Nesting, focado na resolução de instâncias reais destes problemas, considerando rotações contínuas. O problema de Nesting é apresentado com uma descrição das suas características, identificando os desafios principais e problemas relacionados. Isto é feito através de uma revisão da literatura actual, considerando representações geométricas e métodos de resolução mais utilizados, com o objectivo de identificar possíveis oportunidades que possam ser exploradas de forma a abordar o problema de Nesting com rotações livres de forma mais eficiente.

A componente geométrica do problema de Nesting é abordado com recurso a um novo algoritmo que cria uma representação baseada em cobertura por círculos, feita a partir de um esqueleto topológico de cada peça. Este algoritmo iterativo permite ter controlo sobre o erro de aproximação da peça, permitindo lidar com o compromisso entre a qualidade de representação e o número total de círculos produzido. Este algoritmo permite também produzir coberturas com níveis de qualidade distintos, assim como tipos de cobertura de círculos diferentes, orientados para problemas específicos de Nesting, produzindo um impacto significativo na admissibilidade da solução final e na qualidade de compactação.

O método de resolução utiliza modelos de programação não-linear, que foram obtidos através de diferentes formulações matemáticas do problema, tendo em conta o tamanho do problema a ser abordado. Estes modelos matemáticos suportam rotações livres, conseguindo produzir soluções admissíveis com um custo computacional razoável. De forma a poder lidar com instâncias de Nesting de grande tamanho, foram feitas extensões aos modelos matemáticos para reduzir o custo computacional, através da agregação de restrições.

Para poder lidar com vários problemas relacionados com alto custo computacional, soluções com qualidade insuficiente e instâncias de tamanho muito grande, foram propostas três abordagens. A primeira utiliza um processo de compactação baseado em duas etapas, que utiliza resolução baixa na primeira etapa, e uma resolução alta na segunda, de forma a compactar depressa, e ajustar as peças com qualidade. A segunda abordagem usa duas fases, compactando as peças grandes na primeira fase, criando buracos no espaço entre as peças grandes. Na segunda fase, as peças pequenas são atribuídas e colocadas nos buracos, sendo todas as peças compactadas de seguida. A terceira, e última abordagem utiliza várias etapas, separando inicialmente as peças em grupos distintos, e compactando-as numa ordem sequencial, formando camadas de peças, permitindo compactar grandes problemas, com custo computacional reduzido.

Estas abordagens apresentam resultados muito promissores, conseguindo abordar problemas de Nesting com rotações contínuas, tendo em conta as suas vantagens individuais. Combinando estas abordagens, ou aplicando-as de uma forma específica, pode melhorar os resultados de abordagens existentes.

Acknowledgements

First an foremost, I would like to give thanks to Prof. António Miguel Gomes, my scientific supervisor, for his great support and encouragement during the duration of this Thesis. I also would like to thank all professors, colleagues and friends at the workplace, especially Prof. Franklina Toledo, Prof. Marina Andretta, and to my co-supervisor, Prof. Rui Rodrigues for their support. I must also thank everyone who gave their input on this Thesis and its contents. The final thanks will go to my family, and close friends, for their dedication and full support during the progression of this Thesis.

Pedro Filipe Monteiro Rocha

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AABB	Axis Aligned Bounding Boxes
Bdist	Bone Distance
BSP	Binary Space Partition
BVH	Bounding Volume Hierarchy
C&P	Cutting and Packing
CC	Circle Covering
CCC	Complete Circle Covering
CCC-MA	Medial Axis followed by Circle Covering
Cdist	Circle Distance
ESICUP	EURO Special Interest Group on Cutting and Packing
HR	High Resolution
HRP	High Resolution Plus
ICC	Inner Circle Covering
ICCtc	Inner Circle Covering with tip covering
IFP	Inner Fit Polygon
IFR	Inner Fit Rectangle
ISPP	Irregular Strip Packing problem
kCC-MA	Medial Axis followed by k type Circle Covering
LP	Linear Programming
LR	Low Resolution
M1	Model 1
M1E	Model 1 with non-overlapping constraint aggregation
M1S	Model 1 without constraint aggregation
M1T	Model 1 with all constraints aggregated
M2	Model 2
M2E	Model 2 with non-overlapping constraint aggregation
M2S	Model 2 without constraint aggregation
M2T	Model 2 with all constraints aggregated
MA	Medial Axis
MEC	Minimum Enclosing Circle
MIP	Mixed-Integer Programming
NFP	No-Fit-Polygon
NLP	Non-Linear Programming
OBB	Oriented Bounding Boxes
ODP	Open Dimension problem

PCC	Partial Circle Covering
PCCtc	Partial Circle Covering with tip covering
SBSBPP	Single Bin Size Bin Packing problem
SKP	Single Knapsack problem
SLOPP	Single Large Object Packing problem
SSSCSP	Single Stock Size Cutting Stock problem
VHR	Very High Resolution

Chapter 1

Introduction

The Nesting problem is a common problem that arises in industries where a set of small pieces needs to be placed inside a larger container, without overlaps, with the objective of minimizing wasted space, or pieces need to be cut from raw material, with the objective of minimizing waste. This is a 2D problem, nearly identical to the Cutting and Packing (C&P) problem, except that the pieces are irregular. The efficient cutting of raw material in small pieces is a complex task with strong impact in industrial production costs. This problem is also known as the Irregular Piece Packing problem. It is commonly found in garment, footwear, metalworking and other industries. Each of these distinct industrial applications has its own defining characteristics, considering the type and shape of container (closed or with open dimension), the diversity, shape and size of the pieces, admissible orientations (fixed, discrete or continuous), existence of defects, preferable placement regions, among others. Each one of these constraints limits the types of approaches that can be used to address them.

The main characteristics of the Nesting problem are defined by its geometrical and combinatorial components. Taking into account the specific requirements for this problem, a good geometrical representation must be able to deal efficiently with overlap computation, support continuous translations and rotations, and have high quality piece representation with low approximation error. The limitations derived from the geometrical component impose additional constraints in the resolution of this problem, by preventing the use, or development, of approaches that can lead to high quality solutions. The difficulties that arise from the geometrical component are strongly correlated to the complexity of the pieces and their geometric representation. Since overlap detection is one of the most computationally expensive tasks that use computational resources, an efficient method can allow more computational resources to be better used for improving Nesting solutions. Additionally, due to the low computational efficiency of current geometric approaches, when dealing with continuous rotations, the piece placement orientations are usually restricted to a discrete set of admissible rotations. The current geometric approaches are also unable to use the piece representation directly, when the pieces outline contains non-straight edges, and usually

simplify the outline by approximating it by a set of straight lines. Without overcoming the geometric challenges, the combinatorial challenges cannot be tackled properly, and more efficient approaches cannot be developed and used.

The combinatorial component deals with the selection of which and in what order pieces are to be placed in the container. Due to the nearly infinite combinations that exist by combining the different sequences of the pieces, the arbitrary piece orientation and piece placement rule, achieving the optimal solution is an extremely difficult task.

Considering these difficulties, most of the approaches for Nesting problems are based on heuristics, which are capable of achieving good results, although they are not able to efficiently solve real world problems. Other approaches based on meta-heuristics or mathematical models can also be used efficiently to address Nesting problems, but only with discrete orientations. All these methods still cannot deal with the Nesting problem using continuous orientations. The difficulty increases enormously when using a continuous range, since the discrete set (which is usually small) can be searched quite efficiently. Approaches based on exact methods still require much development to be able to achieve good solutions in a reasonable time interval, but they are also the most promising approaches.

1.1 Research Questions

Considering the requirements of the Nesting problem with continuous rotations, the selection of the approaches used to tackle it is based on their promising characteristics. In order to achieve success, several aspects require being determined. The main questions that require being successfully answered are:

> **Can Circle Covering representations be used to represent irregular pieces with the required quality in order to solve the Nesting problem with continuous rotations?**

The accurate representation of irregular pieces, using Circle Covering representation, is expected to enable overlap verifications with a low computation cost, due to the simple comparison between circles. The main difficulty is how to produce a circle covering where its outline closely resembles the outline of the pieces, with the minimum possible approximation error.

> **Can the Circle Covering representation be used to successfully control the trade-off between approximation quality and total number of circles of the representation?**

Generating a very high quality circle covering is useless if it cannot be used in practical terms. Since the overlap computation is based on comparing pairs of circles from different pieces, the computational cost increases exponentially with an increase in the number of circles. Having an approach that can generate high quality circle coverings, but allowing controlling the number of circles produced is very important.

Can Non-Linear Programming models, based on Circle Covering representation, be formulated to successfully tackle the Nesting problem with continuous rotations?

A good solution approach must be able to deal with the requirements of the problem, such as full support for free-rotations, while being able to produce high quality layouts. Since the circle covering representation has a mathematical description that contains non-linear equations, using Non-Linear Programming models is a natural choice. Several NLP formulations might be required in order to obtain a NLP model that is able to successfully achieve a feasible layout, with acceptable quality.

Can the Non-Linear Programming models be used with Circle Covering representation to produce high quality layouts with high computational efficiency, when addressing the Nesting problem with continuous rotations?

The Non-Linear Programming formulations may be able to produce feasible, high quality, solutions but with a prohibitive computational cost. Formulating a Non-Linear Programming model that is able to solve problems efficiently, with different sizes and characteristics, while being able to manage the trade-off between layout quality and computational cost, is required to be able to employ different strategies in order to achieve the best solutions.

Can the developed approaches, based on the combination of Circle Covering and Non-Linear Programming models, be used to solve real world Nesting problems with continuous rotations?

The most difficult types of problems to be solved are real world problems. These problems have specific characteristics and requirements that must be taken into account, and usually have a very large size, with many pieces, and pieces with very complex geometry. Being able to tackle efficiently these difficult problems is the desired outcome, not being severely limited by the complexity of an instance.

1.2 Objectives and Contributions

The main objective of this work is to develop an approach to solve the Nesting problem with continuous rotations, for real world instances. To do this, one aim of this work, considering the characteristics of the problem, will be tackling its geometrical challenges. An efficient Circle Covering representation is to be developed with low approximation error that supports admissible continuous placement positions and orientations of the pieces, with low computational cost. Furthermore, this work also aims to explore new paths that allow solving Nesting problems with continuous rotations in a more efficient way, through the use of Non-Linear Programming models. These improvements are expected to offer similar or better packing results, competitive with current literature results, with an acceptable computational cost.

The Nesting problem is usually addressed through heuristics which usually follow pre-defined placement rules, dealing with either discrete or continuous translations and also discrete or fixed orientations. This combination is usually solved to address problems in the garment industry, where large continuous sheets of tissue have items cut from them, with great precision, while placed in a limited set of discrete orientations, and allowing small adjustments (around 5^o). This problem also arises in the metalworking industry, where pieces are cut from sheets of metal, while placed into arbitrary orientations. The pieces may require to be separated by a small distance, due to technological constraints from the cutting process. Another application arises when dealing with leathers, using the hides of animals to produce items. Their shapes are highly irregular, and may have defects, which limit the available placement positions. The pieces can also be placed in arbitrary orientations, although the hides usually have preferred regions to place certain pieces. Addressing this problem, with a broad support for the different requirements of each type of problem that arise in many industries, requires an adequate geometrical representation and an appropriate solution approach, that finds the best placement positions and orientations for each piece, with the minimum waste. Since many of these problems deal with mass-produced items, even small reductions in waste can produce a significant reduction in the economic cost.

This work focuses into two distinct scientific areas, Operational Research and Computational Geometry. While the main focus will be on the area of Operational Research, one cannot discard the development effort in the area of Computational Geometry since without it, improvements cannot be achieved. While this work is aimed solving problems on the field of Cutting and Packing, considering irregular pieces and continuous rotations, these approaches can be used in any other areas where similar problems appear. This work is expected to address currently existing limitations, and providing contributions in the form of an adequate geometrical representation for the Nesting problem (which enables controllable approximation to the piece, low overlap verification cost and support for continuous rotations) and contributing also a solution approach that produces feasible layouts (which uses the developed geometrical representation, with support for continuous rotations). These contributions will provide an approach that can have a significant impact in many industrial areas, for problems such as the ones presented before.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 describes the State of the Art where the most relevant concepts are presented allowing for a greater understanding of the proposed approach. It includes a description about Cutting and Packing problems with focus on Nesting problems. It also presents Geometric Representations and Solution Approaches used with this problem. Chapter 3 addresses the Geometrical Component of the problem, presenting the developed Circle Covering Representation and comparing it to existing solutions from the literature. Chapter 4 focuses on the solution approach using the previously developed Circle Covering together with a Non-Linear Programming model to solve the Nesting problem. Several model formulations are described, with possible improvements to enhance their efficiency. Chapter 5 presents additional methods that

allow tackling problems of higher difficulty, with greater number of pieces, layouts with defects, and more efficiently. Chapter 6 concludes the document with a discussion about the problem and the developed methods used to address it, comments about the results and mentions the remaining challenges for future work.

Chapter 2

The Problem of Nesting Irregular Shapes with Continuous Rotations

Cutting and Packing (C&P) problems can be described as the problem of cutting small items from a larger object, or placing small items into larger objects. C&P problems are classified as Nesting problems when they deal with irregular shaped pieces. These types of problems arise in many industries (such as industries that deal with garment, furniture, metalworking and footwear) where objects are required to be cut from larger objects of raw material, or required to be packed into an enclosed space. The garment industry deals with fabrics that are usually in the format of a strip. The fabric has a specific orientation, due to the patterns imprinted in it, which limits the possible orientations for the pieces. The pieces can be rotated by 180^{o}, and suffer small adjustments by about 5^{o} without noticeable effects on the final product. The cutting process is very precise, thus enabling the pieces to be packed with a very small separating distance between them. The production of wood components for the construction of furniture (and other components) also requires specific orientations, due to the characteristics of the wood (strength, color, among others) that depend on the orientation. There is also the possibility of forbidden placement regions due to imperfections on the wood. The separating distance between the pieces is larger than the one used in fabric since the cutting process deals with saws and drills that cause higher amount of waste. Considering footwear, the raw material might be natural or synthetic, which impose different constraints. The natural raw material consists in hides, which have all different outlines, with preferable placement positions (due to the variable quality in some parts of the hide) and also imperfections. When the raw material is a synthetic product, it is usually presented in a strip format, with uniform quality, and no defects. In this application, any orientation is allowed, and there is also a separating distance requirement. The problems dealing with the cutting of pieces from metal sheets also deal with some technological constraints. While the sheets can be described by strips or rectangular containers, with no limitations on orientation, the separating distance varies, depending on the tool used to cut the pieces. The requirement is based on the

precision of the tool, if the cutting process uses cold temperature, or the separating distance may take into account the expansion of the metal when cutting the pieces at high temperature, such when using laser or torch. The development of approaches to deal with real-world applications, such as these ones, must take into account their specific technological constraints.

Usually, when addressing these problems, the simpler they are, the easier is to achieve good results. If the particular problem requires the use of a small set of discrete placement positions and orientations, finding the optimal solution requires checking only the combinations that can be done with the values from that set. As the set of discrete placement positions and orientations becomes bigger (with more placement positions) the difficulty increases significantly. In its extreme case, when it is allowed using all possible placement and orientation positions inside a given region, the discrete set becomes a continuous. The difficulty in addressing the Nesting problem with continuous rotation is significantly higher than solving it with a discrete set.

This chapter focuses on reviewing the current literature, by presenting the most relevant information regarding the Nesting problem with continuous rotations. The contents were selected by taking into consideration the characteristics and requirements of the Nesting problem, describing the most used solution approaches, current focus of research, and potential unexplored and promising paths to be developed.

The first section of this chapter contains an introduction about Nesting and other C&P problems. The second section starts with an overview of the geometrical representations, followed by mathematical models that are used with specific types of geometrical representations. The fourth section explores different solution approaches for Nesting problems. The fifth section contains information about specific algorithms used with the geometrical representations in order to assist with several aspects regarding geometry.

2.1 Nesting and Other Cutting and Packing Problems

C&P problems deal with the efficient positioning of pieces into a given region, without overlap, with the goal of reducing waste. This type of problem is characterized by having a geometric component whose difficulty grows exponentially with the increase in shape complexity of the items and objects, and a combinatorial component, which also grows more difficult with the increase in their number. The geometric component of these problems deals with the geometry of the objects, ensuring non-overlap with viable placement positions and orientations. The combinatorial component deals with the selection of small items to be grouped and placed inside the larger objects. A popular approach to solve these problems is to organize pieces into a sequence, which defines the order in which the pieces are placed into the large object thus separating the geometrical component from the combinatorial component, reducing the difficulty of dealing with the problem.

Some examples of different dimensionality of a C&P problem can be found in problems dealing with cutting pipes, cables and steel bars, which are much less complex than problems dealing with the production of furniture, glass or clothing pieces. The first group of examples relates to

1D problems, in which dealing with the geometrical component is simplified, due to only using one axis to define the cutting position. The second group relates to a 2D problem, having the geometrical component to deal with the positioning of the items in two dimensions. A 3D problem is even more complex to solve than the previous two, and some examples can be found on Container Loading problems and Sphere Packing problems. If the shape of the items is congruent (i.e. same shape and size), the combinatorial component is simplified, since the order of the items is not relevant. When their shapes are not uniform, the combinatorial component becomes harder to solve, increasing also with a larger number of items.

Additionally, beyond the combinatorial and geometrical components, C&P problems main objective can be refined into two distinct components: maximizing output (maximizing the number of small items that can be placed into a limited number of large objects) or minimizing input (minimizing the number of large objects necessary to completely contain all small items). The difficulty in solving each variety of C&P problems is directly related to their intrinsic characteristics, and, moreover, the methods that can be applied to a specific problem cannot be directly and efficiently applied to another. Some of these characteristics are the shape of the pieces (strongly homogeneous or heterogeneous), continuous or discrete orientations, among others. Due to the problem specific characteristics, the research is focused on each specific variation of Cutting and Packing problems. This can only be achieved if each kind of problem can be categorized and compartmentalized, and for that reason, typologies are developed.

The distinction and identification of standard types of problems in Cutting and Packing provide a basis of scientific research which enable the development of models, algorithms and problem generators. This led to the development of C&P typologies, such as the typology by Wäscher et al. (2007). The typology organizes and classifies objects into homogeneous categories based on given sets of characterizing criteria. As mentioned in Wäscher et al. (2007), C&P problems have an identical structure which can be summarized into several components. Given two sets of elements, one with large objects and another with a set of small items, group some or all of the small items into subsets and assign them to one of the large objects, while ensuring that all the small items remain completely inside the large object, do not overlap and at the same time, optimizing a given single or multi-dimensional objective function. This implies that when solving the general problem, one is simultaneously finding the solution to several sub-problems, such as selecting the large objects, selecting and grouping the small items, how to assign the subset of small items to large objects, and achieving a layout with valid placement positions for small items. In some cases, when the problem is very complex due to its size, the sub-problems are addressed separately in order to simplify the problem, and make it solvable. In Wäscher et al. (2007), the typology organizes problems into an hierarchical structure. This can be seen on Fig. 2.1.

C&P related problem types may, or may not, be pure C&P problems. This is dependent on the objectives that are required to meet, if they contain additional aspects than those required by a C&P problem. As an example, a cutting problem that aimed the minimization of waste while aiming also for the minimization of the cutting path is considered an extended problem due to its additional aspects beyond pure C&P. If the aspects related to it are only from the C&P problem,

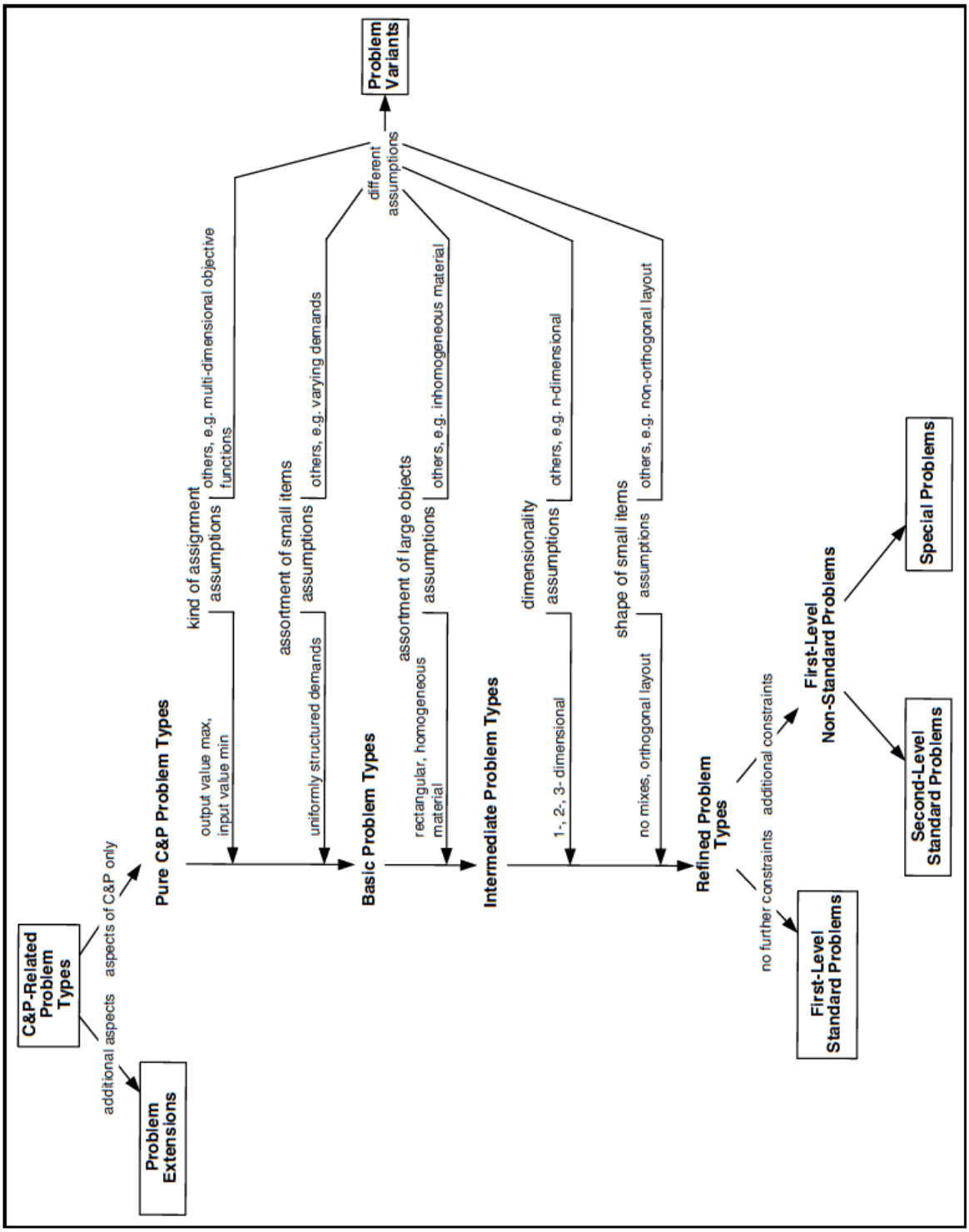


Figure 2.1: Cutting and Packing overview of problem types (adapted from (Wäscher et al., 2007))

then it is a pure C&P problem.

The pure C&P problems can be categorized by their kind of assignment (output maximization or input minimization), dimension boundaries and by the assortment of small items. These categories divide the C&P problem into basic problem types. The Fig. 2.2 illustrates the basic types of Cutting and Packing problems. Problems related to output maximization are known as Identical Item Packing problem, Placement problem and Knapsack problem. Some input minimization problems are Open Dimension problem, Cutting Stock problem and Bin Packing problem.

Basic problems can be further differentiated by the assortment of large objects, leading to the classification of intermediate problems. When the problem is differentiated by the assortment of large objects, problems such as Bin Packing problem can be defined as Single Bin Size Bin Packing problem or Multiple Bin Size/Residual Bin Packing problem, depending on having one or several large objects, and on the similarity of the pieces.

Intermediate problems can be further refined by applying the dimensionality criteria including also the criteria that indicates the shape of small items (rectangular, circular, non-regular, ...), and if they are mixed (such as using rectangles and circles at the same time). For example, if the Bin Packing problem is two dimensional, with several large objects and the small items are irregularly-shaped, it is defined as a Two-Dimensional Irregular Multiple Bin Size Bin Packing problem. When the dimensionality and shape of small items are considered, the Refined Problem Types arise. If they have additional constraints, such as discrete admissible orientations, the problem becomes a non-standard problem, otherwise, it becomes a standard problem.

Focusing on two-dimensional problems, they can be partitioned into regular packing (the pieces to be placed are regular polygons) problems and irregular packing problems (pieces to be placed have one or more non-regular polygons). By definition, regular polygons are all polygons in which the sides are all the same length, and all the angles are equal (Goodman and O'Rourke, 2004). When the pieces are represented by either a single or a set of circles, the problem can be dealt with the same difficulty as problems with regular polygons, but the characteristics of the circles might define the problem as an irregular problem. The differences between regular and non-regular polygons have a great influence on the approaches used to solve problems with them. For example, when dealing only with orthogonal rectangular pieces placement, the problem can be reduced to a discrete set of candidate positions, by defining a grid with feasible placement positions composed by the greatest common divisor of all the rectangular edges. This cannot be done with pieces that have non-regular shapes since there is an infinite variety of sizes and shapes, and this leads to an increased complexity of ensuring their correct placement into feasible placement positions (Bennell and Oliveira, 2009).

A Nesting problem is a two-dimensional Cutting and Packing problem where at least one piece with irregular shape must be placed in a configuration with other pieces, in order to optimize a given objective. Considering Wäscher et al. (2007), if the Nesting problem is defined by its use of non-regular pieces, to be placed into a strip, the problem is defined as 2D Irregular Open-Dimension Piece Placement problem. Another name for this type of problem that is commonly found on the literature is Irregular Strip Packing Problems. Although irregular shapes are usually

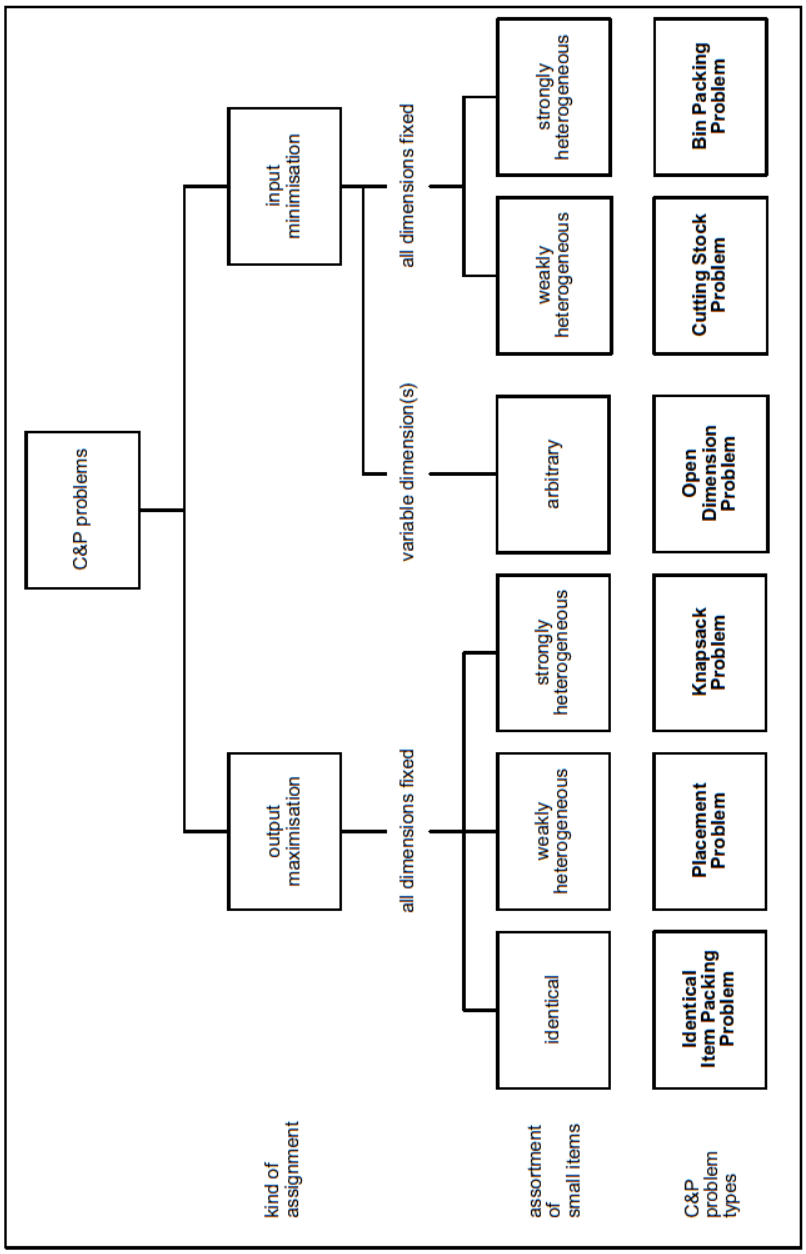


Figure 2.2: Basic C&P problem types (adapted from (Wäscher et al., 2007))

Table 2.1: Number of papers on selected problem types (table adapted from (Wäscher et al., 2007))

Problem types	1D	2D regular	2D irregular	3D	Total
ODP	–	46	49	7	102
SBSBPP	61	17	2	9	89
SKP	49	18	7	12	86
SLOPP	4	32	1	19	56
SSSCSP	29	2	1	6	38
Other	29	35	4	6	74
Total	172	150	64	59	445

defined as non-regular polygons, they can also be just simple polygons, have internal holes (i.e., multi-connected regions) or even have curved contours. Any piece that contains curved edges usually has those edges approximated by a series of linear segments (Bennell and Oliveira, 2008).

In Wäscher et al. (2007), the Cutting and Packing literature was reviewed and categorized, from 1995 to 2004, according to the presented typology. The literature was restricted to papers related to Cutting and Packing problems directly. From the research done, it was revealed that classic standard problems, such as Single Bin Size Bin Packing problem (SBSBPP), Open Dimensional Regular/Irregular Strip Packing problem (ODP), Single Stock Size Cutting Stock Problem (SSSCSP), Single Knapsack Problem (SKP), Single Large Object Packing Problem (SLOPP), and others, were well studied for three or more decades, being the area with greatest amount of publications. Other researches from these traditional problems usually focus on extensions of these problems on a higher dimension. Although relevant C&P problem extensions for real world applications are less commonly found in the literature, research is being continuously extended into those applications. A table with the number of papers on selected problem types, differentiated according to the number of problem-relevant dimensions can be seen on Table 2.1. From it, we can conclude that, in 2D, the focus of research is on Open Dimension problems, and also, that problems with regular polygons are preferred over the irregular ones. The difficulties in dealing with the geometry of irregular pieces, due to their complexity, explain, in these types of problems, the lack of appropriate tools that allow efficient solutions to be developed. With the development of new approaches to deal with the geometry of these problems, the preference of researchers can switch to these weakly explored problems. Only when the relevance of the geometric component is lowered, the combinatorial component can be properly explored and developed.

The focus of this work is the Nesting problem, which is classified as a 2D Open Dimensional Irregular Strip Packing Problem, with free-rotations that can solve real-world instances. For this reason, the remaining literature review in this chapter will have a higher focus on these problems.

2.2 Geometrical Representations for the Nesting Problem

The first obstacle that arises when dealing with Nesting problems is the geometrical component. To find feasible piece placement positions, the pieces have to be checked against each other for overlaps. While this is a simple task for any person, it is a very complex task for a computer

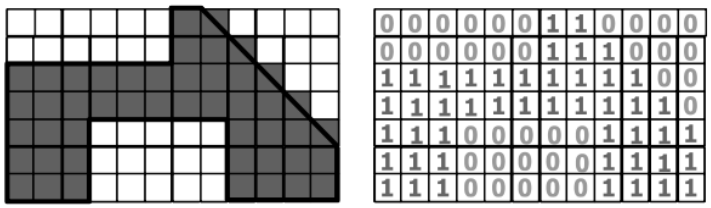


Figure 2.3: Binary Grid representation for irregular pieces (adapted from (Bennell and Oliveira, 2008))

program. Academic research in Nesting suffers from limitations caused by its geometrical characteristics, which cannot be tackled properly due to the lack of efficient tools and methods. However, some solutions exist, each one with its own advantages and disadvantages, not only regarding its performance, but also its difficulty of robust implementation.

The most common approaches to represent irregular shapes in Nesting problems are Grid and Polygonal representations. Another approach, less commonly used, is based on Circle Covering (CC) representation. These representations can either be approximated or exact, depending on the characteristics of the problem, for instance like the existence of curved contours or only orthogonal edges. For each one of these representations, some specific methods are required to efficiently compute overlaps, such as No-Fit-Polygons (NFP), Phi-Functions and Analytical Geometry for Circle Covering representations and Polygonal representations. For Grid representations, overlapping computations are based on matrix operations. The next sub-sections will introduce Grid representation, Polygonal representation, Phi-function representation and Circle Covering representation.

2.2.1 Grid Representation

Grid representation methods are approaches that divide a stock sheet into a grid, represented by a matrix, and also transforms the shapes of the pieces into an independent grid with the same block size as the stock sheet matrix. This approximation allows reducing the geometric information of the piece placement by only allowing discrete placement positions instead of continuous ones. Several placement algorithms exist, each with their different grid coding schemes.

The simplest coding scheme, proposed by Oliveira and Ferreira (1993) uses binary values where 0 refers to an empty space and 1 to a piece, as seen Fig. 2.3. The piece positions are easily represented on the matrix, adding values for 1 unit where the pieces are placed. At any position of the grid, the value correspond the number of pieces that exist in that position. If the number is above 1 then there is overlap among pieces.

A different coding scheme was proposed by Sagenreich and Braga (1986) that allows detection of contacts and overlaps among pieces, using the number 1 to represent the outline of the pieces, and number 3 represents the interior. Pieces are placed into their corresponding places on the matrix by adding the numbers on the matrix with the corresponding numbers of the piece. If

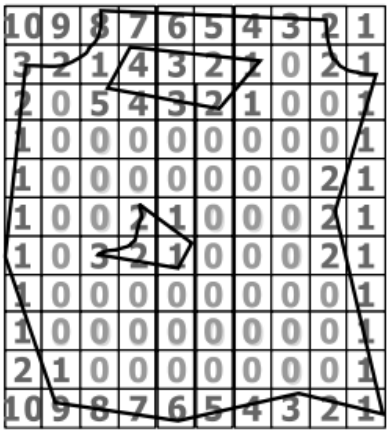


Figure 2.4: Grid representation method proposed by Babu and Babu (2001)

adding the numbers returns a value equal or greater than 4 (according to the referred codification) then the pieces are placed in infeasible positions. This means that the most recently placed pieces are overlapping with previously placed pieces. In this case, either both interiors overlap or an interior and a frontier of a piece overlap. Returning value 2 corresponds to contact among pieces, indicating feasible positions, and returning 0 for an empty space.

Babu and Babu (2001) reverse the previous idea by denoting empty space with numbers greater than or equal to 1 and the piece itself by 0. An example can be seen in Fig. 2.4. This particular codification, assigns a number greater than zero to any block outside of the boundary of the piece, assigning a 1 to the right most non-zero block, and adding 1 unit to each single block moving from right to left of the first block assigned 1.

With this method the value of each cell gives the minimum number of blocks that is necessary to move right to find a potential feasible position for the piece. This codification is particularly effective when using a bottom-left placement approach, since many blocks will be skipped in just one step when adjusting the pieces to a feasible position. As a downside, this codification has higher complexity when updating the layout of the matrix, which makes it more computationally expensive than previous approaches.

The advantages of the Grid representation methods described before are their easy representations of convex, non-convex and complex pieces and their fast verification of the geometric feasibility of the layouts on the matrix. Also, to calculate the distances that pieces must move to achieve a feasible position can be done by counting blocks in the desired direction.

As disadvantages, while these methods can consider parts and sheets with any geometry, being regular or irregular, and even with certain internal features like holes or defective regions (in sheets), they cannot represent accurately pieces with non-orthogonal edges and are more memory intensive than other methods. Increasing the accuracy involves increasing the resolution of the matrix by changing the size of the grid units, which increases the number of matrix blocks, leads to greater memory usage, higher running times and higher cost of feasibility checks. Alternative representations such as quad-trees could reduce the impact of some of these issues.

Grid representations are ill-suited to deal with rotations. The usual way to rotate pieces represented by grids is rotating the piece to the desired orientation and then coding it into the matrix again (which is similar to adding a completely new piece). Alternatively, it is possible to rotate the matrix itself. This is easily done for multiples of 90 degrees rotations, but for other angles it may cause additional difficulties, and therefore it is not used.

2.2.2 Polygonal Representation

An alternative to Grid representations is to use polygons to represent the irregular shape of the pieces directly. Comparing Polygonal representation methods with methods used with Grid representation cannot be done directly due to feasibility checks used in grids, which increase quadratically with the number of the edges of the pieces (Bennell and Oliveira, 2009), while the ones used with Polygonal representations increase exponentially. Polygonal representations can only be exact when the real outline is defined by straight edges. One advantage of Polygonal representations versus Grid representations is that the absolute size of the pieces does not influence the amount of piece representation information.

A shape that is described through straight segments will take the form of a polygon, if the real structures are correctly defined (i.e., with no self intersection edges or overlapped edges). In this representation, curves that may exist in the real world are approximated externally by straight segments, and particular details (which could be holes or others) are defined as other sets of straight edges.

Every curve is approximated with some error, which is reduced as the number of rectilinear edges used to represent it increase. With a polygonal representation of the pieces, the polygons are composed by sets of straight edges that have an oriented direction and the details inside it are defined with the reverse orientation. This orientation allows defining the inside and outside regions of each object. The main advantage of using oriented edges in the geometrical representation of shapes is the simplification of the operations to check intersections and overlaps.

The rotations using Polygonal representations are not complex to execute, neither to implement, but due to the detail of the polygonal geometry, the number of items to deal with increases with the number of vertices of the polygons. To rotate a polygon, all of its vertices need to be fixed relatively to other vertices of the same polygon, and rotated around the selected point.

Besides the previous mentioned operations, the simplification of the geometrical shapes by removing some edges and vertices and adding others to reduce their total number (while containing the original polygon inside) can also remove some of the concavities that increase the geometric complexity of the problem.

Although it is possible to deal with the concavities of the pieces, it is very hard to do it efficiently. One method is to use the approximation to a polygon by its convex hull. This method eliminates all concavities, and reduces the number of vertices, however, it reduces the quality of the resulting approximation. One alternative is the decomposition into convex polygons. Its main advantages are the computation of intersections and overlaps, but the disadvantage is the increased number of polygons, especially when dealing with concavities from approximations to

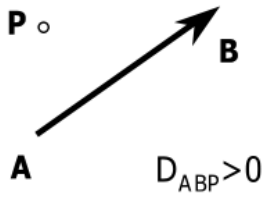


Figure 2.5: Visual representation of the D-function (adapted from (Bennell and Oliveira, 2008))

curved edges. The decomposition can be made in two ways, either by partition of an irregular polygon or by its covering.

Several representations exist that allow dealing with the complexity of overlapping detection and reducing the computational cost, such as polygons and NFP/IFP. These representations are presented in the following subsections.

2.2.2.1 Direct Polygon Representation

The D-Function (Konopasek, 1981), as seen in Fig. 2.5, is an efficient method to detect the position of a point relative to a vector. It is a mathematical expression based on the equation of distance between a straight line and a point. When given two vertices that represent a vector, the mathematical expression can verify the relative position of any other vertex to the supporting line of that segment. Assuming that the origin of the coordinate system is the bottom-left corner, and that the pieces have clockwise orientated edges, the D-function will be able to distinguish between inside and outside of the polygon by knowing that the outside is on the right side of the oriented edge, and the inside is on the left side of the vector.

The formula of the D-function (Eq. 2.1) can be defined as:

$$D_{abp} = ((X_a - X_b) \times (Y_a - Y_p) - (Y_a - Y_b) \times (X_a - X_p)) \tag{2.1}$$

where X_a, Y_a refer to the coordinates of point a, X_b, Y_b to the coordinates of point b, both belonging to the segment, and X_p, Y_p being the point whose relative position we want to determine.

Given an orientated segment, and comparing the relative position of a point to its supporting line using D-Function, the following conclusions can be achieved:

- If the result is greater than 0, the point is on the left side of the supporting line;
- If it is 0 then the point is over the supporting line;
- If the result is less than 0, the point is on the right side of the supporting line.

This function can be used to verify if two edges intersect or touch each other, without computing their intersection point, which would be computationally more expensive. Mahadevan (1984) described how the relative position of two oriented edges can be determined using the D-function.

The D-function is usually modified to return only three values, 1 for positive values, -1 for negative and 0 when is zero, for convenience when dealing with models and heuristics. This method cannot give a definite answer of intersection when the edges are touching, or are collinear, so another set of tests need to be done, by testing the next adjacent edge from one polygon to check if it is entering or exiting the other polygon. If the next adjacent edge is on the inside, then there is overlap.

These intersection tests are done by analyzing the relative position of all oriented edges with the help of D-functions. A further description of this method can be found on Bennell and Oliveira (2008).

The D-functions have the advantage of allowing the use of exact geometric representation of polygons. However, since polygons can be defined by any set of coordinates, therefore using floating-point representation, the computational cost and numerical precision errors are higher than the approaches that use mainly integers, such as Grid representation. To difficult things even further, every time a position is changed, all computations need to be redone, and feasibility of the pieces needs to be checked. This causes Nesting approaches based on iterative search processes to be discarded, since its computational time can be greatly increased by the geometrical computations, exceeding available time for searches, limiting the algorithm performance.

The Polygonal representations that use D-functions represent curved shapes using tangential approximations in the form of straight lines. Although there is always an approximation error, that error can be controlled with the number of straight lines used, but that also increases the complexity of the polygon that is created. A higher number of edges increases computational cost, and may lead to numerical precision errors.

This representation can be rotated in real time, but the rotations are not efficient, since it depends on the amount of vertices contained on each piece. Due to the number of edges of the pieces in this representation, the computational cost is very high. To compute overlap between pieces, every edge is compared with all other edges from all the remaining pieces.

2.2.2.2 No-Fit-Polygon Representation

The NFP is a geometric construction between two polygons, with oriented edges, that allows an easy verification if the two polygons are overlapping each other. This verification is easier compared to direct polygon comparison since it transforms the comparison between two polygons into a comparison between a point and a polygon. The NFP also has the advantage of being computed in a pre-processing stage, but usually only for discrete orientations, since it is not efficient to do so in real-time.

The NFP of polygon B relative to polygon A (NFP_{ab}) is the locus of points traced by the reference point R associated with B, when this polygon slides along the external contour of A. This is better explained through an example where one piece (B) slides around the other piece (A), being fixed in place, where both pieces never lose contact, and the resulting outline drawn by the reference point of the sliding piece becomes the NFP. The relative orientations of A and B are maintained during the sliding movement. Polygon B (the sliding polygon) must never overlap

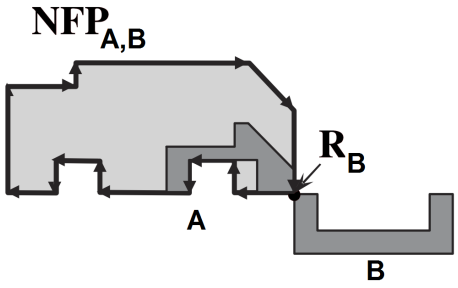


Figure 2.6: NFP between polygon A and polygon B (adapted from (Gomes and Oliveira, 2006))

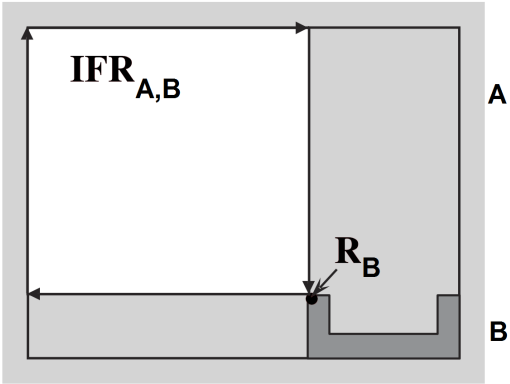


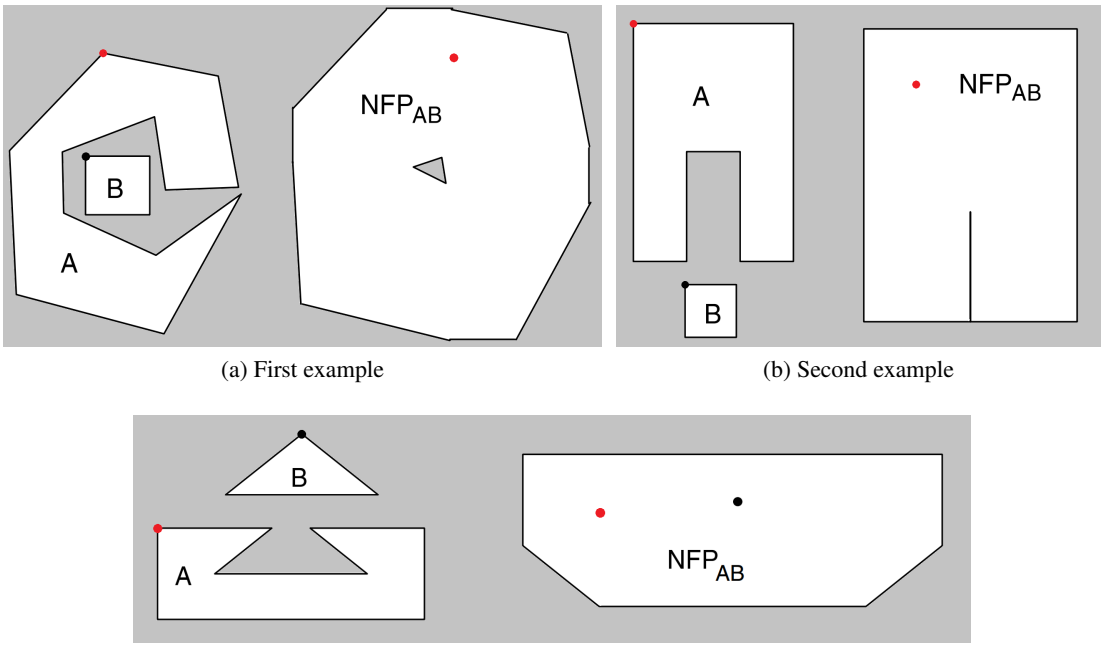
Figure 2.7: IFR between the board A and the polygon B (adapted from (Gomes and Oliveira, 2006))

A (the static polygon) and they must always be in contact (Gomes and Oliveira, 2002). If the operation is reversed, by using polygon B as static and A as the sliding one, the generated NFP, NFP_{BA} is equivalent to NFP_{AB} rotated by 180 degrees (Fig. 2.6).

If the reference point from the sliding polygon is inside the NFP_{AB}, then both polygons overlap. When the reference point is over the NFP_{AB} outline, then the polygons are touching, and when the reference point is outside of the NFP_{AB}, the polygons are far apart. This method also works if the origin of the static polygon is not placed at coordinates $(0,0)$. However, in these situations the reference point of B needs to be translated by $(-x,-y)$ being x and y the coordinates of the reference point A.

The Inner-Fit-Polygon (IFP) concept is derived from the NFP, and represents the set of points that allow the placement of a polygon inside a hole of another polygon, usually the container. The IFP is called Inner-Fit-Rectangle (IFR) if the outline of the IFP is a rectangle. The procedure to obtain the IFP is very simplified if the exterior polygon is a rectangle. This usually happens when generating the IFP from the container, since it is, most of the times, a rectangle. The IFR can be seen on Fig. 2.7.

The NFP can create degenerate cases which may not be simple polygons. This can happen when the polygon to be placed has a perfect fit position, or the other piece has a concavity where the other piece fits, but cannot slide from the outside to the inside and vice-versa. Examples



(a) First example

(b) Second example

(c) Third example

Figure 2.8: Example of combinations of polygons that generate NFPs and degenerated cases (adapted from (Bennell and Oliveira, 2009))

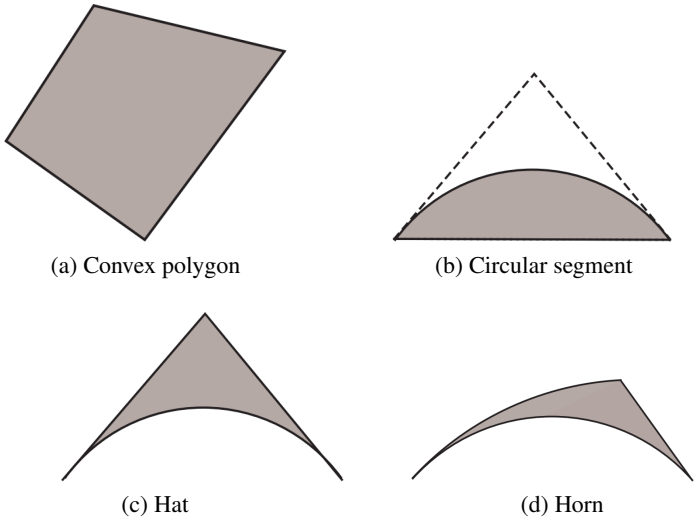
of degenerate cases can be seen on Fig. 2.8a. Fig. 2.8b shows a concavity where the polygon slides perfectly, and Fig. 2.8c shows a perfect fit of polygon A in polygon B, represented by the positioning point.

Updating the NFP due to changes in the orientations of the polygons requires generating the NFP in real-time computation, which is slow and prone to errors. For this reason, the NFPs are usually pre-computed before use, since it is not practical to use them with free-rotations. The difficulties in generating and using the NFP are properly described in section 2.5.4. In it, algorithms such as Orbiting, Minkowski Sum and Polygonal Decomposition are explained, discussing their advantages and disadvantages.

2.2.3 Phi-Function Representation

Phi-functions are basically mathematical expressions derived from the distance equation between two objects. They were developed and applied by Stoyan et al. (2001) and (Stoyan et al., 2004). Phi-functions have the purpose of representing all mutual positions of two objects, which is usually confused with the NFP because they are related concepts. In fact, the NFP corresponds to the zero level Phi-function.

Complex objects are decomposed into basic objects, as shown in Fig. 2.9, and all of these basic elements have circles and lines as their primitives. Any shapes not among the basic objects need to be represented as a finite combination of intersection, complement or union of basic objects,



(a) Convex polygon (b) Circular segment

(c) Hat (d) Horn

Figure 2.9: Basic objects for Phi-Functions (adapted from (Chernov et al., 2012))

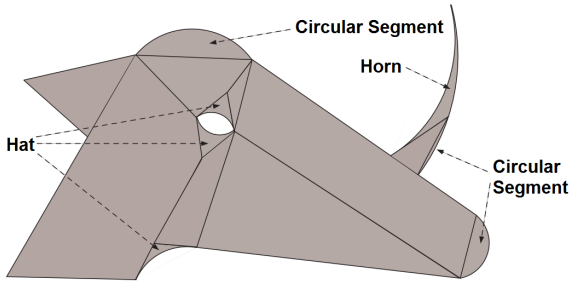


Figure 2.10: Representation of object by primary objects (adapted from (Chernov et al., 2012))

allowing for overlap among basic objects when describing more complex ones. The basis of the derivation of the Phi-functions is analytic geometry.

Phi-functions have several degrees of representation of the pieces. At the most basic level it uses primitive forms for geometrical comparisons, such as distance computation between a point and a line, and a point and a circle. The operations between these primitive forms involve using the D-function or compare the distance between two points (subtracting the radius). Sets of these primitive forms are aggregated to define primary objects, as seen in Fig. 2.9, which are compared against sets of the other primary objects to determine their relative position.

Combinations of primary objects are used to form composed objects that allow representing generic shapes (2.10). Comparing the relative positions between composed objects requires comparing the relative positions for each pair of primary objects from the composed objects. This increases the difficulty marginally, since the comparisons require that all pairs of primary objects from different composed objects not overlap. However, the comparisons between the sets of the low-level primitive forms of the primary objects have an increased difficulty, since at least one of the comparisons between the primitives must hold true to ensure non-overlapping.

The value of a given Phi-function is positive when the objects are far apart, zero when they

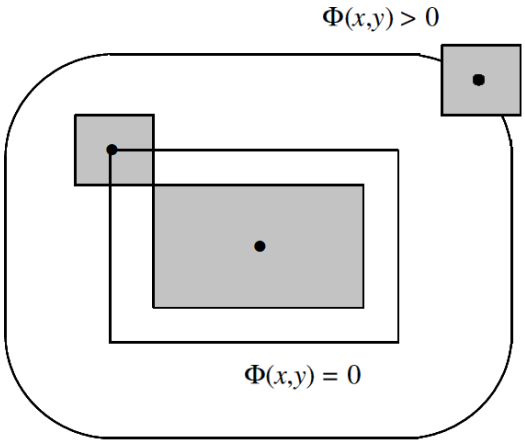


Figure 2.11: Phi-function taking into account two distinct positions (adapted from (Bennell and Oliveira, 2008))

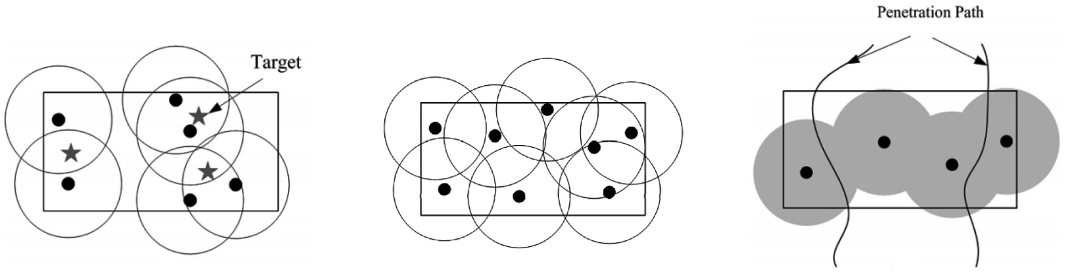
are touching, and are negative when there is overlap. In Fig. 2.11 two examples of Phi-function value can be seen, where one function returns zero (since the pieces are touching), while the other returns a positive value (since they are far apart).

For normalized Phi-functions this value is the Euclidean distance between two objects. Difficulties arise when using the normalized Phi-Functions due to radical expressions being present. Since there expressions are computationally expensive, and using them can increase the numerical precision errors, the solution is use radical-free expressions. This increases the performance of the models, although the Phi-functions will not return the Euclidean distance between objects. This approach has not been widely used due to several reasons, one being that Phi-functions require the use of Non-Linear Programming Models which are still very complex and difficult to deal with. They also increase exponentially in difficulty with the number of objects used to represent the pieces, since one composed object contains one or several primary objects, which in turn, contains several primitive forms. This growth in complexity is the primary reason that limits its widespread adoption. Although it has many advantages, it still needs further research in order to be widely adopted by the research community.

2.2.4 Circle Covering Representation

Covering problems are defined as minimization problems where the objective is to cover a region with one or several objects. A particular problem of this kind is the Circle Covering problem (CC), where the objects that cover the region are circles, and where the region being covered may also be a circle.

The objective of the problem is dependent on the type of problem to be solved. One possible aim is to minimize the radius of equal circles that are required to completely cover a region with a fixed number of circles, and another is to minimize the number of circles while maintaining a fixed radius, among other alternatives. Other problems require solving more than one single



(a) Target covering (b) Area covering (c) Barrier covering

Figure 2.12: Circle Covering examples (adapted from (Wang, 2011))

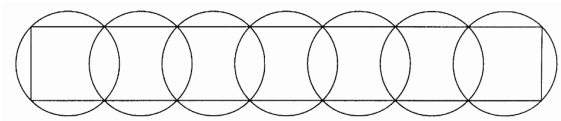


Figure 2.13: Circle Covering of a rectangle (adapted from (Melissen and Schuur, 2000))

objective, such as minimizing the number of unequal sized circles required to cover a region, within a specified approximation error, etc.

The C.C. problem is a fairly common problem in a wide range of scientific fields, such as in wireless networking (cellular coverage), collision detection, and many others. In wireless networking problems, the challenge consists in placing sensors inside a specified region, with the objective of covering the entire region, so that objects inside it can be monitored or tracked. They have maximum sensing ranges, a network without fixed infrastructure and a variable topology. The sensors are represented by circles and the circle radius represents the range of the devices. Wang (2011) surveys the coverage problems for wireless sensor networks.

Three examples of distinct covering objectives are presented in Fig. 2.12. In 2.12a the problem requires that all the targets inside a region be covered, while in 2.12b the whole area must be covered. The example in 2.12c presents an example that requires building a covering barrier or detecting penetration paths (paths from one point of the layout to another without coverage). Other covering problems can be found in Heppes and Melissen (1997), Melissen and Schuur (2000) which completely cover squares and rectangles with circles. One example of the covering of a rectangle with circles can be seen in Fig. 2.13

Using C.C. to represent a specific shape usually requires the complete enclosure of the shape inside the set of circles. However, for some problems, completely containing the covering circles inside the shape may provide better opportunities. A polygon (or any other shape) can be "covered" by circles in distinct ways, being one when the outline generated by the circles completely contains the outline of the polygon. The circles can also fill the polygon from the inside without crossing the outline of the polygon to its exterior, or they may approximate the polygon shape while exceeding the polygon outline in some regions, and being contained inside the polygon outline in other regions. Examples of C.C. without exceeding the outline of the polygons (and others that do

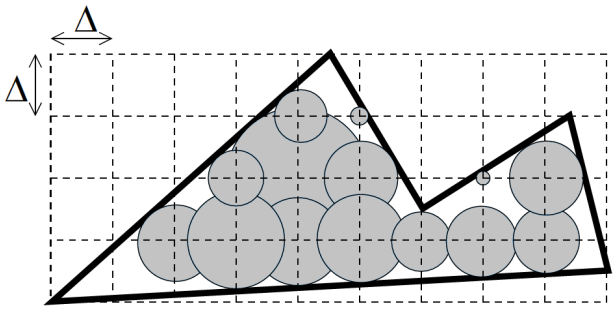
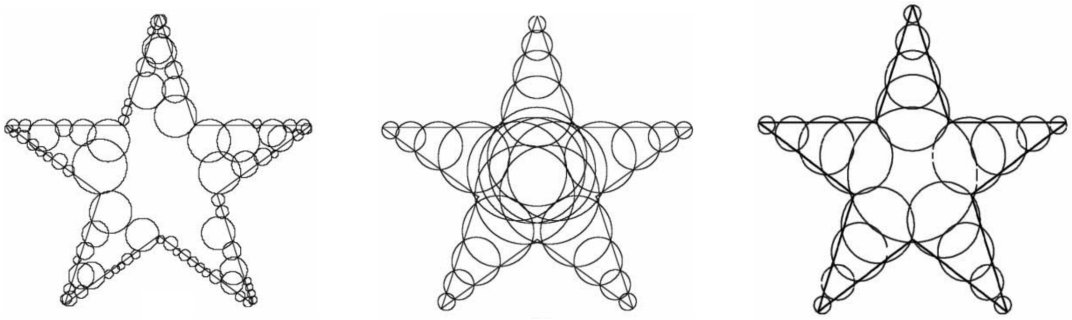


Figure 2.14: Grid based Circle Covering.

exceed it) can be found in (Matisziw and Murray, 2009). In (Matisziw and Murray, 2009), the problem requires dealing with the placement of facilities to maximize service coverage of regional demand.

When using C.C. to achieve a representation that can be used in Nesting problems, it is required that certain characteristics are satisfied. This requires dealing with a trade off of achieving total coverage of the region with a certain degree of approximation error to its outline but with the minimum number of circles. Geometrical form representations through circles have great advantages compared to other representation methods. As one example, this approximation allows reducing the geometric information of a piece outline by using only a set of circles of equal or unequal radius which their coverage outline approximates the outline of the piece. The circles have the advantage of being able to easily adjust to geometrical shapes composed by non-linear segments, straight edges and others. They are placed in a fixed position relatively to the reference point of the piece, with either equal or unequal circle radius between them, and the set of circles is considered a single piece. This set of circles can then be translated, rotated around a point, and placed in a feasible placement position. Rotations are trivial, since to rotate the representation of the piece only the centers of the circles need to be rotated. The computation of the overlap is easily and quickly done, since they only involve comparing the distances between circle centers and their radius. The circles from the same set are allowed to overlap, since they have to cover all the region of the piece, but they are not allowed to overlap other circles belonging to other sets.

For the problem of achieving a good representation through C.C., some approaches have been proposed. Imamichi and Nagamochi (2007) and Imamichi and Nagamochi (2008) proposed an algorithm which approximates an object by a set of circles/spheres and then computes a layout that positions all those sets of circles/spheres minimizing a penalty function that is based on the penetration depth between circles/spheres. His approach approximates the polygons by circles/spheres using an approach that requires creating a bounding box of the polygon and then making a grid with squares of a specific size Δ. Circles/spheres with the maximum size possible are then placed into all grid points that exist inside the polygon, while also removing any redundant circles. This method contains the set of circles that represent a piece inside the polygon that defines the piece outline. An example can be seen in Fig. 2.14.



(a) Contour approximation (b) Complete Circle Covering (c) Covering with gap

Figure 2.15: Outline and Three-Step(+Gap) Circle Coverings (adapted from (Zhang and Zhang, 2009))

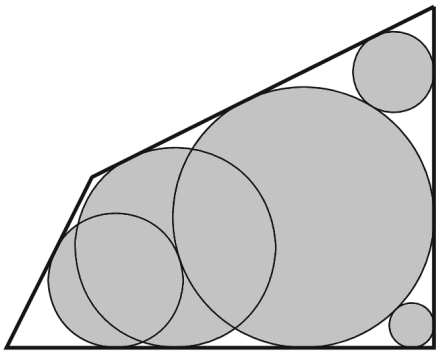
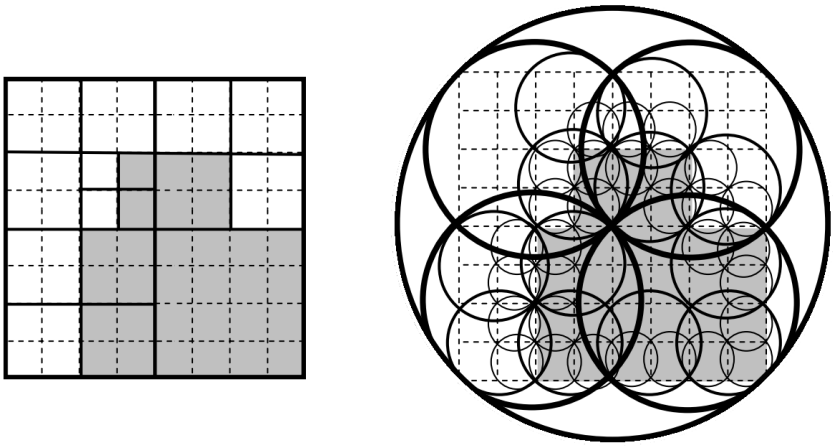


Figure 2.16: Greedy Inner Circle Covering (adapted from Jones (2013))

Several algorithms have also been proposed by Zhang and Zhang (2009). In one algorithm, they approximate the contour of the piece by circles (2.15a), and in another, the piece is covered completely by circles using a threshold value and placing the circles starting from the convex vertices (2.15b), which is called the Three-Step Algorithm. The threshold value is used to control the approximation error to the outline of the piece. A greater threshold allows for a greater approximation error. Zhang and Zhang (2009) also proposes a variation of the second algorithm with a gap (Three-Step Algorithm+Gap) that allows regions in the contour not to be covered by circles (2.15c). This is valid while there is a guarantee that any circle from other piece cannot cover that gap. Although this method does not technically completely cover the piece, since it is only a variation that reduces the required number of circles to "cover it", it can still be considered as such.

Another possible approach can be seen in Jones (2013) where a greedy heuristic starts by iteratively finding the largest circle that can be inscribed in the polygon, without overlapping the previous one, until a specified number of circles is reached. Each generated circle, that touches the outline of the polygon only once, is expanded until it becomes the largest circle where its center is on a line starting on the contact point and perpendicular to the contacted edge of the outline.



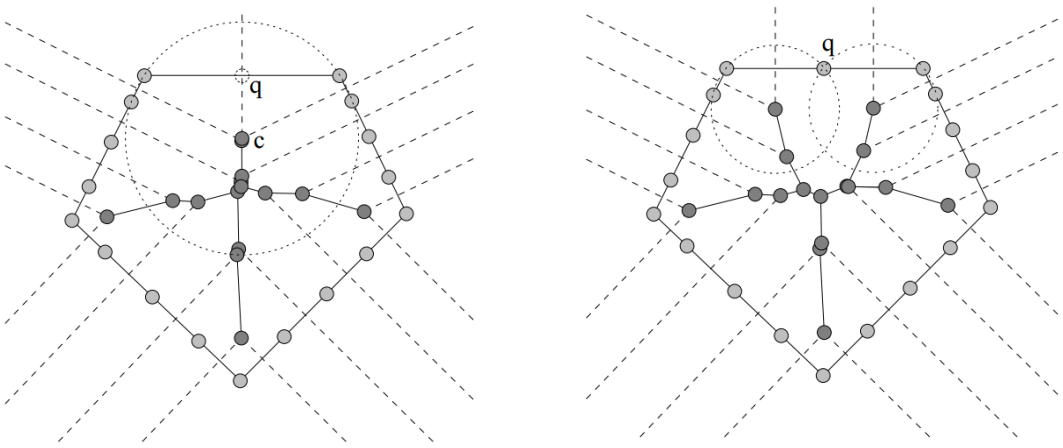
(a) Quad-tree (b) Circular Quad-tree

Figure 2.17: Quad-tree and circular quad-tree (adapted from (Moore, 2002))

The representation of a piece by circles (2D) or spheres (3D) can also be done with the assistance of a tree structure. Focusing in 2D, a simple approach is to divide a region into a quad-tree data structure and assign circles to each block, dividing each block into other four, until the desired level has been reached. Several authors have done this mainly for collision detection in 3D, such as Bradshaw (2002), but their idea is easily translated to 2D. Moore (2002) propose using a circle-tree (Fig.2.17b) , which is the equivalent of a quad-tree (Fig.2.17a) but using circles, to store and access efficiently spatial data. This approach can be used to represent pieces by circles but it produces a large number of them.

Bradshaw and O'Sullivan (2002) proposes a method to construct sphere-trees (circular hierarchical octree) for collision detection, using Voronoi diagram to approximate the Medial Axis. Objects (convex and non-convex) are iteratively divided into regions which are then approximated by enclosing circles. The algorithm tries to balance the size of the regions when dividing, in order to balance the size of the spheres. After placing the initial spheres, the algorithm tries to cover the rest of the uncovered points of the object, changing the structure of Voronoi diagram so that circles can better approximate the object, and also increasing the approximation to the Medial Axis, constructing it as necessary. An example of this can be seen in Fig. 2.18a (before changes) and Fig. 2.18b (after changes) where point q is the dividing point in the region and point c is the center of the circle to be eliminated. With every division, the collection of inner points will increasingly resemble the Medial Axis.

The usage of circles for geometrical representations has great advantages, such as the simplicity of the definition of a circle, which only needs a point and a radius. This feature is what makes so simple the computation of overlaps between circles, and their rotation. C.C. representation also has some disadvantages, mainly the difficulty in generating a covering due to numerical precision errors and the problem of finding the smallest number and positions of circles that cover a piece, within a maximum specified approximation error. If the method used to generate a C.C.



(a) M.A. approximation before division

(b) M.A. approximation after division

Figure 2.18: Reconstruction of Medial Axis approximation through Voronoi diagram (adapted from (Bradshaw and O'Sullivan, 2002))

representation does not return a low amount of circles for a given approximation error, the C.C. representation might be unusable due to the computational cost of overlap computation.

2.3 Modeling the Nesting Problem

A wide variety of different strategies were developed with the objective of producing good quality solutions to Nesting problems. In order to achieve an admissible solution, several methods can be categorized into two distinct approaches, methods that work with partial solutions and methods that work with complete solutions (Bennell and Oliveira, 2009). In the methods such as the former, the layout is constructed one piece at the time, and each piece is positioned into a feasible position, and not moved. The quality of the solution in this approach depends particularly on the placement order of the pieces, and the chosen placement rule. Some improvements to this approach allow changing the sequence of pieces dynamically and/or backtracking. For any method that chooses the latter approach, working with complete solutions, it is based on iteratively making small changes to a complete solution. These changes are based on searches over a sequence (changing sequence and/or orientation of pieces) and searches over a layout (allowing overlaps, compacting and separating pieces, and others).

The methods used are based on mathematical models (linear and mixed-integer programming models, and non-linear models), heuristics and improvement algorithms (greedy heuristic placement methods, meta-heuristic guided search techniques and others). Some papers containing more information about these approaches are Dowsland and Dowsland (1992) and Sweeney and Paternoster (1992). General positioning rules are required when working with either partial or complete solutions. They are not able to be directly applied to Nesting problems since they have to rely on geometric tools to deal with the geometry of the problem. The implementation of robust and

efficient geometry tools can be laborious and can often take considerably longer than the packing strategies themselves. Even with geometric limitations, many different approaches have been developed for Nesting problems. These approaches will be discussed on the following sections.

The geometrical representations (NFP, Circle Covering and Phi-functions) allow the conditions of no overlapping and piece placement to be defined, and through them, mathematical models can be derived with the objective to find a good solution to irregular piece placement problems. Each geometric representation leads to a specific set of constraints and objective function (both can be linear and/or non-linear) and lead to specific mathematical models. These models have in common a set of decision variables, such as the variables that define the position for every object on each axis, and their rotation, except when the model is derived from NFP, due to its fixed orientation. A model in which the objective function and all of the constraint functions are linear, and also continuous, is a linear programming model (LP).

Any model that deals with the piece placement problem of the Nesting problem has the same basic structure. The decision variables consist of the variables that define the positions of the pieces on the orthogonal axis, and the constraints are related to the non-overlapping between pieces, and the containment of the pieces inside the container. Additional decision variables and constraints are introduced depending on the specific requirements that arise due to the geometrical representation selected.

General model structure for Nesting problems:

- Objective Function: Minimize length
- Decision Variables: Positioning variables regarding orthogonal axis
- Constraints: Overlapping and Containment constraints

Since the model structure depends on the geometrical representation used, it leads to different types of model. The NFP produces linear constraints with binary variables, which leads to Mixed Integer Programming models (MIP). When the model focuses on solving the relative positioning problem, such as compaction/separation of the pieces, it uses only linear constraints which is a linear programming model. Phi-functions and Circle Covering representations are solved through Non-Linear Programming models since the constraints are non-linear.

A more detailed description about these models will be presented in the next subsections.

2.3.1 Models derived from No-Fit-Polygon

The model derived from the NFP has linear constraints obtained from the piece placement positions inside the large object. The non-overlapping constraints, besides being linear, also have binary variables associated with them. While the piece placement position conditions grow linearly with the number of irregular polygons, non-overlapping conditions have a factorial increase. The non-overlapping conditions lead to the use of MIP models due to the use of binary variables, which increase with the number of pieces and with the number edges of the NFPs. Since at least

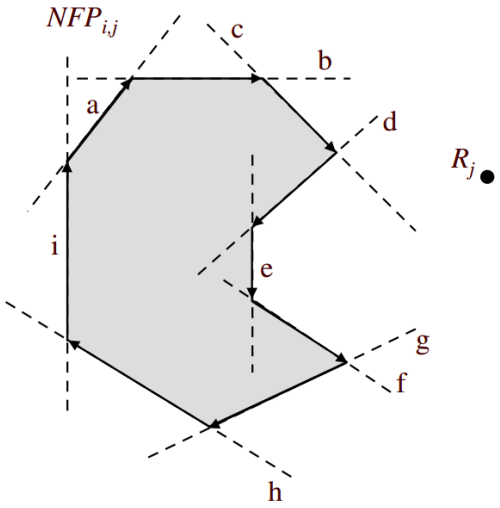


Figure 2.19: NFP edges associated with constraints and conjunctions of constraints (edges d, e and f) (adapted from Gomes and Oliveira (2006))

one of the constraints is required to be satisfied, using a MIP model is necessary. The binary variables are useful for defining which non-overlapping constraints are satisfied regarding the relative position of the point to the edge, or set of edges, belonging to the NFP. If the NFP is a convex polygon, all edges are associated with a single constraint. If the NFP is non-convex, some edges are associated with a conjunction of multiple constraints, composed by the constraints that bound the concavity. In Fig. 2.19, all of the edges represent one non-overlapping constraint for that piece, except the edges d, e and f, which define a conjunction of constraints representing the concavity. For the pieces to ensure non-overlapping, only one of the constraints, or conjunction of constraints, needs to be satisfied, indicating that the point is outside of the NFP. The pieces do not overlap if the point is placed over or on the left side of at least one of the oriented edges of the NFP. In Fig. 2.19, due to the position of point R_j, the binary variable associated with the edge c signals that there is no overlap, as also happens with the binary variable representing the conjunction of constraints for the edges d, e and f. Models with these characteristics have reduced practical applications since it can only be used to solve small/medium size instances due to its expensive computation cost.

Considering literature about Mixed-Integer Programming Models (MIP), a general LP model used in layout compaction and separation has been discussed in Li and Milenkovic (1995). It compares a formerly developed compaction algorithm based on a physical simulation approach to a position based optimization model. The model represents the forces that act upon the pieces as a linear objective function, allowing to compute the non-overlapping polygon positions at a local minimum of the objective function. The position-based model is also shown to be efficient for separation of overlapping polygons. Further development based on the work of Li and Milenkovic (1995) can be found in Gomes and Oliveira (2006) and Bennell and Dowsland (2001). Gomes and Oliveira (2006) present a global mathematical model for the Nesting problem, including simplifications that allow to solve two positioning subproblems of compaction and separation.

Using compaction and separation algorithms allows adjusting the placement of irregular pieces and obtain a locally optimal feasible solution. The compaction algorithm executes a continuous translation of the polygons in the layout, aiming to minimize the length of the container, while the separation algorithm makes a continuous translation of the polygons in order to make the layout feasible. Bennell and Dowsland (2001) use a bottom-left positioning heuristic and a compaction algorithm based on a LP model while Gomes and Oliveira (2006) use a hybrid approach with the bottom-left heuristic and the compaction and separation algorithm based on LP models assisted by simulated annealing algorithm, which is used to guide the search over the solution space.

Although MIP models cannot be used to solve piece placement position problems, simpler models such as compaction/separation models can be derived to solve some positioning sub-problems or to improve solutions. The compaction and separation models are derived from the complete MIP models from selecting, for each pair of pieces, which binary variables will be set to 0 and 1. Setting to 1 corresponds to fixing a relative position between a pair of pieces, and after this operation, the model can be considered a linear model, since the integer or binary variables are used as constants. The model can then be used to compact or separate the pieces on the layout, while maintaining the relative positions of the pieces among each other. Therefore, this method allows to simplify the MIP models, by fixing the relative positions of the pieces, thus becoming a LP model, which can then be solved efficiently.

The compaction mathematical model aims to reduce wasted space on the stock sheet by applying a set of continuous movements to the irregular shapes. The result is a local optimal cutting pattern. While using the set of movements on the irregular shapes, no overlaps among pieces are allowed, neither any change in their orientation. The separation mathematical model allows removing overlapping situations between irregular pieces, and the objective is repositioning the pieces that are not on feasible placement positions, into admissible positions. While using the set of continuous movements of the separation model, the number of non admissible positions cannot increase.

A reduction in the number of constraints was achieved by Bennell and Dowsland (2001) and Li and Milenkovic (1995), by placing bounds on the distance that any piece is allowed to move. The bounds enabled the pieces that are over twice this distance from each other to be discarded from the non-overlapping constraining set.

Fischetti and Luzzi (2009) introduced the concept of slices that consists in partitioning the space outside every NFP into convex disjoint regions that do not overlap each other. This approach is different from Gomes and Oliveira (2006) model in their use of NFP and the definition of variables. A possible way to define slices, according to the method proposed by Fischetti and Luzzi (2009), is shown in Fig. 2.20.

Alvarez-Valdes et al. (2013) propose a mixed-integer formulation for the Nesting problem, expand the work of Fischetti and Luzzi (2009) and present an exact Branch & Bound algorithm. They are based on the same approach, which divides the space around the NFP into disjoint regions. First, if the NFP polygon is non-convex, it is decomposed into convex polygons and a variable is associated to each one. The concavities of the NFP are then transformed, in a recursive

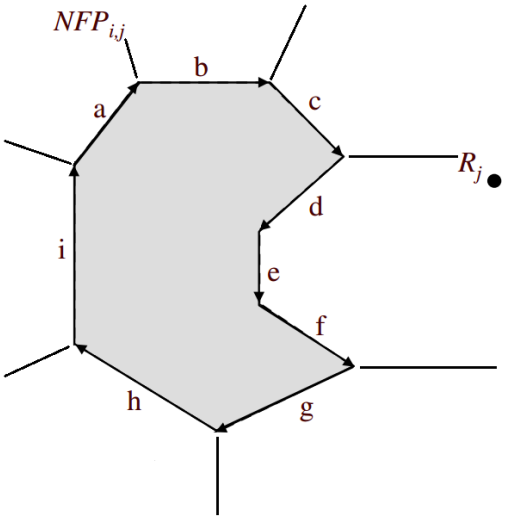


Figure 2.20: Spatial partitioning by slices

way, into a convex set, where the resulting NFP is convex. The final step is to define horizontal slices for the edges of the convex NFP, by drawing horizontal lines in the opposite direction to the NFP, stretching the whole length of the strip. Using variables associated to slices overcomes the disadvantages of the definition of Gomes and Oliveira (2006). Each feasible position of every pair of pieces corresponds to a unique variable (except for the unavoidable common border shared between slices) and defining horizontal slices (Fig. 2.21) enables an easier control of the relative vertical positions of the pieces.

Toledo et al. (2013) propose a MIP model where binary decision variables are associated to every piece type and to every point of a discretized board. The pieces can be convex or non-convex, and can exist multiple pieces of every type. The piece reference point is placed into feasible points on the board, considering the NFP and IFP of the other placed pieces and the board. When a piece of a certain type is placed at a point in the board, the value one is assigned to the variable associated with that point. A feasible layout (Fig. 2.22b) with the pieces placed on the positions defined into the Dotted-board (Fig. 2.22a) are presented in Fig. 2.22

This type of model formulation allows solving large instances of the Nesting problem to optimality, with the downside of having to increase the precision related to the discretization of the board. The optimal solutions of this model apply only to the cases where the Nesting problem is discretized with the same characteristics. When the Nesting problem is defined over continuous domains, these solutions may not be optimal solutions.

2.3.2 Models derived from Phi-Function

Using Phi-functions to address the positioning of the pieces, on a non-overlapping configuration, requires the use of Non-Linear equations. The structure of a Non-Linear Programming model used with Phi-functions ends up being a mix of the expressions used in the other models, since

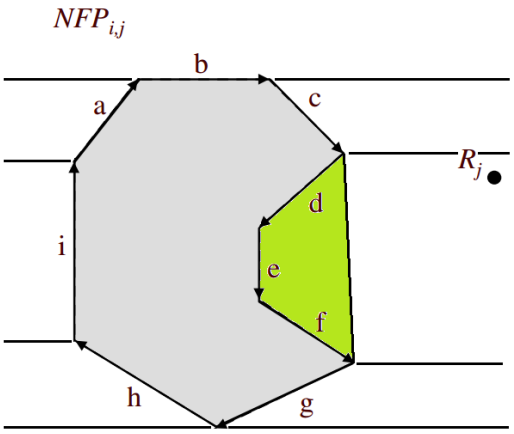
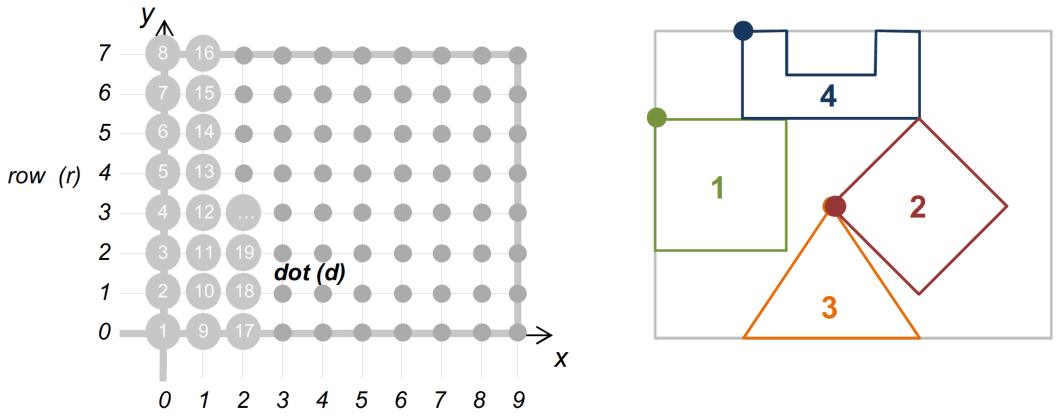


Figure 2.21: Horizontal slices for an NFP with closed concavities



(a) Dotted-board (b) Feasible layout

Figure 2.22: Feasible layout obtained from the placement of the pieces into the board (adapted from Toledo et al. (2013))

it compares pairs of pieces which can be composed simultaneously by lines, circles and arcs. For this reason, models derived from Phi-functions have non-overlapping conditions represented by linear and non-linear constraints. These mathematical expressions are used to express the relative positions between pieces, just like the NFP, but they have positive, zero or negative values, indicating the pieces are far apart, touching or overlapping each other, respectively. When the value of the Phi-function is zero, it is similar to the NFP. Since Phi-functions cannot be generated for every type of arbitrary shapes, approximations are done by decomposing the original shape into several pieces that belong to a specific set of shapes (Bennell and Oliveira, 2009).

Stoyan et al. (2012) present a mathematical model for the packing problem, with phi-functions derived from circles and non-convex polygons, supporting continuous translations and rotations, for placement inside a strip with prohibited areas. In it, he constructs the starting points through an optimization method by groups of variables, and then uses a local search methods to find an approximation to a global minimum.

2.3.3 Models derived from Circle Covering

Deriving a model from Circle Covering representation produces non-linear constraints which results in a non-linear programming model (NLP). Models derived from Circle Covering have non-overlapping constraints derived from the Euclidean distance between centers of a pair of circles, from different pieces, while taking into account their individual radius. Also, since they use trigonometric operations, any equation that also uses them will also be non-linear. The non-overlapping conditions derived from the piece placement positions inside the large object, relative the large object outline, are usually linear, but may also be non-linear if the frontier of the big object is defined by sets of circles, instead of edges. The biggest advantage of this type of model is that the non-linear non-overlapping constraints derived from circles are simple to compute, and they are independent of any piece orientation. Unfortunately, with the increase on the amount of circles that describe each piece, the non-overlapping constraints also increase, and the computational cost grows higher.

If the formulation of the model is derived from the simplest form of the problem, which is when all the pieces are represented by a single circle, the problem becomes significantly easier to address. The formulation of the problem becomes equivalent to circle packing. Birgin and Sobral (2008) addresses the problem of circle packing through a variety of non-linear models, either with a fixed set of identical or different-sized circles into with circular, triangular, squared, rectangular and also strip shaped containers. His formulation can be used in 2D as also 3D. Several strategies are described to reduce the complexity of non-overlapping constraints, such as spatial partition of the layout into identical squared regions, for overlapping evaluation. For circle packing inside ellipses, Birgin et al. (2013) explores multi-start strategies and non-linear programming models.

Wang et al. (2002) proposes a quasi-physical quasi-human algorithm for the unequal circle packing problem. The method is an analogy to the physical packing of cylinders inside a container where human behavior is mimicked, by removing cylinders to prevent being stuck on local minimum.

Stoyan et al. (2012) proposes a non-linear mathematical model for strip packing, supporting not only circles but also non-convex polygons. The Phi-function is used in conjunction with the non-linear model, and their behavior is analyzed. It is also discussed a possible approach to construct feasible starting points and the development of search methods that allow finding local minima and approximation to the global minimum.

Returning to piece representation through multiple-circles, Zhang and Zhang (2009) introduces a method for 2D packing optimization called Finite-Circle Method. This consists in representing every component of the problem through circles, and defining all non-overlapping constraints as constraints between pairs of circles. They discuss three algorithms that approximate pieces by circles and experiment with genetic and gradient based algorithms together with the Finite-Circle Method to solve the packing problem.

Imamichi and Nagamochi (2007) use an approach called Multi-Sphere Scheme that allows efficient design of algorithms that compute compact layouts. It consists in approximating every object by a set of spheres and then search for the placement position while minimizing a penalty function, which is based on the penetration depth of the spheres. They use a non-linear formulation together with a proposed Iterated Local Search algorithm to pack rigid objects, which applies the quasi-Newton method to the packing problem. Although this example is focused on spheres, it is also compatible with circle packing. In Imamichi et al. (2009), they proposed a separation algorithm based on a nonlinear program, and incorporated it, together with a piece swapping algorithm, into their Iterated Local Search algorithm.

The approach used by (Jones, 2013) focuses on solving to full optimality the Nesting problem using global optimization methods for certain types of quadratically constrained, quadratic programming problems. The method is based on inscribing several circles within each irregular shape and relaxing the constraint that the shapes do not overlap with the inscribed circles from another shape. It starts the compaction with a low number of circles per shape, and adds circles to the pieces where they are needed, to increase the quality of the covering, until the required accuracy is obtained.

2.4 Solution Approaches to the Nesting Problem

The methods used to solve Nesting problems can be defined into distinct categories, approaches that rely on specific solvers to solve mathematical models, approaches based on greedy constructive algorithms that build a solution from scratch and approaches that start from a feasible solution and try to improve it.

The approaches based on algorithms can be distinguished from mathematical model approaches by their aim, which focuses on achieving a good solution in a reduced computational time. Mathematical models, on the other hand, search for the optimal solution which can take too much time. The constructive heuristic approaches have the disadvantage of not being able to return a feasible solution at any time during the computation, since it only returns a solution when it finishes constructing it. In the next sections, an overview of these approaches will be discussed.

2.4.1 Mathematical Solvers

The approaches based on mathematical models require the definition of the problem in mathematical terms and its resolution with the assistance of a specific solver. Depending on the type of problem, the definition of the mathematical model will be different, due to the different types of variable and constraints. Variables can be binary, integer and real, while the constraints can be linear and non-linear constraints. The different combinations of variable and constraint types produce distinct mathematical models which require specific approaches to be efficiently solved.

The mathematical solvers are specifically designed to deal with certain types of mathematical models, and depending on the characteristics of the problem, they may be able to compute the global optimum solution. Linear Programming solvers can solve huge instances with hundreds of pieces, as long as the relative positions between the pieces are fixed, as shown in Gomes and Oliveira (2006). The Mixed-Integer Programing solvers can solve a reasonable number of pieces, 7 to 16 pieces for global optimum and considering continuous placement positions with discrete rotations, as seen in Fischetti and Luzzi (2009) and Alvarez-Valdes et al. (2013). When the placement positions and orientations are discretized, instances with a larger number of pieces can be solved, up to 56 pieces, as seen in Toledo et al. (2013).

Non-Linear Programming solvers can solve problems with more than 75 pieces, considering continuous translations and rotations (Stoyan et al., 2012), but returning local optimum solutions. Finding the global optimum solution using NLP models can only be achieved when using two or three pieces at most with continuous translations and rotations. NLP problems struggle to achieve optimal solution with four pieces (Jones, 2013).

State-of-the-art solvers oriented to solve LP and MIP mathematical models are CPLEX[1] and Gurobi[2]. The resolution of NLP mathematical models can be done using solvers that do not ensure global optimality, like Algencan[3], and IPOPT[4], solvers that ensure optimality, like LINDOGlobal[5], BARON[6] and GloMIQO[7].

The computational requirements for achieving the global optimum grow exponentially with the size of the problem to be solved. Since achieving a global optimum, or even an admissible solution, may not be possible within an acceptable computational time, alternatives based on constructive and improvement algorithms are useful.

2.4.2 Constructive Algorithms

One of the approaches to solve Nesting problems is building complete layouts from scratch, by selecting and placing pieces one-by-one. When using single-pass construction heuristics, some reasonable quality solutions can be found without much computational effort. Since feasibility

[1] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[2] http://www.gurobi.com/
[3] http://www.ime.usp.br/~egbirgin/tango/
[4] https://projects.coin-or.org/Ipopt
[5] http://www.lindo.com/
[6] http://archimedes.cheme.cmu.edu/?q=baron
[7] http://helios.princeton.edu/GloMIQO/

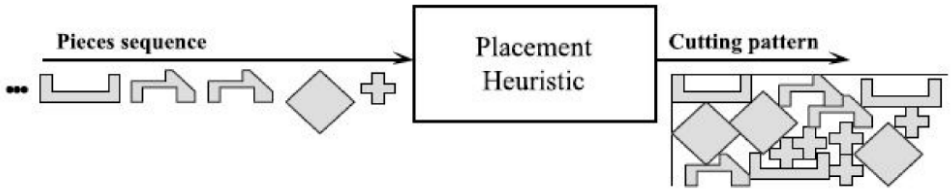


Figure 2.23: Cutting Pattern derived from Placement Heuristic used with a given sequence of pieces (adapted from (Gomes and Oliveira, 2002))

tests are included into these heuristics, each piece is placed on an admissible placement position on the stock sheet, and fixed to that position (Bennell and Oliveira, 2009). The quality of the solution derived by this type of heuristics depends mostly on the sequence of the placement order of the pieces, which can be in a random order, decreasing size, or other ordering criteria. The placement rule adopted to select the placement point also has a strong impact on the final result. More elaborated placement rules can have the ability to fill holes. Although the placement rule is critical to achieve a good final solution, the sequence in which the pieces are ordered also plays an important role. These two components determine how good is the approach to achieve a low waste, compact cutting pattern. An example of an application of a placement heuristic to a sequence of pieces can be seen in Fig. 2.23.

2.4.2.1 Placement Rules

Placement rules are useful to find an admissible piece placement position on the stock sheet. The most commonly used placement rule is the Bottom-Left. This rule works by moving each piece horizontally to the left until it cannot move in that direction, and then proceeds to move vertically until it is able to resume horizontal movement or touching another piece/bottom of the stock sheet. The piece is placed in its final position ensuring that it is fully contained in the stock sheet and that it does not overlap other pieces that have been already placed.

The geometric representation chosen for the pieces has great influence on the bottom left placement heuristic since if a grid representation is selected, the pieces must be moved in steps, and in each step the placement feasibility must be checked. If overlap is detected, the piece is returned to the previous position, and the movement is directed towards the bottom. If this bottom movement is feasible, then it proceeds until no movement is allowed in this direction, and returns to left movement direction. The final piece placement position is found when no more movement to the left or bottom is allowed (Bennell and Oliveira, 2009), as is shown on Fig. 2.24.

An important issue in placement heuristics is the ability to fill holes that the pieces have, or that may be created between previously placed pieces. Authors as Burke et al. (2006), Dowsland et al. (1998) and Sagenreich and Braga (1986) suggest searching the layout from the left side through the infeasible positions for holes, and therefore admissible placement positions in internal holes are searched before looking for a position on the right side of the stock sheet. One way to find

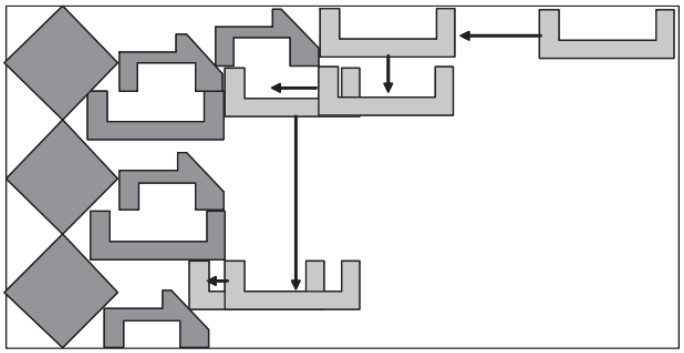


Figure 2.24: Bottom left approach example (adapted from (Bennell and Oliveira, 2009))

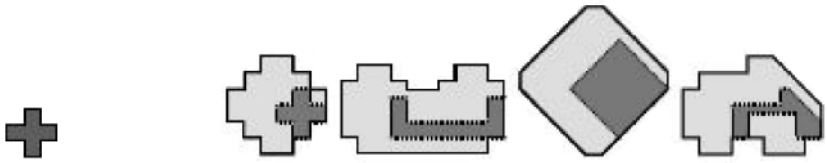
feasible positions is to build a grid over the stock sheet in order to define discrete positions on the layout for testing. The grid can be used to represent both the layout and the approximated pieces, using a Bottom Left approach, like in Sagenreich and Braga (1986), or reducing the stock sheet to a grid while alternating the search between top and bottom moving to the middle and having started from the left side (Dowsland et al., 1998).

Some other more advanced strategies, like one from Burke et al. (2006), use discrete movement in horizontal direction for the pieces while maintaining a continuous vertical movement which is achieved by geometric functions that check for admissible piece placement positions and compute the minimum vertical distance needed to an admissible piece placement position. This strategy uses the NFP to check for overlap in a given placement position avoiding the necessity to use a grid. With this strategy, all the feasible placement positions can be identified, as long as they do not belong to the inside of the NFP of the positioned pieces, and still maintain admissible positions inside the IFP of the layout. One example of the feasible placement positions, can be seen in Fig. 2.25d, using NFP examples and its degenerated cases, including the IFP, as seen on Fig. 2.25.

When having a partial solution composed of a set of previously placed pieces in fixed locations, it is possible to compute the set of NFP between each of those placed pieces and the piece to be placed next. This returns a feasible placement region that may be infinite in size, depending on the stock sheet parameters, with infinite placement positions.

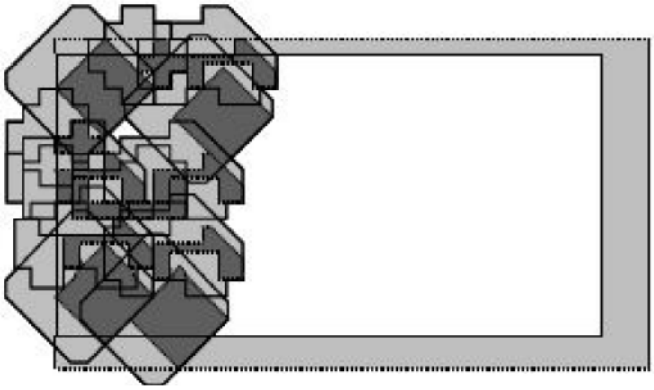
Gomes and Oliveira (2002) managed to reduce the number of possible placement points by considering only intersection points between NFP's and between NFP's and IFP's. These placement points are a subset of the vertices present in all NFP, and are created in the intersection of all NFP edges and also in the intersection of all NFP and IFP. If one of these vertices is inside another NFP, then it is removed since it is not a feasible placement position. With a discrete and finite set of vertices representing admissible placement positions, finding the Bottom Left placement position for a given piece is a simple task.

Another alternative to the Bottom Left placement rule, is suggested by Oliveira et al. (2000). This approach consists in using a floating origin while fixing the relative positions of the pieces to each other, and defining the width of the partial solution to be equal or smaller to the stock sheet



(a) Next piece to be placed (b) NFP's according to other pieces on the layout

(c) IFP between piece and container



(d) Feasible placement positions, marked as white color

Figure 2.25: NFP examples and degenerate cases (adapted from (Gomes and Oliveira, 2002))

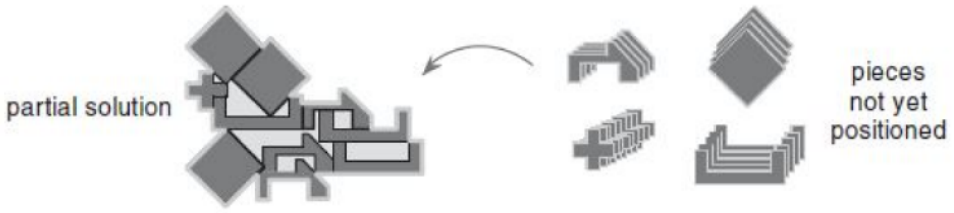


Figure 2.26: TOPOS partial solution and pieces not yet positioned (adapted from (Oliveira et al., 2000))

width. An illustration can be seen in Fig. 2.26.

The partial solution is described by the frontier defined by all the pieces, including the last additional piece, and the space between the pieces is considered waste, discarding any possibility of positioning a new piece in that region. In Oliveira et al. (2000), the NFP's are used to find feasible placement positions, and for each new additional piece, a new NFP needs to be computed between that piece and the partial solution. Several rules were created to maintain the layout as compact as possible. Their aim is to minimize the area or length of the rectangular enclosure of the newly created partial solution, or maximize the overlap between the rectangular enclosure of the actual partial solution, and the rectangular enclosure of the piece to be place, without any overlap among pieces. An advantage to this alternative approach is that it can be used with irregularly-shaped stock sheet.

2.4.2.2 Placement Sequences

The positioning sequence in which the pieces are ordered and placed on the stock sheet contributes greatly to the solution of the final layout. The selection of pieces can be done using several criteria, such as Random Selection, using a Monte Carlo based algorithm, which generates a great variety of placement sequences. This approach is mostly used on methods that improve an already complete solution, and also when multiple initial solutions are required. When a constructive algorithm is being used, this approach is discarded in favor of another that uses a more intelligent strategy other than random generation. Dynamic selection assigns pieces into groups defined by types (identical shapes), with a given number of discrete orientations, and allows the placement of any pieces as long as pieces from a given type have not yet been placed. Several criteria, which were evaluated by Oliveira et al. (2000), determine which type of piece is the next to be placed, such as relative waste, overlap, relative distance, waste minus overlap and relative waste plus relative distance. In this method, all the piece types are positioned, and the best solution is selected, repeating the same process (inserting all the remaining piece types with the previous best pieces placed) iteratively until all pieces are positioned. An example can be seen on Fig. 2.27.

Dynamic piece selection can be extended, by constructing a search tree, in which every node in the search tree corresponds to a partial solution, and a branch from a node represents a piece that was added to the partial solution. The leaf nodes represent all of the complete solutions. Since

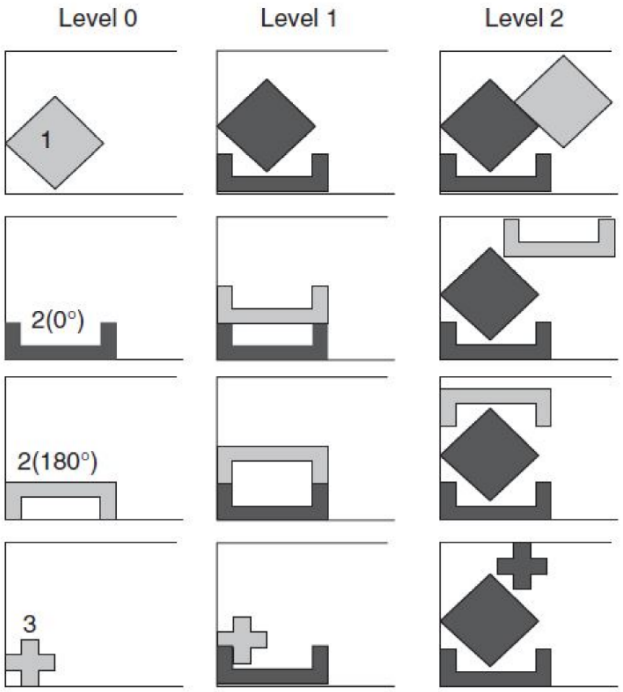


Figure 2.27: TOPOS piece selection and positioning method (adapted from (Oliveira et al., 2000))

this is a greedy search, the path selected while navigating the tree may not lead to the best solution, preventing wasting time building the entire tree while restricting the number of successor nodes at each level. Several possibilities to improve this approach exist, as in Albano and Sapuppo (1980), such as using backtracking within a given number of levels and pruning the tree using a waste criteria to evaluate each partial solution. One path is pursued only if waste limit is slightly higher than others from partial solutions further up the tree. Another possibility is beam search (Bennell and Song, 2008) where the tree is pruned accordingly to two evaluation functions, one evaluates the partial solution quality and the other makes an estimation of the final solution considering all of the remaining pieces, packed on a greedy way.

2.4.3 Improvement Algorithms

Some approaches to Nesting problems start with a complete feasible solution, and proceed to iteratively make small changes in order to achieve better results. This technique is usually known as local search. Due to the combinatorial component of nesting problems, local search is one of the many possible approaches, but some others have been implemented like Tabu Search (Bennell and Dowsland, 1999), (Burke et al., 2006), Simulated Annealing (Heckman and Lengauer, 1995), (Oliveira and Ferreira, 1993), and Genetic Algorithms (Babu and Babu, 2001), (Jakobs, 1996). Layout improvement approaches to Nesting problems can be divided into two types, by either searching over a sequence of pieces and relying on a low-level placement rule to build

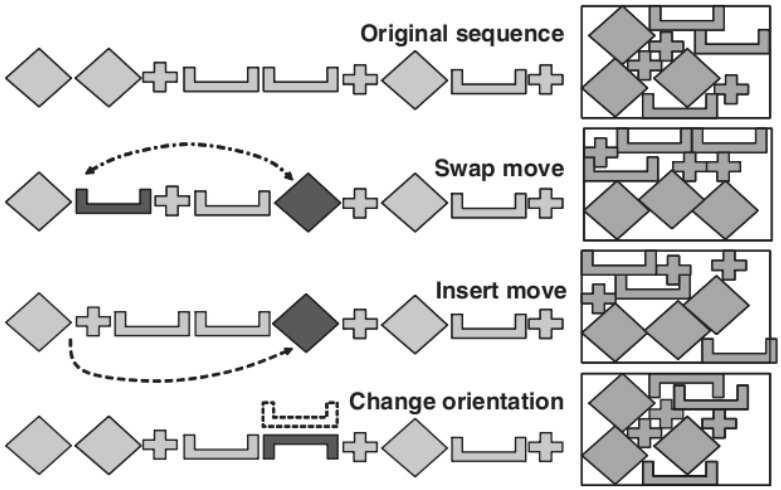


Figure 2.28: Effect of Bottom Left placement rule with selected movements (adapted from (Bennell and Oliveira, 2009))

the actual layout, as seen in the previous section using partial solutions, or by working with the solution directly and moving the pieces around in the physical layout.

Both approaches have advantages but also limitations. When working with sequences of pieces, the sequence and its solution needs to be analyzed in order to select a different sequence with better solution. This approach is more computationally expensive than searching for a piece move within the layout that improves the solution. The quality of the solution also depends on the placement rule chosen, but as an advantage this approach ensures that the solutions are feasible. The same cannot be said about the other approach, since moving pieces within the layout might create overlaps among them, while searching for better positions, which in some cases difficult achieving a feasible solution.

2.4.3.1 Sequence Shuffling

Sequencing problems are commonly found in optimization applications. A placement rule produces a solution (i.e., a layout) from a given sequence of pieces but its performance depends on the sequence of the pieces. Some commonly used movements in approaches as local search are swap movements which change the position of two pieces in the sequence, and insert movements which inserts a piece into a given position into the sequence. When dealing with packing problems, another movement can be included, namely the change of orientation of a piece. These movements can be easily interpreted in the Fig. 2.28.

The original sequence, on the top, is modified considering each of the lower movements. Its result is the layout on the right of each sequence. From what can be observed, the swap movement has a higher chance of to generate smaller changes to the global layout than the other movements, since replacing one piece by another leaves all the other pieces in the sequence in exactly the same position. An insert move changes many more pieces out of their original position in the

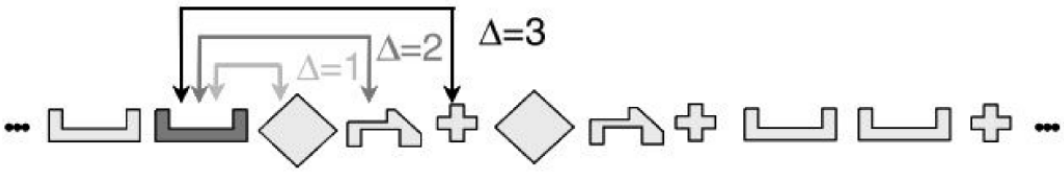


Figure 2.29: Piece swap limited by the maximum distance parameter (adapted from (Gomes and Oliveira, 2002))

sequence, since pieces will move after the position where the piece is inserted. Comparatively, the movement to change orientation cannot be always used, since in some cases orientation changes are not allowed, but can cause great modifications to the layout. It is usually used with a swap or insert move.

Due to the number of pieces in a real world Nesting problem it is necessary to restrict the size of the neighborhood. One approach is used by Gomes and Oliveira (2002), where a parameter limits the maximum distance a swap can occur among pieces, using a probabilistic two-exchange heuristic. An example is presented on Fig. 2.29.

The layouts generated by the swaps are evaluated, making a decision based on criteria that chooses layouts that improve the current solution (first-better), selects the best layout among all (best), and a random solution, among all the solutions better than the current one. Burke et al. (2006) uses tabu search to find local optima, and restrict their neighborhood size to five solutions selecting each one of them randomly, and applying a random move between insert, pairwise swap, three-way swap and n-way swap. The pieces for the swap move or the position for the insert move are randomly selected. Takahara et al. (2003) propose a neighborhood search based exclusively on insert moves, with a normal and adaptive state, in which pieces have the same move probability or their move probability depend on their relative weight, respectively. Whenever a movement improves the solution the weight of the piece increases by one unit. Jakobs (1996) uses genetic algorithms to search piece sequences. The offspring are generated through random selection of a number of consecutive pieces, starting from a random number and being placed at the beginning of the offspring, and receiving the unmodified sequence part from the second parent. Mutations are introduced by a rotation of pieces in 90 degrees. Also with genetic algorithms, Babu and Babu (2001) deals with multiple heterogeneous stock sheets using genetic algorithms. Their genetic codes have included the stock sheet sequence, orientation of the pieces and the sequence of the pieces. It uses three mutation operators, one being the swap movement, and the other two based on the piece orientation change movement. Dowsland et al. (1998) present the Jostle algorithm which uses an analogy to the oscillation of a physical layout to shuffle the pieces from the left end to the right end of the stock sheet. This switching among pieces makes the last pieces to be placed in the last iteration to be the first to be moved and placed into the next iteration. The placement rule used in this approach has a hole-filling capability which allows pieces that appear in the final positions of the sequence to slide to positions between larger pieces. A representation of the Jostle

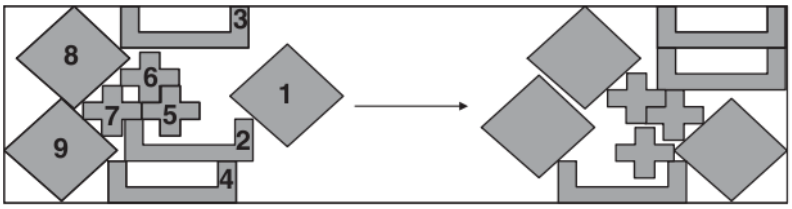


Figure 2.30: Movement from the left end to the right end with the Jostle algorithm (adapted from (Dowsland et al., 1998))

algorithms can be seen in Fig. 2.30.

2.4.3.2 Layout Shuffling

A critical aspect of working with complete solutions is ensuring that all operations end with a feasible solution. However, infeasibility is used when searching for a better solution by allowing overlap among pieces, which is penalized in the objective function. Allowing infeasible solutions usually leads to more fully connected landscapes, increasing the efficiency of the search algorithms. However they may have problems when converging to good local solutions with no overlap. This may prevent the search algorithm of returning a feasible solution. Eliminating the overlap and later adjusting of the pieces will result in a degradation of the solution quality.

Since the stock sheet is a continuous area in which exist an infinite number of admissible placement positions, the search algorithms have to work with a continuous solution space. The discretization of this solution space allows the search algorithms to compute faster at the expense of accuracy, and eliminating some placement points while keeping the most promising among them. This approach tries to ensure that the global optimal solution is still available as a placement position. The problem with this approach is the difficulty in evaluating the physical layout in order to find a movement. Usually this is tackled in two different ways, by moving a single piece, or by moving a limited set of pieces. A single piece movement can be considered an insert movement because that piece is removed from its current location and inserted into a new one. Considering multiple piece movement, can be considered a swap movement since piece positions are exchanged among them. Some other movements can be applied, like rotation or symmetry transformation. Some strategies based on local search approaches reduce the solution space to a discrete set of points, by means of imposing a grid over the stock sheet, and using only those points as placement positions. The greatest advantages besides reducing the number of admissible piece placement positions are its ease of implementation, easy to manage the dimension of the grid, and the search positions are evenly distributed over the stock sheet.

A simulated annealing algorithm was presented by Oliveira and Ferreira (1993) that places the pieces randomly on the stock sheet and executes a neighborhood movement upon one piece, moving it one grid unit each time. Other similar approach using grids is presented by (Bennell and Dowsland, 2001) in which their neighborhood keeps track of all the generated solutions through piece movement into a new grid positioning. Taking into account the fact that in an optimal

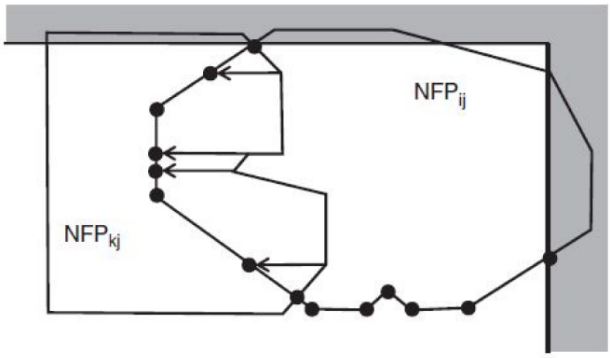


Figure 2.31: Finding the point with minimum overlap (adapted from (Bennell and Oliveira, 2009))

solution all the pieces will touch the stock sheet edges or some other piece, only those edges need to be searched for admissible placement positions. When using the NFP, that task is simplified. This is used by Bennell and Dowsland (2001) and improved by finding a subset of points where the point of minimum overlap is contained. The point of minimum overlap between the current piece and the current layout is contained in the set of vertices of all the NFP's and vertices created by the intersections between NFP's and IFP's. An example of a set of points with a point of minimum overlap is presented in Fig. 2.31.

In Błażewicz et al. (1993), a tabu search algorithm is presented that converts the infinite piece placement positions into a finite and discrete set. It uses a Bottom Left placement rule ordered by non-increasing area, meaning that a piece is moved from its current position to a hole within the layout, or to the end of the layout. This approach has the disadvantage of having to be aware of the position of holes, which is a difficult and computationally expensive task. As an alternative to the discretization of the layout space, and extending their previous work in Oliveira and Ferreira (1993), they imposed a limit to the maximum distance that a piece can move and select a randomly generated direction and distance. Another approach is presented by Gomes and Oliveira (2006) which uses a hybrid algorithm, using a simulated annealing meta-heuristic with linear programming models. The simulated annealing is used to guide the search over the solution space while the linear programming models come from compaction and separation algorithms. The main differences to other similar approaches are the type of movements used to move between solutions (swap movement) and the criteria used to accept or discard feasible and unfeasible solutions. When a movement results into an unfeasible layout, the layout is not accepted.

When moving the pieces around, allowing overlapping but penalizing it creates an additional objective when the primary objective is minimization of the length of the layout. Some approaches can be taken. Defining weights for each cost element is the first one. Heckman and Lengauer (1995) classify the total layout cost as being a weighted sum composed of the total layout length, the sum of the overlap among each pair of pieces, and one additional cost related to the overlap across horizontal and vertical dimensions. The balancing of the weight trade-off is done dynamically. In Bennell and Dowsland (2001) the overlap is considered as a horizontal translation, and

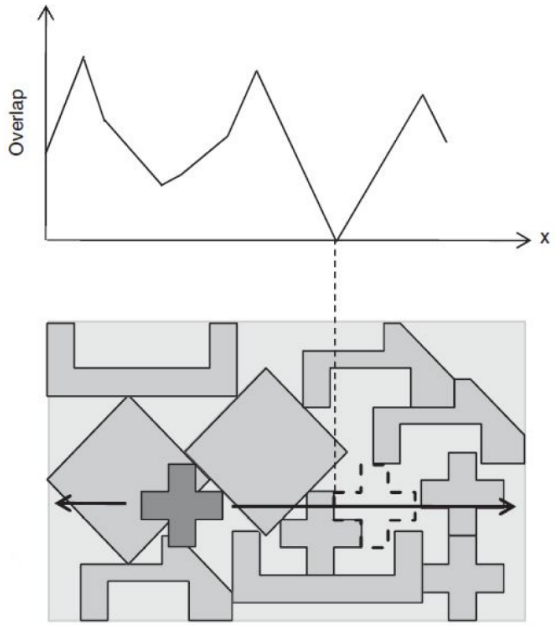


Figure 2.32: Overlap function of movement in horizontal direction (adapted from (Bennell and Oliveira, 2009))

receives the direct sum of overlaps and weights without attributing any weights. They balance the trade-off between overlap and weight using compaction and separation.

Another approach is to focus on the optimization of one objective, while fixing the other, and then switch among them. Bennell and Dowsland (1999), Imamichi et al. (2009) and Egeblad et al. (2007) focus on the minimization of the overlap while fixing the stock sheet length. After achieving an admissible layout, the stock sheet length is reduced. In Egeblad et al. (2007) they chose to minimize the overlap and only search for admissible placement positions in a horizontal or vertical direction, thus simplifying the search by excluding one dimension and evaluating all the positions along the other dimension. The overlap is seen as a function and its minimum is selected as the position for the piece. We can see the overlap function in Fig. 2.32.

2.5 Specific Geometric Algorithms for the Nesting Problem

Building a layout from scratch or improving an existing one is a complex task, requiring appropriate methods to assist in the search for the best layout possible. These methods cannot be implemented without depending on another set of complex operations, such as the detection of overlapping among pieces. Since it is desired to avoid complex and repetitive trigonometric computations when dealing with overlap detections and computation of the relative position of the pieces, several useful concepts, such as D-functions, No-Fit-Polygon (NFP) and phi-functions, are

commonly used. Since most geometric representations cannot, by themselves, provide enough capabilities to deal with the piece placement problem, other tools are required to assist them.

These tools are available from software libraries, free or proprietary software, or they might require being developed, adapted and implemented by the user. A library that includes many functions and algorithms specifically tailored to geometrical problems is CGAL. CGAL is a software library developed by several research groups around the world. It provides a flexible implementation of computational geometry algorithms and data structures to be used in industry and also in academia. It consists of a core with several geometric primitives and data types, a large collection of basic algorithms and specific data structures (for triangulations, planar maps, among others) and support libraries for I/O, debugging and visualization. Considering the discussed topics in this work, CGAL contains algorithms related to NFP, convex decomposition, and many others. When used for commercial applications it requires a license.

Besides the support for the geometrical component, the combinatorial component also requires a specific set of tools that are not directly usable. Mathematical models, Heuristics and Meta-Heuristics, among others, are used to achieve feasible solutions. These models and algorithms are derived or depend directly on the chosen geometrical representation and can be improved by using specific algorithms such as the ones discussed in this section.

In this section, commonly used mathematical tools and algorithms are described.

2.5.1 Spatial Partition Algorithms

Spatial partition involves dividing a region of space into several regions, often represented by a hierarchical structure, where the partitioning algorithm is recursively applied to each sub-region created. If a region is divided into equal discrete blocks, and the information about the contents of that region is stored into a structure, the information about the placement of objects and empty area will become redundant, since both the interior and exterior of the objects (either filled or empty regions) do not contain information about the frontier of any object. This redundant amount of information increases the computational cost of operations such as overlap detection, and limits the maximum spatial detail (or resolution) due to memory consumption, which increases as the number of discrete blocks grows.

Several partition methods can be used to efficiently reduce the amount of redundant information. In Binary Space Partition (BSP) each region is recursively divided into two sub-regions, so each node will have two children. Quad-trees divide every region into four sub-regions, so each node of the tree has four children. An example of a Quad-tree can be seen on Fig. 2.33.

These spatial partition methods also have disadvantages. When storing information in the tree about lines and curves, depending on the maximum depth allowed (due to the desired resolution detail), the tree may become very unbalanced (with some leaf nodes of the tree very close to the root node, and other leaf nodes very far), which may negate the benefits of a search tree, since traversing it adds computational cost depending on the number of nodes.

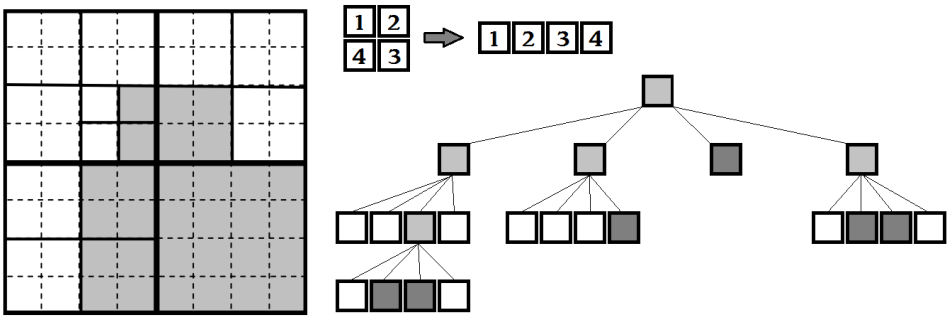


Figure 2.33: Quad-tree (adapted from (Moore, 2002))

2.5.2 Collision Handling

In literature, one can find many problems where there is the need to deal with the collision between objects. The usual way of handling collisions include dealing with how the detection of collisions is achieved, how to model the contact and how to respond to the collision. Collision handling is usually used within simulations, where the progress of the motions of all objects is affected by a temporal interval called simulation time. For each time period, the motions of all objects are updated, and the collisions between them are computed while placed into their new positions. The changes in the velocities and direction of movement of all objects are also updated.

2.5.2.1 Common weaknesses in basic collision detection algorithms

Cases that involve interactive simulations or animations require high consistent frame rate, which can be difficult to attain due to some difficulties that arise. As discussed in Hubbard (1995), the basic detection algorithm has three weaknesses:

Fixed time step weakness – By dividing the simulation time into fixed time steps of a given period, if the period is too small, the accuracy of the collision detection will improve, at the expense of a higher computational cost, and lower frame rate. If the algorithm uses a large period for the time step, the algorithm becomes more efficient but on some occasions the collisions of objects are not detected. This happens due to the non-overlapping of initial and final position of both objects, although they overlap somewhere in the middle of their movement, and that overlap is not detected. This can be seen in Fig. 2.34.

All-pairs weakness – In some simulations, besides collision detection there are some tasks computed in real-time, such as frame renderization. If the number of objects to compare for overlap is high (at the worst case the collision detection between N^2 pairs objects need to be compared) the computation time will increase quadratically and the frame rate will drop substantially.

Pair-processing – A pair-processing algorithm determines if the surfaces of two objects intersect in a particular moment in simulation time. The efficiency and accuracy of these algorithms

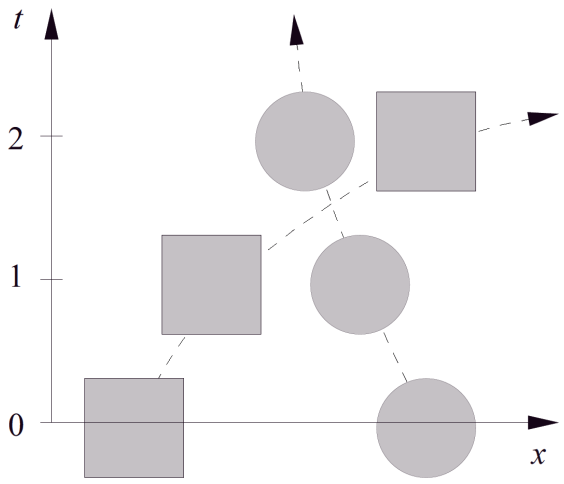


Figure 2.34: Undetected overlap situation due to large time step (adapted from Hubbard (1995))

are heavily dependent on the geometrical representation used, which may have a high computational cost, and added complexity when special intersection cases arise, that are difficult to process with efficiency.

The fixed time step problem can be addressed using an adaptive time step period that changes depending on the situation, being smaller when collisions are more likely to happen, and larger when they are less likely. Another possible approach requires extending the problem into four dimensions (the new dimension representing time) as discussed in Cameron (1990). This expansion creates a geometrical shape that can be tested for intersection against another. Cameron (1990) also shows a visual example how this method works with a two-dimensional problem, expanded to the third dimension (time) and the intersection of the resulting polygons. Another possibility is to assign each object its own simulation time, where its state only gets altered when a collision is found ((Mirtich, 2000)). The main difference between this method and a retroactive method is that this method only steps back the simulation time of the colliding objects, instead of all objects. Discussion about further approaches for these fixed time step weaknesses can be found in Bradshaw (2002).

Solving the problems related to all-pair and pair-processing is usually done through a multi-phase algorithm (with a Broad, Narrow and Exact phase) that consists in a few steps:

Broad Phase: Discard overlap comparisons between distant pieces – The Broad phase consists in the first phase of the collision detection, where most of the overlapping possibilities between objects far apart are discarded. An accurate selection of the objects that do not overlap has great impact on the performance of the algorithm, since the number of potential collision detection tests increase quadratically with the number of objects. The more collision detection operations are discarded, the more efficient the algorithm will be. This has the effect of minimizing the effects of all-pairs weakness. Approaches that deal with his problem can be found on Cameron (1990) where he uses a fourth dimension related to time

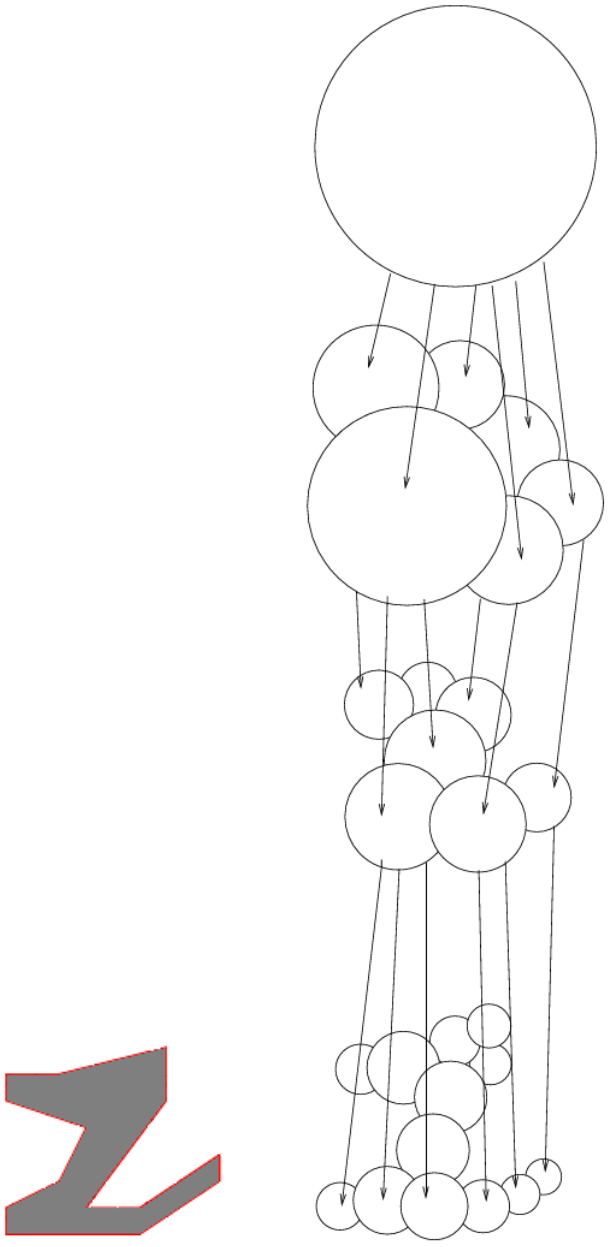
to predict collisions, Palmer and Grimsdale (1995) uses bounding spheres to represent objects and computes collisions using sphere-trees and exact polygon intersection tests, among others.

Narrow Phase: Discard overlaps through Bounding Volume Hierarchy(BVH) tree – The Narrow phase makes use of spatial localization techniques to reduce the complexity of the objects that need to be considered for collision detection. Bounding Volume Hierarchies are very useful to simplify collision detections, since they use a hierarchy with simplified objects on one end and on the other the accurate representation of the object. This allows to use simple representations of the object, such as sphere-trees (Fig. 2.35), Axis-Aligned Bounding Boxes (AABB) (Fig. 2.36a) and Oriented Bounding Boxes (OBB) (Fig. 2.36b), among others. In order to support a consistently high frame rate Hubbard (1995) proposed an algorithm that approximates contact points as fast as possible. The algorithm is capable of dynamic accuracy adjustment allowing to reduce accuracy of the algorithm to maintain frame rate.

Exact Phase: Perform accurate collision detection between objects – The Exact phase uses the results previously obtained in the Broad and Narrow phases, and performs accurate collision detection between objects using their highest level of detail available, taking into account the used geometrical representation. One approach used in the Exact phase is to use BVH that contain triangles in their leaf nodes, as mentioned in Bradshaw (2002), where in case of possible overlap, the triangles within the leaf nodes are used to test it. Other algorithms for collision detection in the Exact phase are mentioned and discussed in Bradshaw (2002), referring to line clipping that tests for edge-faces intersection between convex polyhedra, and also closest point algorithms.

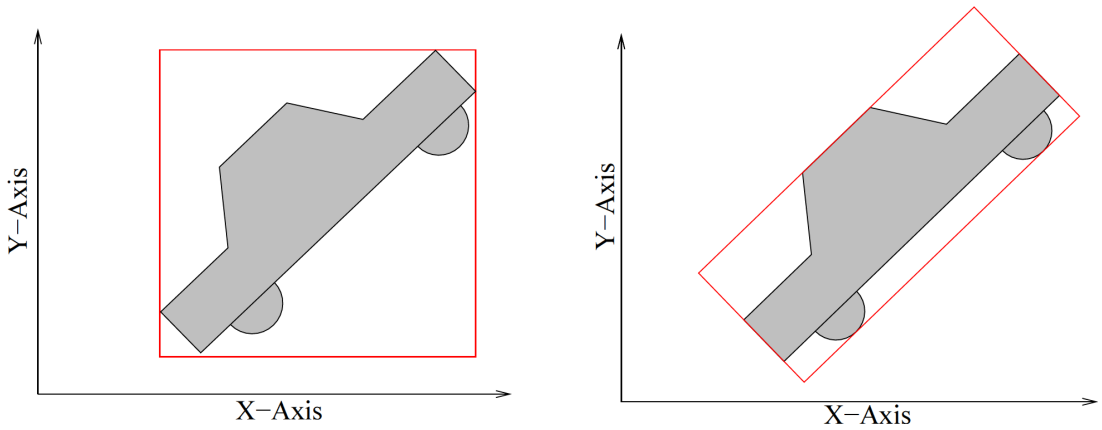
2.5.2.2 Interruptible collision detection

Real-time simulations may require additional tasks to be executed in parallel, such as rendering frames. For achieving consistent high frame rate, without dropping below a certain level, Hubbard (1995) proposed an algorithm that approximates contact points in the objects as fast as possible, using a sphere-tree. The algorithm is capable of dynamic accuracy adjustment allowing to reduce accuracy to maintain frame rate. This is achieved through a space-time approximation that tries to predict the future position of the objects, focusing only on impending collisions, and making sure that no collisions will be missed. The algorithm maintains a list of objects that are potentially colliding, and as the tests return positive, they generate new nodes (children) that require further collision testing. By testing one hierarchical level at a time, and progressing deeper into the collision tree, the algorithm verifies a tighter approximation of the object and increases the accuracy of the collision information. As it builds the collision tree, it tests the overlaps at each successive hierarchical level, until the time available has been expended. When no more time is available, the algorithm stops the search and continues from the last node on the next iteration. This allows



(a) Original polygon (b) Sphere-tree

Figure 2.35: Bounding Volume Hierarchy of Sphere-Tree (adapted from (Broutta et al., 2009))



(a) Axis Aligned Bounding Box (b) Oriented Bounding Box

Figure 2.36: Axis-Aligned and Oriented Bounding Boxes (adapted from (Bradshaw, 2002))

refining the search for collisions at each time step, but using only the incomplete information from the collision detection algorithm. This approach allows interrupting the collision detection algorithm, maintaining the currently state of information, and continuing the search for collisions on the next iteration.

2.5.2.3 Traversal Algorithm for Bounding Volume Hierarchies

Bounding Volume Hierarchies require a proper method to navigate through the structure, in the most efficient way (i.e. being able to confirm collisions with the minimum amount of comparisons between objects). While BVH can be composed by multiple different object representations, such as bounding boxes, enclosing circles, convex polygons, among others, in order explain how the traversal algorithms works, an example with sphere-tree's will be used.

Having a representation of an object by an hierarchy of spheres (or circles in 2D), or boxes(3D or 2D), every time its not possible to discard the possibility of overlap between objects, the search for collisions continues on the next (more detailed) hierarchical level of the tree. When objects do not overlap at the current level, then the objects do not overlap, and further search of collisions between the current objects can be ignored. Two possibilities exist for making a traversal into the BVH trees: one of them progresses to the next hierarchical level on both object trees, and in the other the search only advances one of both object trees to the next hierarchical level. The second option (proposed by Palmer and Grimsdale (1995)) provides a reduced amount of comparisons that result from positive overlaps, when compared to the first option.

The traversal of the BVH tree of two possible colliding objects can be seen in Fig. 2.37, for a two-dimensional example. The algorithms starts at both objects root nodes, where the overlap test returns positive. The nodes from hierarchical level 1 of object B are tested against the root node of object A. All nodes from object B, from the current hierarchical level, that do not return a positive overlap are discarded. The following step advances one hierarchical level of object A and tests its nodes against currently selected nodes in B. Negative nodes are discarded, and the process repeats

in the same way, by advancing the hierarchical level of object B, and testing against the currently selected nodes from object A. This is repeated until the final hierarchical level is reached, and collision can be definitely confirmed.

This traversal method has also been adapted to support collision detection with interruptions, as can be seen in O'Sullivan et al. (1999). The main change is that all leaf nodes in the tree are introduced into a list with resolved collision tests, and when the algorithm needs to be interrupted, the pending collision tests get the current nodes introduced into a queue. The current results from the collisions are returned even if they had not completed the tests up to the leaf of each tree. The collision tests are then resumed while taking into account the nodes already processed and placed into the queue, with a selection algorithm based on human perception that differentiates between high and low priority collisions.

2.5.2.4 Collision detection adapted to Nesting problems

In Nesting, objects are normally represented by polygons, where the usual method to compute overlap is through D-functions. The BVH consists, at the root node, of an AABB that contains the whole polygon. The next hierarchical level of the BVH tree contains the AABB of the edges of the same polygon. On the last hierarchical level, the edges and vertices of the polygon are used directly to compute for overlaps. These simple tests can be used to avoid comparing all the primitive components (such as points, lines, arcs, etc) between the pair of pieces, which is even more important in Nesting problems. This hierarchy of overlap exclusion tests can be found in Bennell and Oliveira (2008). A simple test as the one proposed by Preparata and Shamos (1985) can be used to verify if one polygon is included in another. Comparatively to other problems where Collision Detection methods are required, in Nesting problems there is a higher difficulty in discarding possible collisions between objects since the aim is to get objects into a tightly packed configuration.

2.5.3 Convex Decomposition

Convex decomposition is used as a simplification method for complex geometry since most geometric problems can be made simpler and faster if the objects are convex, instead of non-convex. Convex polygons are much easier to work with than the original objects. This method can be employed in a preprocessing stage, and the resulting convex polygons are used in real-time. The simplest form of convex decomposition is triangulation, where the interior of the polygon is completely decomposed into triangles, however, using triangulation creates too many derived polygons which makes a problem more computationally expensive. There are two conflicting objectives when considering literature in convex decomposition, one is time complexity and the other is the number of convex polygons generated. The computation of the optimal decomposition is considerably slower than a suboptimal solution.

An optimal decomposition algorithm can be found in Greene (1983) which runs in $O(N^4)$ time, being N the number of vertices of the original polygon.

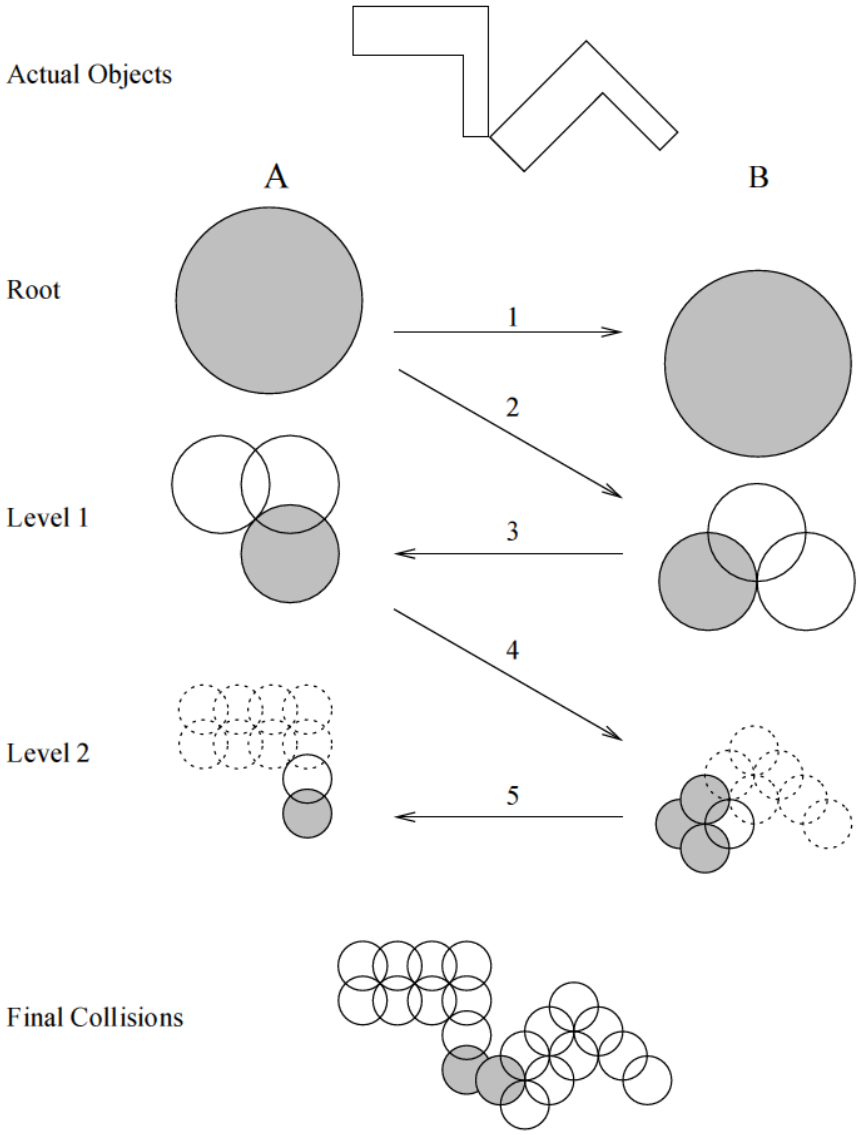
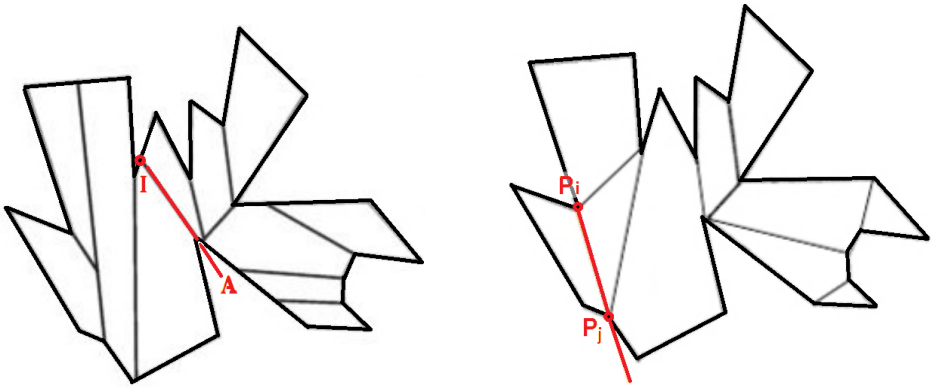


Figure 2.37: Traversal of Bounding Volume Hierarchy Tree (adapted from Palmer and Grimsdale (1995))



(a) Example of Angle Bisector Decomposition, with intersection point I (adapted from (Chazelle and Dobkin, 1985))

(b) Example of Small Side Angle Bisector Decomposition, with concave vertices P_i and P_j (adapted from (Agarwal et al., 2002))

Figure 2.38: Angle Bissector Decomposition

Hertel and Mehlhorn (1983) has an algorithm that decomposes a polygon no more than four times the minimum amount of convex polygons. It starts with an arbitrary triangulation of the polygon and removes non-essential diagonals iteratively, until none are left. It runs in O(N) time.

Chazelle and Dobkin (1985) proposed an algorithm called angle bisector decomposition that works by initially extending the angle bisectors of the concave vertices of the polygon until it intersects the outline or another angle bisector. This procedure repeats itself until concave vertices cease to exist. An example of this algorithm can be seen in Fig. 2.38a, where the angle bisector A is extended and intersects point I.

Agarwal et al. (2002) proposes an algorithm called small side angle bisector decomposition. It is an improvement over the algorithm proposed by Chazelle and Dobkin (1985) that analyses all pairs of concave vertices of the initial polygon, and traces a dividing line between a pair of those vertices that has the minimum number of concave vertices between them, thus eliminating the vertices. When the connection between two concave vertices is not possible, the process eliminates only one vertex by tracing a single line closest to the angle bisector of that vertex. This decomposition has a low computational cost and returns good results when compared to others in the literature. The time complexity of this algorithm is O(N^2). An example of this algorithm can be seen in Fig. 2.38b, where the concave vertices (P_i and P_j) are connected and eliminated simultaneously.

2.5.4 No-Fit-Polygon construction

The NFP is used to compute overlaps between two polygons. It is a very efficient method to compare overlaps between two polygons, since it transforms the comparison between two pieces into a comparison between a piece and a point. However, computing the NFP in real-time has a great computational cost, making the algorithms that compute it very slow. For this reason, the computation of the NFP is usually done in a pre-processing phase, but only for discrete orientations, and

not with free-rotations (while being possible to do so).

A similar concept to the NFP is the Minkowski sum. The Minkowski sum can be described as the sum of all the pairs of points of two different sets, both sets representing, polygons. It is defined as seen in Eq. 2.2 and is equivalent to the NFP if polygon B is transposed into its symmetrical B', as seen in Eq. 2.3, while B' also maintaining the same orientation as A. This relation was first formalized by Stoyan and Ponomarenko (1977). A procedure that generates the NFP, attending to this relation, is presented in Ghosh (1991). The procedure is based on a set of theorems for convex and non-convex cases, which represents the Minkowski sum through slope diagrams.

$$S = A \oplus B = \{a + b | a \in A, b \in B\} \tag{2.2}$$

$$B' = \{-b : b \in B\} \tag{2.3}$$

The computation of NFPs is done through three main approaches, which are known as Sliding algorithms, Slope diagrams and Polygonal decomposition into convex polygons. The first two approaches allows to obtain the full NFP, while the decomposition approach requires another step, which rebuilds the full NFP by merging the individual convex NFP components together.

The following subsections present the algorithms used to compute NFPs, Sliding algorithm and Slope diagram, and Polygonal decomposition which is divided into two parts, one that focuses on decomposition only, and the other on the merging of the decomposed NFP components.

2.5.4.1 Sliding Algorithm

The sliding algorithm, as proposed by Mahadevan (1984), is used to compute the NFP from pairs of polygons through a method that mimmics the sliding of a mobile piece around a static piece, without losing contact. The NFP is constructed by drawing the set of positions of the mobile piece reference point around the static piece until it reaches the original position. The NFP generated is simple, and convex, if both polygons are convex. When the polygons involved are not convex, it may lead to NFPs with very complex outlines. Difficulties arise when one or both polygons contain concavities with narrow entries or internal holes. When the sliding polygon is capable of reaching all touching positions on the outline of the fixed polygon, the resulting NFP is simply connected. When pairs of polygons that may be simply connected themselves but cannot reach all points of their outlines when sliding, they are referred as multiple connected polygons and result in an NFP with multiple components. Some other cases can also be referred as being not strictly multiple connected, for example when a polygon slides into a concavity in one direction only. Several improvements to the sliding algorithm have been proposed by Whitwell (2004), that improve computational efficiency and allow to deal with exact fit sliding through a “passageways”, among other improvements.

The sliding algorithm starts with two pieces touching each other, A and B, where the vertex with the highest value of y-axis coordinate of piece A, is touching the lowest y-axis coordinate of the piece B. An illustration about the sliding is shown on Fig. 2.39. The piece A is denominated

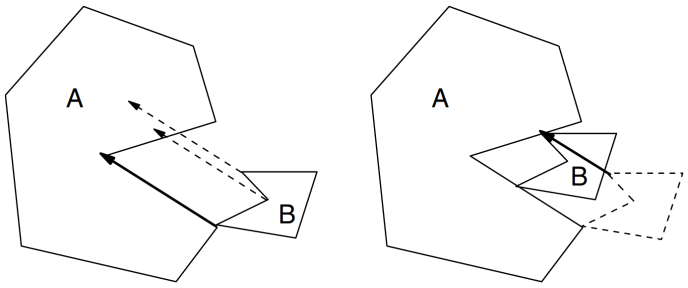


Figure 2.39: Sliding distance according to the available length that maintains no overlapping between the pieces (adapted from Bennell and Oliveira (2008))

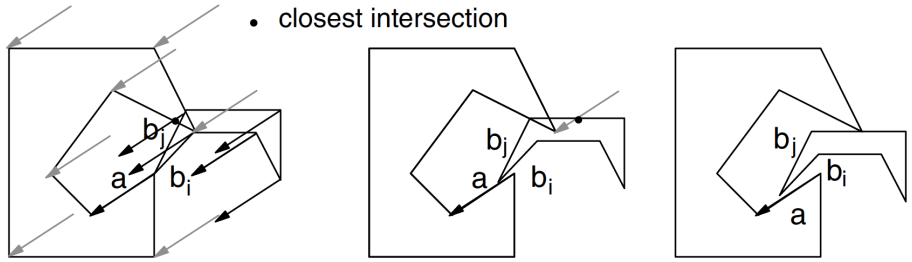


Figure 2.40: Search method to find a possible positioning place on the hole of piece A (adapted from Bennell and Oliveira (2008))

static piece, and piece B as the orbital piece, since piece B will slide around piece A without losing contact, while drawing the NFP with the reference point of B. If both pieces are convex, B slides all the length of the edges of piece A, until B has completely traveled around A. When the polygons have concavities, there is the possibility some edges might not be partially or totally slided. While sliding, to avoid overlapping between the polygons, the vertices of the orbital polygon, B, are projected on the direction of the sliding, the same length of the edge of the static polygon that is being slided upon. The D-function is used to compute for intersections, and the slided distance is the shortest available which conforms to no overlapping between both pieces.

The limitations to this algorithm is that it does not identify possible placement positions of B inside a hole of A, or into unreachable positions in the outline of piece A, as in the case of concavities that do not allow the orbital piece to slide inside while being able to contain B inside without overlap. This example can be seen on Fig. 2.40. Extensions to this algorithm exist, such as Burke et al. (2006), which tackle this issue by detecting which edges of A are in contact with B, and then searching for holes. All the edges that were not slided or were partially slided are possible candidates to being part of a hole. This is done by first finding the vertex of B that can slide upon those edges, and then using the same previous method to find the available distance to be slided that still maintains the no overlapping condition. If a valid positioning point is found, the normal sliding algorithm is applied, and the respective IFP is computed.

Although this method enables the detection of holes and perfect fits, it may not be the most efficient method to compute it.

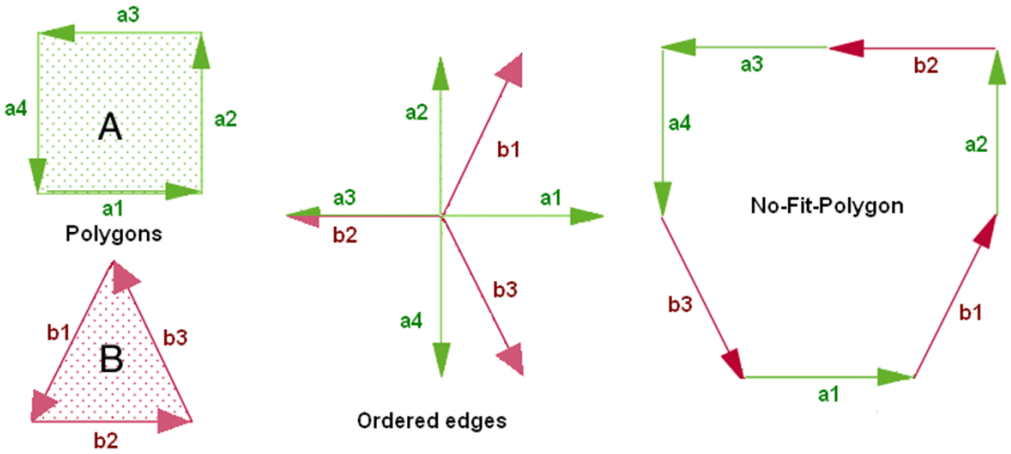


Figure 2.41: NFP between two convex polygons (adapted from Bennell and Oliveira (2008))

2.5.4.2 Slope Diagrams

Cuninghame-Green (1989) proposed a method to compute the NFP for pairs of convex polygons, based on slope diagrams. Slope diagrams can be used with convex polygons without difficulty, since it keeps the order of edges of both polygons, and will generate the NFP while navigating through the list and adding each new edge to the previous edges. The specific procedure to generate the NFP requires inverting the orientations of the edges of one polygon, while maintaining the orientations for the second polygon, and ordering all the edges by slope which defines the sequence of edges in the NFP. This procedure is illustrated on Fig. 2.41. On the left side of Fig. 2.41, both polygons are shown, polygon A and polygon B, with their edges oriented in counter-clockwise direction. In the middle, the oriented edges are placed accordingly to their slope, with the direction of edged of polygon B inverted (in this diagram only the slopes of the polygons are considered, the size of the edges is not represented). On the right side, the NFP between polygon A and B can be seen. The difficulties arise when dealing with non-convex polygons, due to difficulties in developing a robust NFP generator for them, limiting its use.

Ghosh (1991) extended Cuninghame-Green (1989) method for non-convex cases. When one of the polygons has concavities, this approach changes, since the edges of the concave polygon will not be ordered by slope. The edges of the convex polygon, which have slopes between the slopes of the edges of the concave polygon, will be repeated every time they pass through the concavity, when building the NFP. They are added with a positive sign to the NFP when entering the concavity and added with a negative sign when leaving it. This method also works when both polygons are concave, as long as their concavities do not interact.

Bennell et al. (2001) deals with the problem of two concave polygons, where the concave edges of polygon B are replaced by artificial edges that end up simulating a convex polygon. This produces an NFP from polygon A with the "convex" polygon B, where the original concave edges will replace the previous artificial ones including any edges from A that are traversed wherever they appear on the slope diagram. Dean et al. (2006) present an extension of Ghosh (1991) algorithm,

using manipulation of sorted lists of polygon edges to find the NFP efficiently. Bennell and Song (2008) proposed modifications to Bennell et al. (2001), which refers to an approach that retains the concavities of the polygons while partitioning one of the polygons (such as B) into groups of convex or concave sequential edges. These groups are then merged with the slope diagram of the complete polygon (such as A), then following the original algorithm from Ghosh (1991) to create the NFP, and finally, removing any internal edges or intersections that were created, maintaining only the ones that belong to the outer limits.

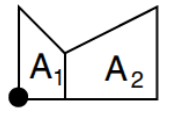
2.5.4.3 Polygonal Decomposition

Another alternative is to use polygonal decomposition, where each irregular polygon is decomposed into several convex sub-polygons. The NFPs from every pair of sub-polygons from different pieces are generated, and merged, to obtain the NFP from the original polygons. The most complicated step of this process is to merge the polygons efficiently and without precision errors. Achieving the minimum number of convex sub-polygons derived from the decomposition of the original polygons reduces the computational cost of the merging procedure, but it is computationally expensive by itself. The use of a greedy algorithm, while not achieving the optimal decomposition, might be good enough in reducing the numbers of sub-polygons. The number of total NFPs depend on the number of the convex sub-polygons generated from each piece, since each convex sub-polygon from one piece will be used with every convex sub-polygon from the other piece in generating the NFP. The procedure to generate the NFP from polygon decomposition also suffers from mathematical precision problems. Some authors that use convex decomposition are Watson and Tobias (1999), Babu and Babu (2001) and Agarwal et al. (2002). This procedure is presented in Fig. 2.42, where Fig. 2.42a is piece A and Fig. 2.42a is piece B, Fig. 2.42c shows the reference point at coordinates $(0,0)$ of every component of the pieces (A_1, A_2, B_1, B_2), Fig. 2.42d shows the NFP computed from pairs of components of different pieces, and Fig. 2.42e shows all NFPs overlapped to return the complete NFP derived from both complete pieces.

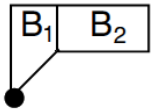
Although the NFP brings many benefits in its use, mainly the efficient verification of the relative position between pieces, calculating it is not a trivial task. Each approach has limitations, but some are more critical than others. Holes and perfect fits need to be correctly detected and represented, which can become difficult due to numerical precision problems. The major limitation of the NFP is that it cannot represent the relative position between two pieces in any rotation, without recomputing for the desired rotation.

2.5.4.4 NFP Component Merging algorithm

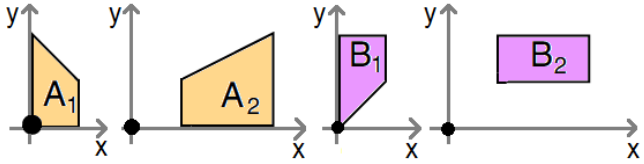
An important tool to generate NFPs can be found in Sato et al. (2013), where it explores the concept of collision free region, which represents the set of translations that places a movable item in the interior of the container, without colliding with the already placed items. The collision free region is derived from the union of the NFPs generated between the movable item and the container with the already placed objects in it. The union of the NFPs is achieved through Boolean



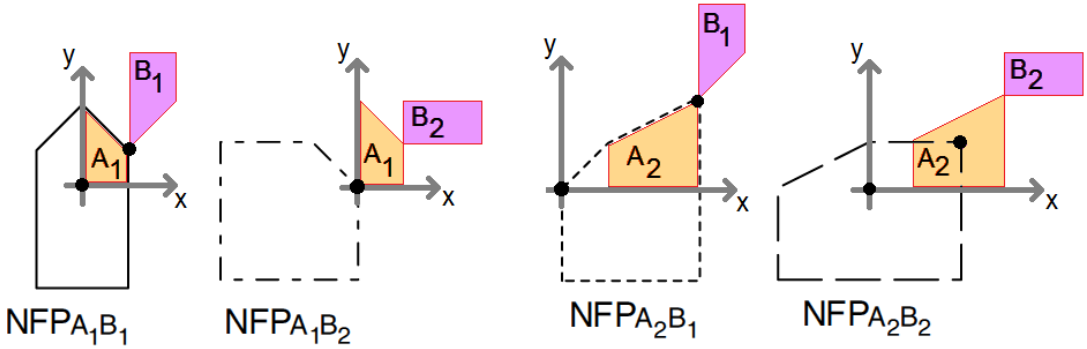
(a) Polygon A



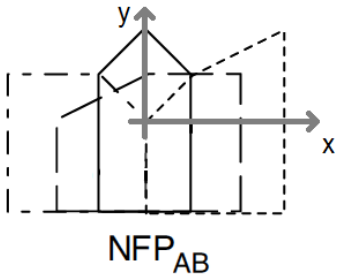
(b) Polygon B



(c) NFP between convex components



(d) NFP between pairs of components



(e) Union of all NFPs between Piece A and Piece B

Figure 2.42: NFP computation through convex decomposition (adapted from Bennell and Oliveira (2008))

operations. The union of NFPs generates degenerate edges and vertices, which represent a sliding fit (movement allowed in only one direction) or a perfect fit(no overlapping position, but no movement allowed), respectively.

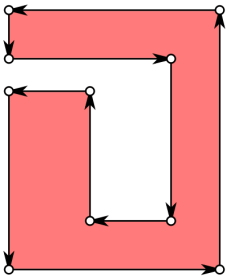
In Sato et al. (2013), the Boolean operations generate the NFP from two polygons, polygon A (Fig. 2.43a) and polygon B (Fig. 2.43b). The overlap between both polygons generates the intersections shown in Fig. 2.43c. The intersecting vertices between both polygons are determined, and edges are divided by the vertices in both polygons. The vertices of the polygons are classified as internal, external or in the boundary, while considering cases where both polygons share an edge that may have opposed or identical orientations. The edges of the polygons are also classified as internal, external, coincident shared or opposite shared, as seen on Fig. 2.43d. The resulting union of the polygons can be seen in Fig. 2.43e and their subtraction is seen in Fig. 2.43f. In the union case, the opposed shared edges create internal degenerated edges, where in the subtraction case, the coincident shared edges generate external degenerated edges.

The main advantages of using a single polygonal structure to define the collision free region, instead of multiple simpler polygons (such as in the case of convex decomposition), consists in having a reduced number edges and vertices to verify against the item reference point. However, the comparisons are more complex due to concavities and degenerated cases that appear, than those made when using only convex polygons obtained from convex decomposition. The instances where this approach provides benefits occurs when a large number of convex polygons used to describe the collision free region is so computational expensive, that using a single NFP, even with more complex operations, is less computationally demanding. The trade-off is trading the computation of a large number of simple items, against a single item with higher complexity.

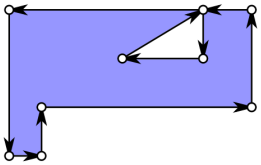
2.5.5 Medial Axis Construction

The Medial Axis (MA) of an object represents, according to Blum and Nagel (1978), a particular description of an object that is centered on the object. It is also known as skeleton or symmetric axis transform, and is described as the collection of maximal discs that fit inside the object but not inside any other disc. The description of the M.A. consists on the locus of centers and the radius at each point. Using this description, it is possible to reduce a shape into its skeleton, and when required, reconstruct the boundary of the shape from it.

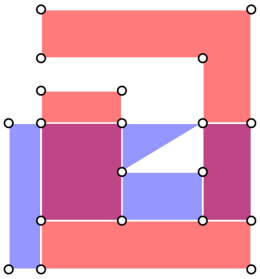
The M.A. use has been limited due to several difficulties. One difficulty consists in its volatility, since very small changes on the outline of a shape can induce large changes in the M.A. Another difficulty is its complexity of accurate computation, due to underlying geometrical complexity. The construction of the medial axis for polyhedral models is a complex and computationally expensive task. As mentioned in Foskey et al. (2003), when computing the M.A. for a polyhedron, the surfaces that constitute the M.A. are quadrics, and seam curves can have degree four, while when dealing with solids with curved boundaries, the M.A. sheets and seam curves can have much higher degrees, which makes the M.A. computation extremely difficult to do reliable and with enough computational efficiency.



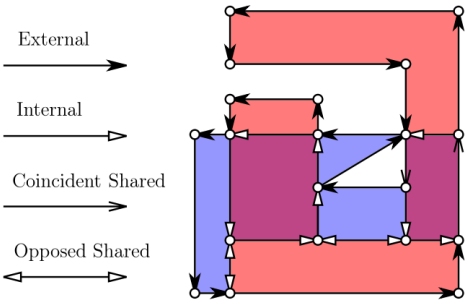
(a) Polygon A



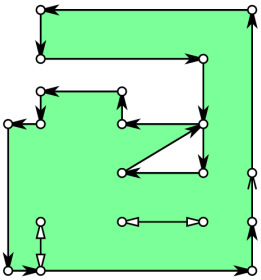
(b) Polygon B



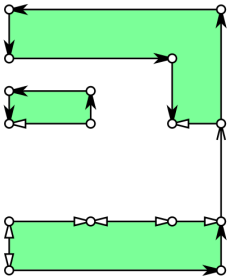
(c) Polygon A and Polygon B Intersection



(d) Edges from both polygons are classified as internal, external, coincident shared and opposite shared



(e) Union of Polygon A and B



(f) Subtraction of Polygon A and B

Figure 2.43: Polygonal Boolean operations for collision free region (adapted from Sato et al. (2013))

According to Foskey et al. (2003), the M.A. construction can be classified into four different categories, being them thinning algorithms, distance field based algorithms, algebraic methods and surface-sampling approaches, where their main differences are the representation used and the method of construction.

Thinning algorithms use voxel or pixel based representation of the initial object, and perform erosion operations to arrive at the set of pixels/voxels that are an approximation to the Medial Axis. Thinning is commonly used in digital image processing, pattern recognition, image analysis, signature verification, among others. A survey of these approaches can be found in Lam et al. (1992), and a comparison in Zhang and Wang (1993). Another comparative survey can also be found in Vincze and Kovári (2009) where they compare several thinning algorithms to a reference skeleton. Németh and Palágyi (2011) present six different 2D thinning algorithms based on a generalization of curve/surface interior points that are called isthmuses. An example of this tining algorithm can be seen in Fig. 2.44, where a skeleton is generated from the letter P, using on the left side endpoint preservation (i.e. the external vertices of the piece are preserved into the final skeleton) and on the right side, the isthmus interior points are the points that define the skeleton.

Distance fields also use pixel or voxel based representation of the objects, but every pixel/voxel contains the euclidean distance to the nearest point on the boundary of the object. Danielsson (1980) uses this approach for 2D, and an extension to 3D can be found in Ragnemalm (1933). Another approach consists in using spatial subdivision of vertices (that represent the M.A.) in different Voronoi regions depending on a threshold (Vleugels and Overmars (1995)). Hoff III et al. (1999) uses interpolation-based graphics hardware to construct a polygonal approximation of the distance field that is created in the depth buffer. Foskey et al. (2003) extended the algorithm of Hoff III et al. (1999) to compute the gradient of the distance field and using it for fast computation of the M.A.. Siddiqi et al. (2002) uses a differential equation simulating the inward progress of a front that starts at the objects boundary and moves towards the center. Every point on the boundary moves according to a vector created from every point inside the outline oriented the nearest point on the outline. An example of a thinning algorithm by Siddiqi et al. (2002) can be seen in Fig. 2.45.

Algebraic methods used to construct the M.A. can be seen in Etzion and Rappoport (1999), where spatial partition is used, dividing regions until the curves are all connected or minimum size of the regions is achieved. Tracing approaches Milenkovic (1993) start from a junction point, and follow the seam that leaves it, recursively arriving at another junction point and following another seam. An overview of other tracing approaches can be seen in Foskey et al. (2003). One method that allows to compute the MA is presented in Yao and Rokne (1991), but a more detailed description can be found on a technical paper by Edwards (2010). It allows the construction of the MA from simple polygons, returning only straight

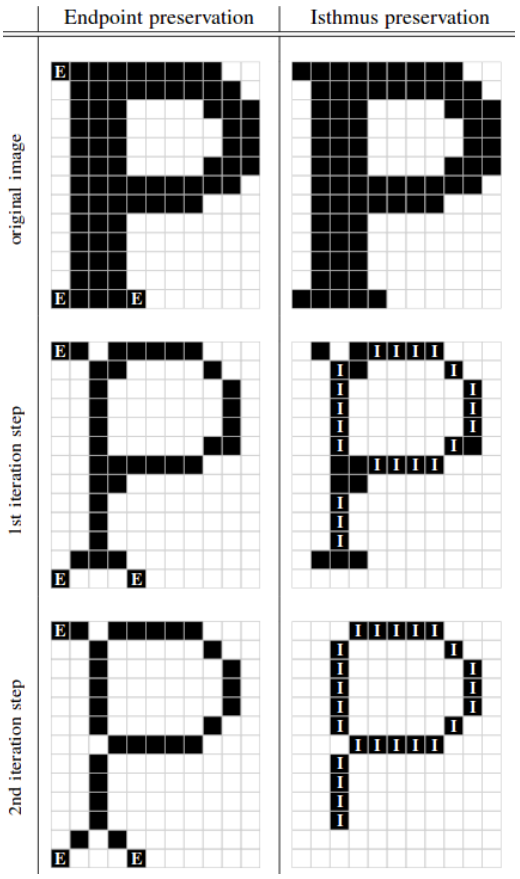


Figure 2.44: Thinning algorithm for skeleton generation (adapted from Németh and Palágyi (2011))

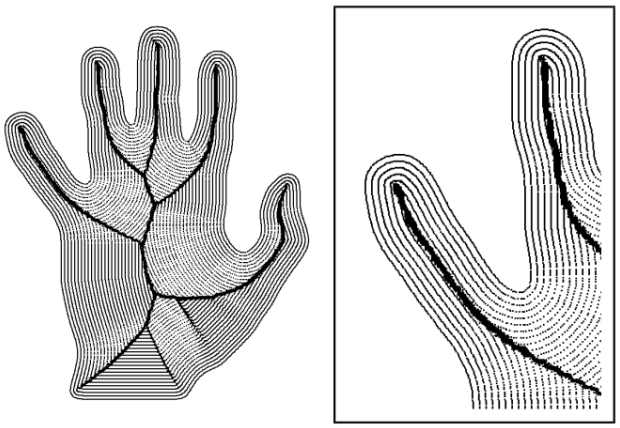


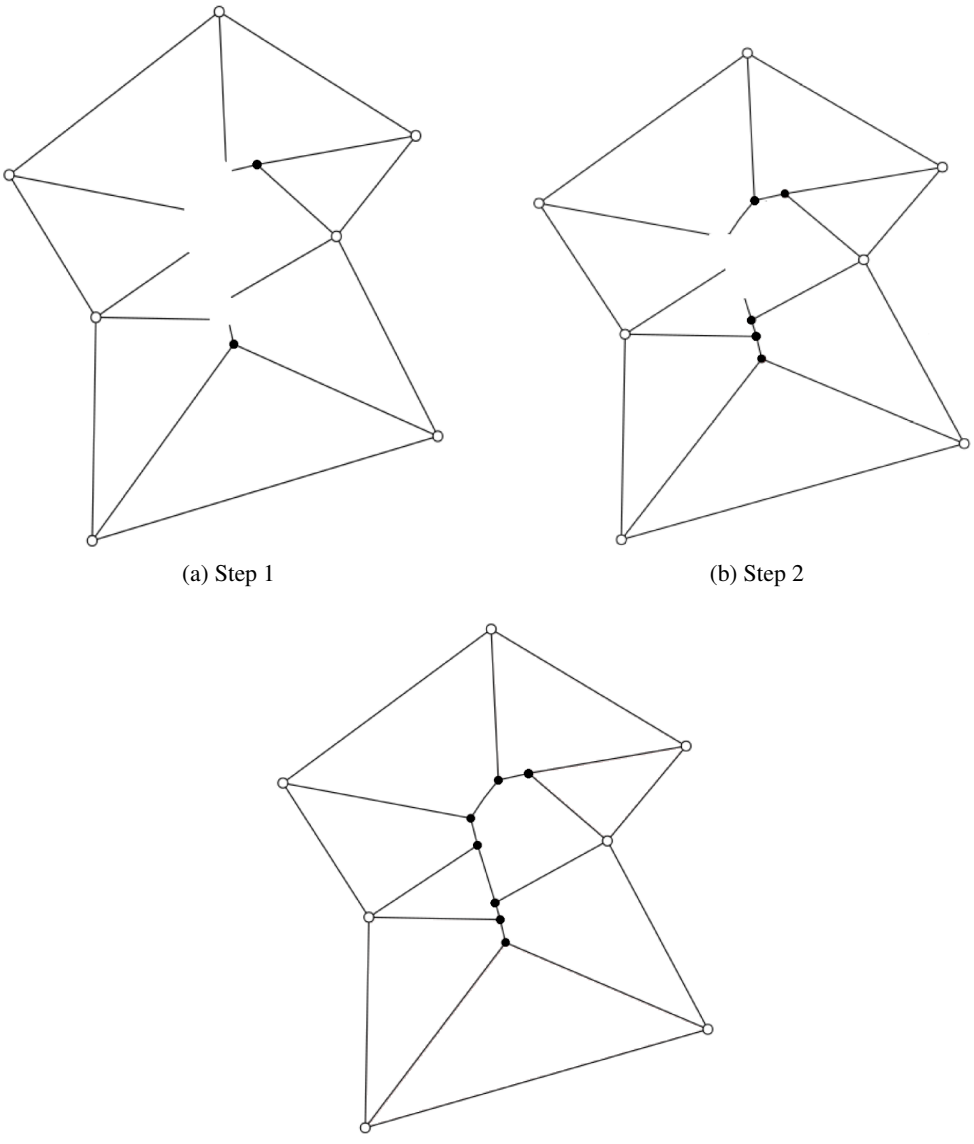
Figure 2.45: Distance fields to approximate the Medial Axis (adapted from Siddiqi et al. (2002))

lines on the skeleton of convex polygons, and returning a set of straight lines and parabolic arcs from irregular polygons, i.e. non regular and non convex polygons. This method is iterative, which leading to increased numerical precision errors as the complexity of the object increases. An example with an iterative construction of the M.A. from the technical paper by Edwards (2010) can be seen in Fig. 2.46

Surface sampling approaches consist in approximating the M.A. through a subset of the Voronoi diagram of a point cloud that is concentrated on the boundary of the object. Several authors use this approach, such as Amenta et al. (2001), Dey and Zhao (2002), Dey and Zhao (2003) that create a simplified model of the M.A., and Turkiyyah et al. (1997) focuses on increased accuracy instead of a simplification. A difficulty that arises when applying these methods is that in order to achieve a tight approximation to the M.A. the number of points on the boundary needs to be large. Bradshaw and O'Sullivan (2004) uses a Voronoi diagram to represent the M.A. and extended Hubbard (1995) method, which was based on a static M.A. approximation, to support dynamic M.A. approximation while building a sphere-tree. A example of a surface sampling approach can be seen in Fig. 2.47, where the darker points refer to the points that hare approximated to the real M.A., the white points refer to the external M.A., and the gray points are the points on the boundary of the objects from where the inner points are deduced and the approximation to the M.A. is made.

2.6 Concluding Remarks

The current literature presents multiple approaches to solve C&P problems, without any clear definitive answer to which is the most promising path to be explored. C&P problems, and also Nesting problems, without fixed orientations, have received focus in their development due to their simpler requirements compared to the same problems with continuous rotations. The geometric representations available (grid, polygons, NFP) are properly adapted to be used with fixed or discrete orientations, but unable to be computationally efficient when dealing with continuous rotations. Using Phi-Functions can address the problem of free rotations at the expense of high complexity and computational cost. In order to adequately address the Nesting problem with continuous rotations a proper geometric representation, that takes into consideration its specific requirements, is necessary. Among the several explored representations, the most adequate representation to deal with the requirements of efficient overlap computation and continuous rotations is a geometric representation based on Circle Covering. The overlap computation between circles is very straightforward, and with a reduced computational cost, however, methods that allow representing pieces, with controlled accuracy, and a reduced number of circles are hard to find. A promising method uses the algorithm from Zhang and Zhang (2009). However, more efficient circle covering representations might be achieved through other methods, such as the Medial Axis. Considering the characteristics of the circle covering geometrical representation, and the requirements of the Nesting problem, a possible approach is through non-linear mathematical models.



(a) Step 1

(b) Step 2

(c) Step 3

Figure 2.46: Iterative construction of a Medial Axis (adapted from Edwards (2010))

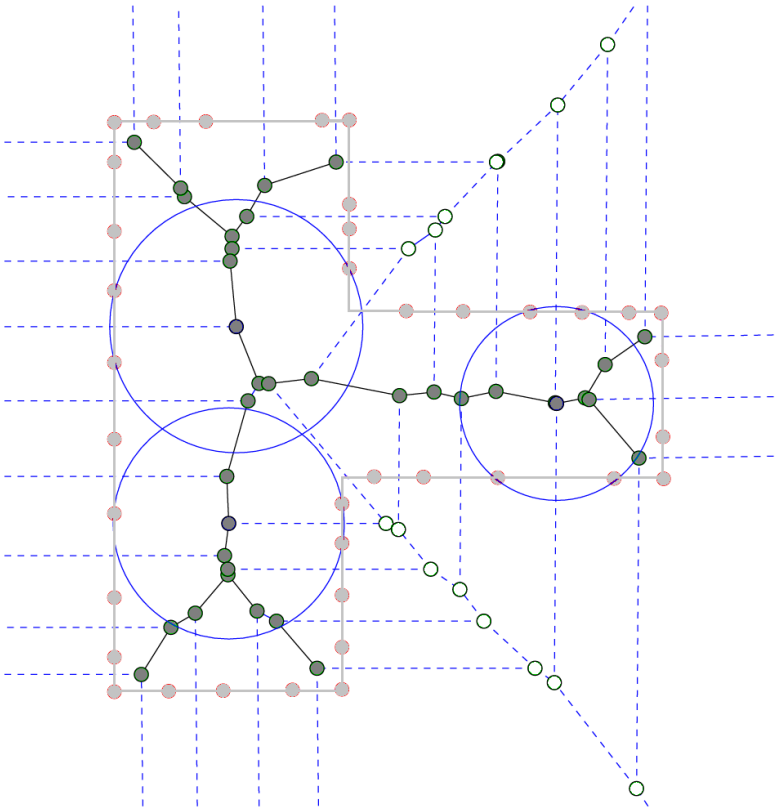


Figure 2.47: Surface sampling to approximate the Medial Axis (adapted from Bradshaw and O'Sullivan (2004))

NLP models have not been as explored as other types of models, due to their difficulty, although it is an area that has significant potential to be developed, which can bring significant improvements compared to currently used solution approaches. The use of NLP models is appropriate due to the non-linear equations derived from the non-overlapping constraints between pairs of circles. Non-Linear Programming Models able to deal with continuous translations and rotations are desired, that are able to return good quality solutions, but also at a reduced computational cost. Heuristics and other solution approaches might be required to complement the use of NLP models, to solve efficiently the Nesting problem with continuous rotations. For this reason, using a combination of Circle Covering Representation with Non-Linear Programming models may be a promising approach to address the Nesting problem with continuous rotations.

Chapter 3

Geometric Representation based on Circle Covering

In this chapter, the main focus will be on the geometrical component of the Nesting problem with continuous rotations. This problem requires an approach with complete support for full rotations of the pieces involved, and easy overlap verification. Using methods such as the NFP or grids to ensure feasibility (i.e., to check overlap) is not an efficient approach when pieces/polygons can rotate freely. Representations based on Phi-Functions, while able to support continuous rotations, may not provide the most efficient method to represent and verify overlaps, since it uses multiple non-uniform primary shapes to represent pieces. Comparing pieces this way requires having constraints to check overlap between different types of primary shapes, which may not be efficient if pieces are composed by many primary shapes with high complexity. For this reason, using only circles to represent pieces is a good solution, since the circles do not need to be rotated around their centers (only rotating their centers around a reference point), and they have simple overlap computation between each other. Since the Nesting problem deals with irregular pieces, which is usually represented by irregular polygons, the choice of representation based on circles necessarily leads to an approximated representation. Better approximations to pieces require the use of more circles, which becomes prohibitive due to the exponential increase in the number of non-overlapping conditions. The main difficulty is to achieve a good trade-off between the approximation quality and the number of circles used, i.e., to find a good approximation to the piece with the minimum number of circles. So, finding the best placement configuration of the circles is an important issue, which is also a great challenge.

The geometrical aspects of the Circle Covering (*CC*) representation are discussed in this chapter, introducing the concepts of different types of CC, defined as Complete, Partial and Inner Circle Covering (*CCC*, *PCC* and *ICC*, respectively), and multiple levels of approximation quality, such as Low and High resolution (*LR* and *HR*). These developments enable a CC approach capable of generating circle coverings that take into account a set of characteristics, derived from a specific

problem, which enable tackling the problem with great efficiency. Since this approach enables generating the CC representation without human intervention, it can be adjusted in real-time, allowing for a fully automatic production process.

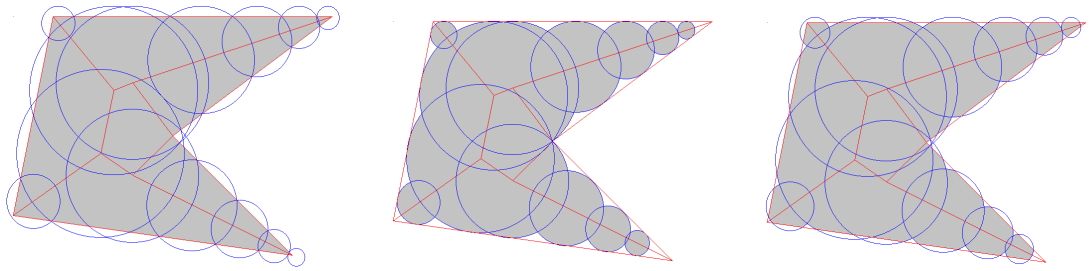
This chapter will start with a section presenting the different types of CC. The following sections discuss the approaches that were explored in order to find an adequate CC, taking into account the characteristics of the Nesting problem with continuous rotations. The first approach is a Hierarchical Complete Circle Covering, followed by an iterative approach with the aim to also generate a Complete Circle Covering representation. This iterative approach is then expanded to support the other circle covering types, while also exploring further enhancements. The final sections deal with the impact of different levels of quality of approximation, and ending with the final remarks.

3.1 Types of Circle Covering

The Nesting problem with continuous rotations requires an adequate geometrical representation of the pieces so that approaches that tackle it are computationally efficient. Depending on the required quality of the approximation and the specific characteristics of the Nesting application, several types of coverings can be defined. The Complete Circle Covering (*CCC*) (Fig. 3.1a) consists in completely covering the piece by a set of circles. When the piece is not completely covered by circles, the covering is defined as an incomplete covering. Among incomplete coverings, a specific case occurs when the set of circles is partially covering the piece without any circle exceeding its outline, which is defined as Inner Circle Covering (*ICC*) (Fig. 3.1b). An intermediate situation, defined as Partial Circle Covering (*PCC*), occurs when the set of circles does not entirely cover the piece and neither does the piece contain completely the set of circles (Fig. 3.1c). The concept of *PCC* is similar to the *Three-Step+Gap* approach from (Zhang and Zhang, 2009), where circles are placed, and separated by a specified gap between each other, when measured on the outline of the piece. Other approaches that generate circle coverings can be seen in Imamichi and Nagamochi (2008), where the circle covering is achieved through the use of a grid, where the circles are placed (if the points are inside the piece) and expanded until they reach the outline; in (Zhang and Zhang, 2009), where circle covering focus on covering outlines of pieces; and Jones (2013), where an iterative process is used recursively to place the biggest possible circle inside a piece.

3.2 Hierarchical Circle Covering Approach

The initial approach to the CC problem focuses on achieving the *CCC* of a polygon with a set of unequal circles, based on convex decomposition and the Minimum Enclosing Circle (MEC) algorithm. It is an iterative approach that basically decomposes the pieces into convex polygons and then covers each convex polygon with its MEC. When a particular MEC leads to a bad approximation, the respective convex polygon is further divided.



(a) Complete Circle Covering (*CCC*). (b) Inner Circle Covering (*ICC*). (c) Partial Circle Covering (*PCC*).

Figure 3.1: Types of Circle Covering.

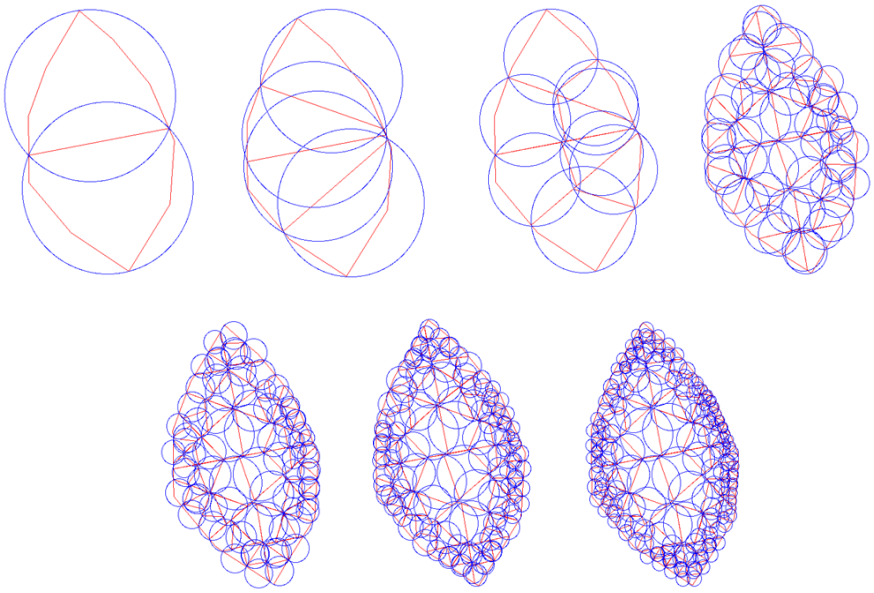


Figure 3.2: Convex decomposition with Minimum Enclosing Circles.

The first step of the approach is based on using convex decomposition to form convex polygons which are used to define the placement of the circles. In the second step, the MEC is created for each convex polygon, which is defined as the circle of minimum radius that contains the convex polygon. The third and final step consists in verifying the difference in area between the original piece, and the covered area by all of the MEC. If the difference is above a given threshold, further subdivision is done. This subdivision is done only on the polygons whose MECs lead to bad approximations, by dividing them into two convex polygons through the longest line that connects its most distant points. The main advantage of this approach is its inherent hierarchy. As a disadvantage, it is hard to compute exactly the difference between the area of the polygon and the area of the circle covering, due to the high number of circles overlapping each other. An example of the progression of this method can be seen on Fig. 3.2.

The efficiency of this method greatly depends on the algorithm used for convex decomposition, since it may not achieve the minimum number of convex polygons (optimal solution). When very tight threshold values are required (difference of areas), the number of circles will grow exponentially. This method can be used to dynamically adjust the approximation error of the piece, on demand, by using a different approximation level. A less refined level has a lower number of circles, which is beneficial when trying to reduce the computational cost of comparing overlap between many pieces. The main benefit of this approach is that it allows different levels of approximation to the original piece, organized as an hierarchy of low to high quality approximations. Although this approach can achieve a *CCC* with high quality, its number of circles is prohibitive.

3.3 Complete Circle Covering Approach

In order to achieve a *CCC* with a reduced number of circles, without having to compromise quality of approximation, another approach was explored. The main difficulty that arises when generating a *CCC* is where to place the circles in order to achieve a low approximation error with the lowest number of circles. Some details can be deduced beforehand, such as: the position of the center of the circles cannot be outside of the outline of the enclosed polygon, and that better results are achieved if each circle covers the maximum possible area. This means that each circle should have the highest possible radius without exceeding the admissible distance outside of the enclosed piece. This observation leads to the approach that obtains the Complete Circle Covering with the assistance of a topological skeleton called Medial Axis (MA). Placing the circles on the MA will lead to a low number of circles since it allows to place bigger circles and to achieve a significant reduction in the number of possible placement positions for the circle centers.

3.3.1 Medial Axis Algorithm

The MA is a topological skeleton which defines inner points of a polygon that are equidistant to at least two points in its outline. The MA derived from convex polygons is composed entirely by straight lines, but when derived from non-convex polygons, it also produces parabolic arcs. The parabolic arcs, which are generated by the concave vertices, are approximated by straight edges to simplify the process. An example of a MA with parabolic arcs generated from an irregular piece can be seen in Fig. 3.3. The bones can be distinguished as interior and exterior bones, where the first are all bones of the MA that do not have connections to the outline of the polygon, and where the latter consists of the bones that have connections to the outline. Further details about possible methods can be seen in Chapter 2, section 2.5.5.

Our current approach to generating the MA is based on the method from Yao and Rokne (1991) but following the algorithm described in (Edwards, 2010). The greatest problem that this method presents is the difficulty in dealing with numerical precision problems, due to its iterative nature. When extracting the MA from a very complex polygon, the skeleton might become deformed due

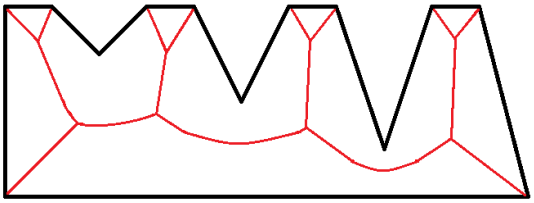


Figure 3.3: Medial Axis of an Irregular Piece.

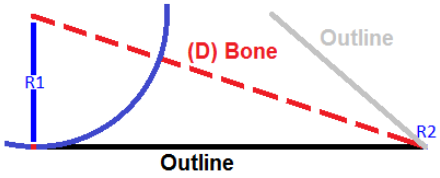


Figure 3.4: A bone of the skeleton (dashed line) and part of the outline of the piece.

to accumulated error. Yet, despite its difficulties, this method is very promising due to its generation of parabolic arcs that allow computation of placement positions on its MA with controlled approximation error.

3.3.2 *CCC-MA* Algorithm

The *CCC-MA* algorithm works upon each bone of the skeleton individually. A visual example can be seen in Fig. 3.4, where the iterative method places the center of the circles in the MA, while taking into account the threshold that regulates the approximation error to the outline of the polygon. It operates using the distances to the outline ($R1$ and $R2$) at the extremities of the bone and the length of each bone (D). Distances $R1$ and $R2$ are measured perpendicularly to the edge of the outline, since a circle that is touching the outline has its center contained in the same line (perpendicular to the outline) that contains the contact point.

The extremities of each bone contain a circle with a radius equal to the nearest point in the piece outline. Both circles are then expanded by the threshold value that is defined by the approximation error. The threshold can be seen in Fig. 3.5. The algorithm iteratively computes the next center position, starting from the biggest circle (in order to cover the largest area possible), making sure that the circles intersect each other on the outline of the piece, until they reach the opposite circle. The radius of each new circle is easily computed since it is a linear function that depends on the difference of both radius (at the extremities of the bone) and the distance that separates them. When dealing with bones that connect to the outline of the piece, the algorithm stops placing circles when they cover the bone completely. This procedure is also presented in Fig. 3.6.

The algorithm computes every bone of the skeleton independently, taking into consideration the radius of the circles at the extremities of each bone, their length and the threshold that each

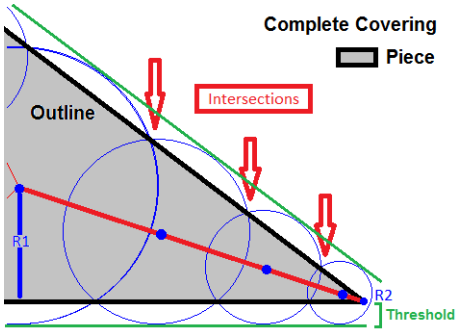


Figure 3.5: Circle placement positions, with $Threshold = Excess\ distance$.

***CCC-MA* algorithm**

for all bones **do**
  Expand both circles radius by the threshold value
  Select biggest circle
  repeat
    Compute circle and outline intersection
    Compute next circle center position on the bone[1]
    Save circle position
  until Circles completely cover the bone
end for

[1]Taking into account that it must also intersect the previous circle/outline intersection and that the radius of the circle depends on the position of its center on the bone.

Figure 3.6: *CCC-MA* pseudocode.

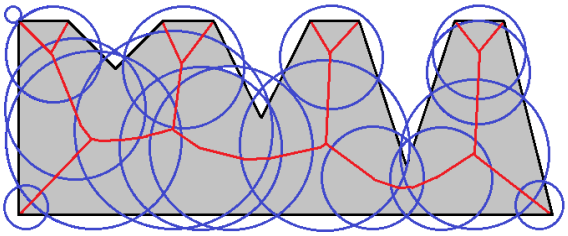


Figure 3.7: Complete Circle Covering of a piece.

circle is allowed to expand. After repeating the same operation on every bone of the skeleton, the *CCC* of a piece is achieved. An example of a piece with its *CCC* computed, using its MA skeleton, is shown in Fig. 3.7.

3.3.3 Results and Discussion

Using Circle Covering representation for Nesting problems with continuous rotations requires a good management of the trade-off between the quality of the piece approximation and the number of circles generated. Good quality approximations are obtained at the expense of a greater number of circles, which reduces computational efficiency. The variation of the threshold value that controls the approximation can produce different levels of approximation for each Circle Covering, therefore it is useful to be able to control that trade-off with precision.

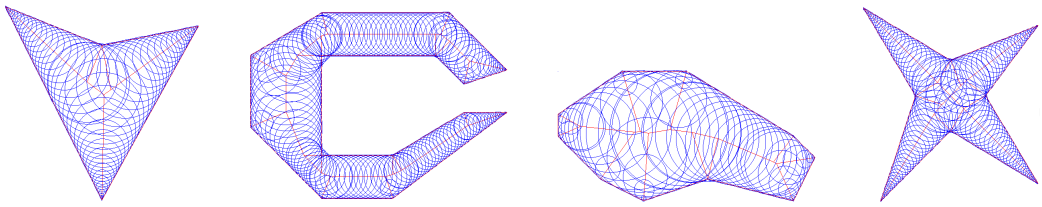
To assess and evaluate the behavior of the CCC-MA algorithm, a set of experiments were performed regarding the variation of the threshold that defines the approximation error and its impact on the approximation quality and number of circles. In order to verify the efficiency of the approach, four pieces with different characteristics were chosen and tested with approximation threshold values ranging from 0.25 to 0.01 units. The threshold is expressed in units of length, and is not relative to the size of the piece. As an example, if we use 0.01 units for threshold, it will be the same distance whether the rectangle has 10 or 15 units of length. The results obtained are summarized on Table 3.1, where for each piece two columns are presented (titled, respectively, circles # and area %). The first column shows the number of circles used in the covering, and the second one shows the additional area related to the original piece area. In this table, it can be observed that a linear reduction in the threshold leads to an exponential increase on the number of circles and an almost linear reduction in the additional area.

The *CCC-MA* (Medial Axis followed by Circle Covering) of these four pieces denoted $P1$, $P2$, $P3$ and $P4$, can be seen on Fig. 3.8, with threshold of 0.01 units.

An important issue that must be analyzed is the trade-off between the increase in the number of circles and the reduction of the additional area. Table 3.1 already shows this trade-off for pieces $P1$, $P2$, $P3$ and $P4$, where initially a decrease in the threshold value (for example, from 0.25 to 0.20) leads to a small increase in the number of circles and a significant reduction in the additional area (one more circle in $P1$ reduced the area by 3.2%), while later, the same reduction in the threshold value leads to large increase in the number of circles and a small reduction in the

Table 3.1: Number of circles and added area for different threshold values.

	Pieces							
	$P1$		$P2$		$P3$		$P4$	
Threshold	circles	area	circles	area	circles	area	circles	area
Value	#	%	#	%	#	%	#	%
0.25	13	14.1	35	13.2	12	12.1	28	20.6
0.20	14	10.9	37	10.6	13	9.6	29	15.7
0.15	17	8.3	41	7.9	14	7.1	36	12.2
0.10	21	5.5	48	5.2	15	4.4	41	7.9
0.05	31	2.7	70	2.6	25	2.1	61	4.0
0.01	72	0.7	161	0.1	55	0.4	141	0.9



(a) $P1$. (b) $P2$. (c) $P3$. (d) $P4$.

Figure 3.8: *CCC-MA* results for pieces $P1$, $P2$, $P3$ and $P4$ with 0.01 units for threshold.

additional area (Reducing threshold from 0.10 to 0.05 in $P1$ leads to an increase of 10 circles, and area reduction of only 2.8%) (Fig. 3.9). The graph on Fig. 3.9 shows a similar behavior for all 4 pieces. The evolution of the circle covering of piece $P3$ can be seen in Fig. 3.10, where the covering with the largest number of circles is also the closest to the real shape of the piece.

The complexity of the piece also has a negative impact on the performance of the algorithm, since the numerical precision errors increase at each iteration, due to having to compute multiple intersections, leading to incorrect skeletons being built. For complex pieces where the MA cannot be correctly constructed, convex decomposition is used on the original piece, computing the skeleton for each one, but with the downside of increasing substantially the number of circles. Comparisons of this *CCC-MA* approach were also made against the *Three-Step* and *Three-Step+Gap* approaches from (Zhang and Zhang, 2009). To compare the algorithms, the original pieces from (Zhang and Zhang, 2009) were used, according to the dimensions of the pieces and their threshold, as presented in (Zhang and Zhang, 2009). The pieces are presented in Fig. 3.11 and Fig. 3.12, having the threshold of 0.1 and 0.2, respectively.

Taking into consideration the results obtained from (Zhang and Zhang, 2009), Table 3.2 is presented, with the focus being on the total number of circles used to represent the same pieces, with the same threshold. Although (Zhang and Zhang, 2009) had three distinct algorithms, we compared only against the ones that allowed complete or nearly complete covering of the piece.

According to the results, as shown in Table 3.2, the *CCC-MA* approach managed to achieve 3

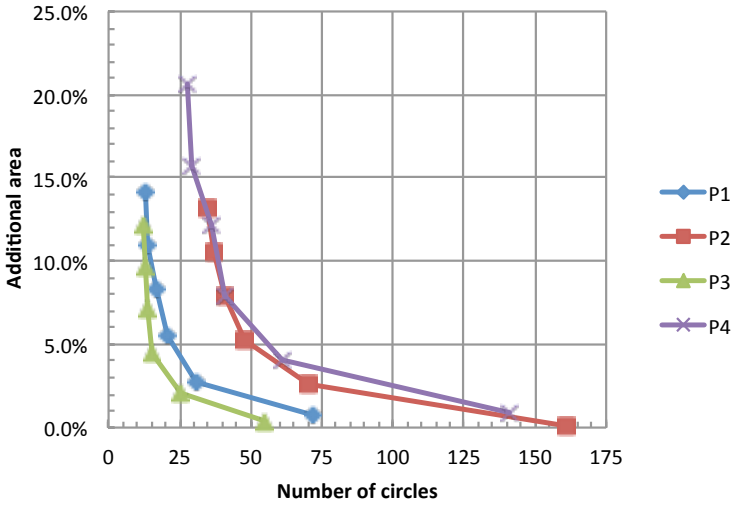
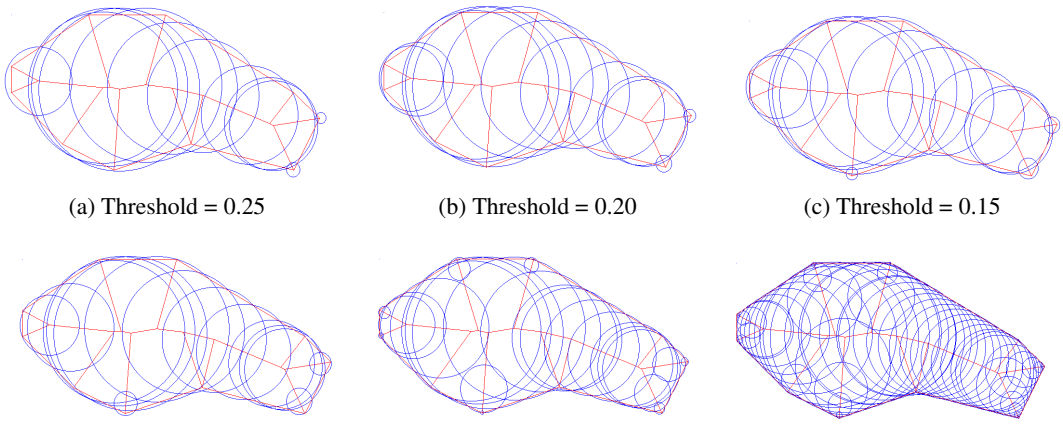


Figure 3.9: Trade-off between the number of circles and the additional area.



(a) Threshold = 0.25 (b) Threshold = 0.20 (c) Threshold = 0.15

(d) Threshold = 0.10 (e) Threshold = 0.05 (f) Threshold = 0.01

Figure 3.10: *CCC-MA* results evolution for piece *P*3

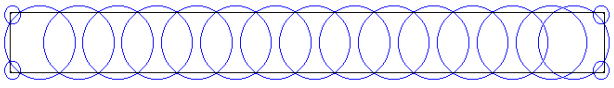


Figure 3.11: Circle covering of a rectangle, with 19 circles.

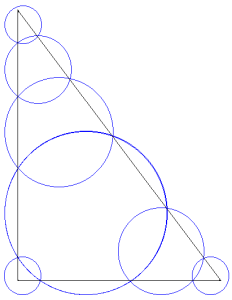


Figure 3.12: Circle covering of a triangle, with 7 circles.

Table 3.2: *CCC-MA* approach results.

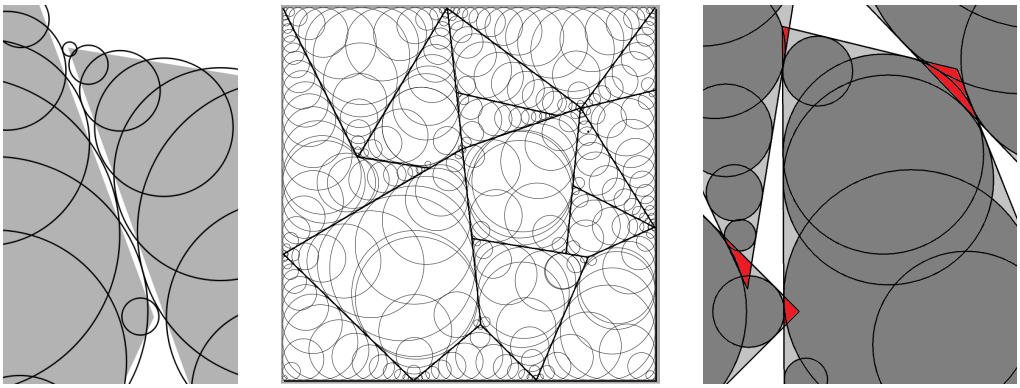
Algorithm	Pieces	
	Rectangle (# circles)	Triangle (# circles)
CCC-MA	19	7
Three-Step	22	8
Three-Step+Gap	21	6

and 2 less circles for the rectangle and 1 less and 1 more circles for the triangle, than the *Three-Step* and *Three-Step+Gap* approaches, respectively. The comparison against the *Three-Step+Gap* algorithm is not entirely fair since this approach does not ensure complete circle coverings.

3.4 Partial and Inner Circle Covering Approaches

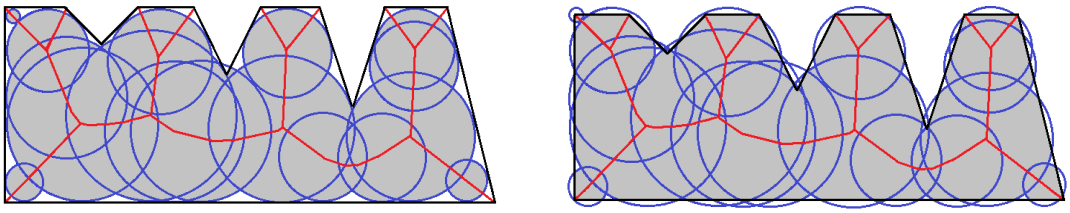
The different types of Circle Covering are useful for solving the Nesting problem with continuous rotations while considering different characteristics. The main advantage of using Complete Circle Covering (*CCC*) is that it creates layouts that are always admissible, as long as the circles representing different pieces do not overlap each other. One disadvantage is that due to the excess covering of the circles that protrudes the piece (defined by the threshold value that controls the approximation error) the pieces are kept apart by a maximum distance of twice the threshold value. This effect can be seen in Fig. 3.13a, where the two pieces (in gray) do not touch each other due to the presence of the circles used to represent each piece. This characteristic is clearly a disadvantage since it prevents perfect fits between pieces, which are crucial to obtain compact layouts. In order to minimize this effect, one could increase the resolution to achieve tighter coverings at the cost of a much greater number of circles, or selecting another type of covering.

To overcome the limitations inherent to the Complete Circle Covering *CCC* representation other circle covering representations can be used, such as Inner Circle Covering (*ICC*) and Partial Circle Covering (*PCC*). To be able to use these other circle covering representations the *CCC-MA* algorithm was extended into the *kCC-MA* algorithm. An example of the *ICC* can be seen in Fig. 3.14a and an example of the *PCC* can be seen in Fig. 3.14b. The main advantage of using *ICC*



(a) *CCC* piece separation. (b) *ICC* perfect fit. (c) *ICC* piece overlap.

Figure 3.13: *CCC* vs *ICC*.



(a) Inner Circle Covering with Medial Axis (b) Partial Circle Covering with Medial Axis

Figure 3.14: Inner and Partial Circle Covering

can be seen when attempting to solve Nesting problems with perfect fits, as seen in Fig. 3.13b, where the *CCC* would be unable to achieve those placement positions. The pieces represented by *ICC* can produce the closest placement positions at the expense of producing infeasible layouts in some cases due to overlaps, as seen in Fig. 3.13c. The *PCC* allows to achieve a trade-off between the layout admissibility and the separating distance between the pieces.

3.4.1 *kCC-MA* Algorithm

In the *CCC-MA* algorithm the quality of the approximation to the piece outline is controlled through the threshold parameter that measures the distance that the circles can exceed the outline. In the *kCC-MA* algorithm the threshold parameter is extended to measure also the maximum internal distance from the outline not covered by circles. This maximum internal distance will be denoted as the maximum penetration depth (Fig. 3.15) and the threshold is equal to the sum of the exceeding distance and the penetration depth. The *kCC-MA* algorithm supports not only *CCC*, but also *PCC* and *ICC* types of covering, where the *k* in *kCC* is used to identify the covering that is produced. The different configurations regarding protrusion (excess) distance and penetration depth allows the *kCC-MA* algorithm to obtain the three different types of coverings (*CCC*, *PCC* and *ICC*). To obtain a *CCC*, the threshold represents the maximum distance allowed for the circles to exceed the outline, while penetration depth being zero. To obtain a *ICC*, the threshold represents

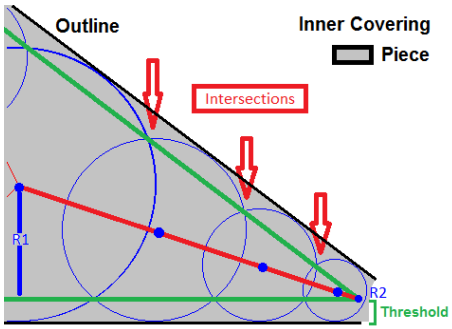


Figure 3.15: *ICC* placement positions, with $Threshold = PenetrationDepth$.

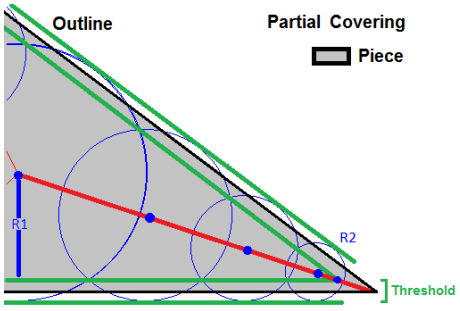
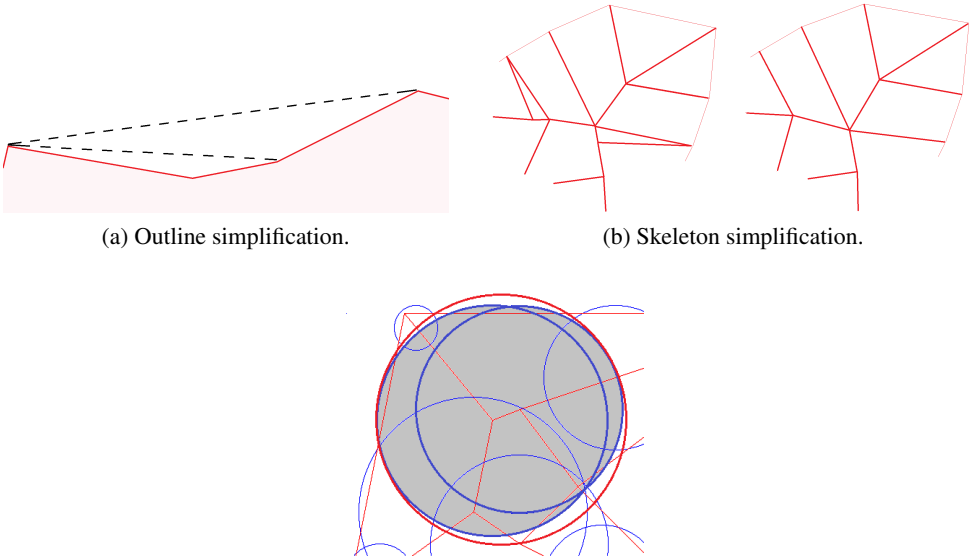


Figure 3.16: *PCC* placement positions, with $Threshold = ExcessDistance + PenetrationDepth$.

the penetration depth, while the maximum distance allowed for the circles to exceed the outline is zero. In order to compute the circle placement positions for *ICC*, the circles expand until touching the outline of the polygon and compute their intersections with the edge that corresponds to the $PenetrationDepth$ edge. From that point, the next circle center position can be computed. To obtain a *PCC*, the excess distance and penetration depth are both equal to half the threshold value (Fig. 3.16). The procedure to obtain the circle center placement positions follows the same logic as the *ICC*, by computing the intersections with the $PenetrationDepth$ supporting edge. However, the circles must now be expanded up to the supporting edge of the $ExcessDistance$.

PCC and *ICC* coverings obtained by the *kCC-MA* algorithm may have poor quality when covering pieces with very acute angles, i.e., narrow tips, which may lead to infeasible layouts. This is because the penetration depth is measured perpendicularly to the edges of the outline, causing a distance higher than the penetration depth to be uncovered from the vertex to the last placed circle. The algorithm keeps placing circles until the next circle has a radius that is inferior than the threshold value, which can occur with the *PCC* and *ICC*. This situation does not happen in *CCC* because the circle at the extremity of the bone never has a radius inferior than the threshold value.



(a) Outline simplification.

(b) Skeleton simplification.

(c) Circle merging.

Figure 3.17: Coverings simplifications.

3.4.2 Coverings Simplifications

Covering simplifications are required to decrease the total number of circles generated by the *kCC-MA* algorithm, since further improvement of the CC quality of the algorithm is very difficult to obtain. The complexity of a MA skeleton increases with the complexity of the piece outline, and the total number of circles increases with the complexity of the MA skeleton. For this reason, in order to reduce the total number of circles, simplifications to both piece outline and piece skeletons are required. The piece outline simplification consists in replacing two adjacent edges that form a wide angle on the concave vertex that connects them, by a single edge. This simplification takes into account the additional area that is added to the polygon, since it is always oriented to the reduction of concavities of the piece. Other possibility to simplify the piece outline is to erase the concave vertex on the outline that adds the least area to the polygon. This simplification process can be seen in the example presented in Fig. 3.17a. The simplification related to the piece skeleton consists in removing the bones of the skeleton smaller than a specified value, and connecting its extremities together. An example of this process can be seen in Fig. 3.17b.

Its possible to introduce another simplification when pairs of circles are almost fully contained one inside the other. Replacing both circles by an enclosing circle can allow further reduction of the total number of circles, as long as the reduction of the approximation quality of the covering is not greater than a specific value. The pairs of circles can be merged by replacing them by their Minimum Enclosing Circle (MEC). This process can be seen in Fig. 3.17c, where the circles merged produce a MEC that contains both circles, with a diameter equal to the distance between the two most distant points among them.

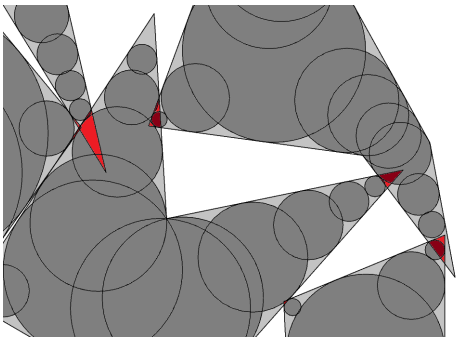
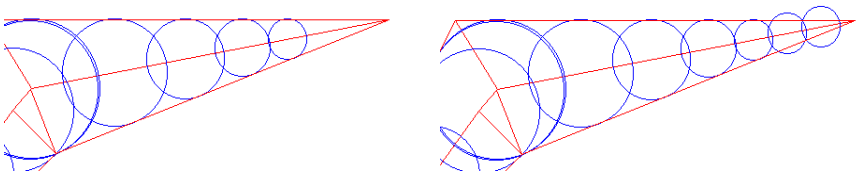


Figure 3.18: Overlapping between pieces due to uncovered tips.



(a) *ICC*. (b) *ICCtc*.

Figure 3.19: Correction of covering for acute angles for *ICC*.

3.4.3 Tip Covering Correction

The types of coverings that allow a positive penetration depth in the piece covering, such as *PCC* and *ICC*, require a correction of the covering for their narrow tips. Since the algorithm computes the circles positions based on the intersection between the circles and the outlines created by the excess (protrusion) distance and penetration depth, when the outline of the penetration depth meets the bone, the algorithms stops. Beyond this point, to place circles, the penetration depth needs to be reduced, considering the diminishing distance from the bone to the outline created by the protrusion distance. The tip covering correction is used to deal with this problem, by adding additional circles to cover the tips of the pieces with very acute angles. This problem only exists in the *ICC* and *PCC*. An example of an area of a layout where this effect appears can be seen in Fig. 3.18.

In order to correct this, several circles with constant radius are placed until their distance to the tip vertex is within the value of the penetration depth. The correction to this problem can be seen in Fig. 3.19. Using circles that would reduce their size proportionally would cause an exponential growth on the number of circles, since their covering radius would be reduced at each iteration, resulting in a much greater number of circles. This effect is much more noticeable when the angle of the tip is very acute. Considering this correction for very acute tips of the outline of the pieces, we can distinguish between five types of coverings: the *CCC*, *PCC*, *ICC*, *PCC* with tip covering (*PCCtc*) and *ICC* with tip covering (*ICCtc*).

Table 3.3: Number of circles generated by each covering with 0.05 TH.

	Total # of circles for piece coverings				
Piece	*CCC*	*PCC*	*ICC*	*PCCtc*	*ICCtc*
*P*1	33	31	31	31	31
*P*2	73	73	73	73	74
*P*3	25	25	25	25	25
*P*4	60	60	59	60	64

Table 3.4: Area of circles generated by each covering with 0.05 TH.

	Total area of circles for piece coverings					Piece
Piece	*CCC*	*PCC*	*ICC*	*PCCtc*	*ICCtc*	Area
P1	41.06	40.27	39.47	40.27	39.47	40.0
P2	116.99	114.92	112.83	114.92	112.84	114.0
P3	49.54	48.82	48.07	48.82	48.07	48.5
P4	40.58	39.43	38.29	39.43	38.32	39.0

3.4.4 Results and Discussion

The different types of coverings are oriented to enable dealing with problems with specific characteristics. In order to estimate the impact that each type of covering has on the quality of approximation of a piece, and on the potential computational cost, a set of four pieces with different characteristics, denoted as *P*1, *P*2, *P*3 and *P*4, are used. Since it is not yet possible to determine the impact of the coverings on the layout solutions, regarding their computational efficiency, the number of circles produced by the set pieces is analyzed, and presented in Table 3.3.

Table 3.4 shows the difference between the area of the circle coverings and the polygonal area of the piece. The values presented in Table 3.3 and Table 3.4 show that no significant differences exist between the number of circles generated from each covering, nor their area, therefore the computational efficiency of the circle coverings should be similar.

The simplifications to the piece outline and the piece skeleton, and also the enhancements done to the covering generated by the algorithm all have an impact on the approximation quality of the covering, but they cannot be considered independently since the simplifications on one of them will impact the others. This is due to the sequential application of each one of these simplification methods. The piece outline returns the piece skeleton, which is then used to produce a circle covering that is then modified and improved. Simplifications in all these steps modify the approximation error to the original outline of the piece.

In order to estimate the impact that each simplification has on the total number of circles, each simplification is analyzed independently of the others. Table 3.5 shows the simplification of the outline of piece 10 from instance *swim*, as seen in Fig. 3.20. The algorithm was defined to eliminate adjacent edges considering the value of their angle, with preference to collinear, or nearly collinear adjacent edges, instead of defining a number of vertices to be removed. Since this

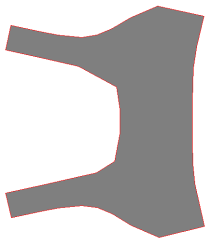


Figure 3.20: Piece 10 from instance *swim*.

Table 3.5: Outline simplification of piece 10, instance *swim* (piece area reference value: *1107.0*).

Smp. Angle (^{o})	Polygonal representation			*CCC* representation		
	Vertices (#)	Area (10^3)	Area (%)	Circles (#)	Area (10^3)	Area (%)
0	36	1107.0	100.0	58	1405.4	127.0
5	29	1115.1	100.7	45	1405.9	127.1
10	23	1155.0	104.3	33	1427.8	129.0
15	21	1170.9	105.8	31	1438.4	129.9
20	16	1242.7	112.3	24	1508.3	136.3
25	13	1258.4	113.7	21	1520.6	137.4
30	12	1347.1	121.7	20	1602.3	144.8
35	10	1402.6	126.7	17	1652.9	149.3
70	9	1588.2	143.5	14	1829.3	165.3
110	8	1956.9	176.8	8	2161.5	195.3

piece has many adjacent edges that are nearly collinear, erasing those edges allowed the biggest reduction in the number of vertices, with a small increase in the area of the piece. This small adjustment to this piece also had a great impact on the reduction of the number of circles but without showing a decrease in the circle covering area. Using the outline simplification in pieces that have this complexity (many near collinear adjacent edges) can substantially reduce the number of circles, without significant impact on the quality of the circle covering. Table 3.5 also shows that as the simplification process advances, a lesser number of edges are removed, but with higher increases of the piece area and circle covering area. The cost of reducing the number of circles at each step grows, returning significant diminishing returns for the reduction in collinear edges. In this case, we accept values for angles that lead a to a difference in the covering area by about 3%.

The process of the simplification of the skeleton can be seen in Table 3.6. The Bone Distances (BDist) parameter defines the bones which will be erased, by imposing a minimum bone length. This is done by connecting their endpoints together, at the middle of the erased bone. The initial skeleton used in this table was obtained from the original outline of the piece, without simplifications. Since the piece has a complex outline, with many edges that are nearly collinear and a high number of vertices, the skeleton that is produced has a high number of bones. As shown in

Table 3.6: Piece skeleton simplification, piece 10, instance *swim* (piece area reference value: *1107.0*).

Bone distance	Bones (#)	Circle covering representation		
		Circles (#)	Area (10^3)	Area (%)
0	113	58	1405.5	127.0
25	93	38	1425.9	128.8
50	83	28	1470.2	132.8
75	78	23	1498.6	135.4
100	74	21	1573.8	142.2
125	71	17	1676.4	151.4
150	67	17	1773.0	160.2

Table 3.7: Circle merging, piece 10, instance *swim* (piece area reference value: *1107.0*).

Circle distance	Circle covering representation		
	Circles (#)	Area (10^3)	Area (%)
0	58	1405.5	127.0
25	39	1437.5	129.9
50	35	1466.3	132.5
75	34	1503.9	135.9
100	33	1568.8	141.7
125	31	1595.3	144.1
150	31	1604.2	144.9

Table 3.6, the elimination of the smallest bones reduces the number of circles by a large amount without a substantial increase in the covering area. As bigger and bigger bones are eliminated the reduction in the number of circles becomes more difficult without reducing significantly the approximation quality to the original piece. Considering bone distance, we accept values for it that lead to a difference in the covering area by about 3%.

The reduction in the number of circles by merging the ones that are nearly contained inside another can be seen in Table 3.7. Pairs of circles are replaced by a MEC if the distance between their centers plus their radius is smaller than a defined value, defined as Circle Distance (CDist). As this value increases, more circles are replaced by MECs at the expense of a larger covering area. Small CDist values are particularly beneficial in cases where the coverings have nearly identical circles almost fully overlapping. Regarding the value for the circle distance, we consider an acceptable value if it reduced the CC representation area up to 3%.

The simplifications to the piece outline, skeleton bones and circle covering have a significant impact on the number of circles, at the cost of a lower quality of circle covering. Table 3.8 shows the effect that these simplifications have when they are all used sequentially. Finding the right combination of parameters can have a significant impact in the reduction of the circle number.

Table 3.8: Combination of simplifications, piece 10, instance *swim* (piece area reference value: *1107.0*).

Angle (^{o})	Bone distance	Circle distance	Circle covering representation		
			Circles (#)	Area (10^3)	Area (%)
0	0	0	58	1405.5	127.0
0	25	25	30	1446.3	130.7
5	25	25	27	1434.3	129.6
5	50	25	23	1454.2	131.4
10	25	25	24	1440.8	130.1

For the currently used piece, the best configuration was achieved by using a very low degree of simplifications. This configuration allowed reducing the number of circles from 58 to 27 at the expense of an increase of about 2.5% in the the additional covering area. Taking into account the combination of all simplifications approaches, our accepted value for the resulting difference regarding the CC area is also 3%.

3.5 High and Low Resolution Coverings

A high number of pieces together with high number of circles can make the Nesting problem with continuous rotations very difficult to solve within a reasonable time. For this reason, reducing the quality of the approximation can reduce the number of circles, up to a certain point, where additional reductions in quality have diminishing returns in the number of circles. In order to analyze the impact of the different covering types and the effect that different resolutions have on the number of circles, two different resolutions (*LR* and *HR*) and pieces from two nesting instances (*poly1a* and *jakobs1*) were selected. The selection of these instances is due to their geometry. Pieces from instance *jakobs1* have many concavities and square angles, while pieces from instance *poly1a* are composed by mixed types of pieces, mostly convex, some being small, others big.

The difference between *LR* and *HR* is the value of the threshold that controls the approximation error of the circle covering. The value used for the *LR* threshold in Table 3.9 is 0.25 units and for the *HR* threshold is 0.10 units. These two levels of approximation, *LR* and *HR*, were selected by choosing values from where the effect of diminishing returns starts to become noticeable. Below a certain point, reducing the quality of the approximation will hardly reduce the number of circles, and beyond another point, increasing the number of circles will not increase significantly the approximation quality of the piece. This trade-off can be seen in Fig. 3.21, where the points *A* and *B* are considered the points where the effect of diminishing returns are noticeable, regarding the quality of approximation, and the number of circles.

The initial data about the several types of covering for the selected instances, is presented in Table 3.9. This table shows the number of circles of *HR* coverings, for all the pieces of each

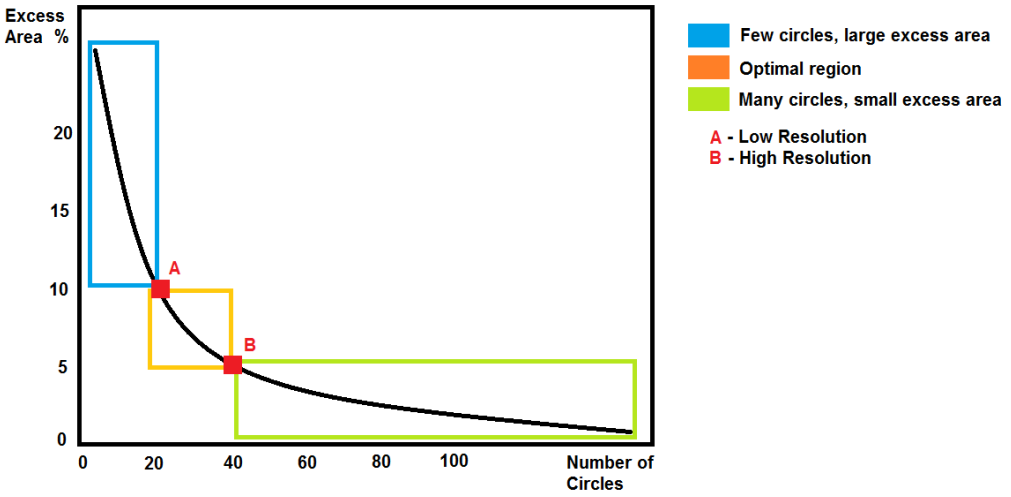


Figure 3.21: Three regions of the trade-off between the quality of the CC representation and the total number of produced circles.

Table 3.9: CC used for computational experiments.

		Total # of circles for circle coverings				
Instance	Resolution	*CCC*	*PCC*	*ICC*	*PCCtc*	*ICCtc*
poly1a	*LR*	154	149	145	149	155
	HR	260	254	250	254	261
jakobs1	*LR*	208	202	198	202	198
	HR	368	368	340	368	340

instance. It can be seen that, usually, when the complexity increases (with higher number of vertices), the total number of circles required to represent all pieces also increases. The different coverings do not have significant variation among them in the total number of circles. However, that variation is more easily noticeable between the representations with different levels of approximation, i.e. between high and low resolutions. The selected points for *LR* and *HR* show a difference in the number of circles of about 68% for *poly1a* instance and a 71% difference for the *jakobs1* instance.

3.6 Concluding Remarks

The approaches that were presented in this chapter can be used to accurately represent pieces, with a controlled approximation error and number of circles, that take into account the requirements of the Nesting problem with continuous rotations. The first approach to this problem, using an Hierarchical Circle Covering has shown to be able to achieve a very good approximation to the pieces, while completely enclosing them in the set of circles, but with a very large number of circles. In

order to achieve a reasonable approximation error to the covering of the piece, the number of circles is so large that the application that uses these coverings will have a prohibitive computational cost. However, this approach also enables producing several distinct covering approximations, that when organized into an hierarchical structure, allows using an overlap exclusion mechanism with more efficiency.

The second approach successfully achieved a circle covering representation, with a controllable approximation error to the piece outline, and a much reduced number of circles (when compared to the hierarchical Circle Covering). Deriving the Medial Axis of a given polygon and using an iterative algorithm to compute the placement positions of the circles on the Medial Axis has shown to produce circle coverings with a very low number of circles, and with very good quality of representation. Several problems were identified, such as the difficulty in generating accurately the Medial Axis of very complex pieces, due to the numerical precision errors that arise. The circle placement algorithm was extended to support additional types of circle representation, producing two additional types of circle coverings defined as Partial Circle Covering and Inner Circle Covering. These circle coverings were aimed at problems with characteristics than could not be properly addressed by the Complete Circle Covering. The Inner Circle Covering allowed addressing Nesting problems that produced layouts with perfect fits, and the Partial Circle Covering was able to reduce the excessive covering of the Complete Circle Covering at the expense of allowing pieces to have overlaps up to a certain penetration depth. While the Partial Circle Covering is also unable to ensure feasible layouts, it may produce tighter layouts than the Complete Circle Covering (less protrusion distance) and with less overlap (considering the polygonal representation of the pieces) than the Inner Circle Covering (due to less penetration depth). These three types of covering enable addressing a great variety of problems with different characteristics.

The *kCC-MA* algorithm that generates Partial Circle Covering and Inner Circle Covering has shown to have some limitations, regarding the approximation quality of the pieces and the total number of circles. Since the final Circle Covering representation is always an approximation to the polygonal representation of the piece, the *kCC-MA* approach was extended to include simplifications that could lead to a reduced number of circles, without affecting significantly the approximation quality of a piece. These simplifications focused on the polygonal representation of the piece, the Medial Axis skeleton that it generated, and the final circle covering produced. The computational experiments shown that just a small degree of simplification leads to a significant reduction in the amount of circles, which can be explained due to the sequential procedure in generating the circle covering. The simplifications in the polygonal outline (smoothing of the outline of the piece) generated simpler Medial Axis skeletons (lower number of skeleton segments), which when also simplified (by eliminating the smallest ones, and connecting the nearest), generated a circle covering with less circles. The simplification of the circle covering replaced pairs of circles by a single circle when the approximation error would not increase more than a certain specified value. All these simplifications managed to reduce the number of circles required to represent a piece by circle covering, while maintaining a desired approximation level. Another limitation of the *kCC-MA* algorithm that was addressed deals with the difficulty in covering narrow tips of pieces due

to their very acute angles. The algorithm was extended to allow using circles with equal radius to cover the bone until the tip is fully covered, or force the covering of circles to continue until the uncovered distance to the tip is equal or lower than the penetration depth (at a cost of a significant increase in the number of circles, depending on the number of tips to be covered).

All these developments allow addressing the geometrical component of the Nesting problems by producing coverings with controllable approximation error, and total number of circles produced. This approach allows adjusting the quality of representation depending on the computational cost that is acceptable, where if the solution approach is very computationally efficient, a higher quality circle covering can be used, while having a solution method with low computational efficiency may require a reduced quality circle covering representation.

The computational experiments that were presented in this chapter only focus on the quality of approximation of the coverings and their total number of circles. The impact that the different resolutions and types of covering have on the resolution of the Nesting problem will be analyzed in Chapter 4 after the introduction of the Non-Linear Mathematical Models and their Solution Approaches. The approaches presented in this chapter can be improved by exploring multiple paths. These will be introduced and discussed in Chapter 6.

Some of the contents presented in this chapter can also be found in (Rocha et al., 2013a). It contains the approach used to tackle the geometric representation of pieces, using Circle Covering with Medial Axis, for the Nesting problem with continuous rotations. It describes the constructive algorithm used, showing possible covering types and also comparing to other approaches used to achieve piece representations through circles.

Chapter 4

Non-Linear Programming Approach for the Irregular Shapes Placement Problem

The requirements imposed by the Irregular Shapes Placement problem (i.e. Nesting problem) with continuous rotations, require that the pieces must be placed inside a container in a non-overlapping configuration. A Circle Covering representation was selected to represent geometrically the pieces, due to its intrinsic capability in dealing with continuous rotations. Using a Circle Covering representation, naturally leads to the usage of a Non-Linear Programming formulation for the Nesting problem. This is due to the non-overlapping constraints between circles, which are described by quadratic non-linear equations, and to deal with trigonometric functions. Among the different variants of the Nesting problem, the Irregular Shapes Strip Packing problem was selected. This problem requires a set of irregular polygons (i.e., simple non-convex polygons) to be placed without overlap inside a rectangular container with a given width and infinite length, where the objective is to minimize the strip length.

This chapter focuses on solution approaches based on Non-Linear Programming models to tackle the Nesting problem with continuous rotations. The relations between the geometrical representation and the formulation of the model are explored, by developing alternative NLP formulations, that enable evaluating and assessing the Circle Covering Representation presented in the previous chapter. Finding a model formulation that generates good quality solutions in a reasonable amount of time, when used with Circle Covering representation may enable more complex problems to be tackled efficiently.

The first section of this chapter discusses two alternative formulations to tackle the Irregular Shapes Strip Packing problem. One model is based on considering each circle individually and then use constraints to force the relative positions between circles from the same piece, and the

other one on considering each piece as a whole. In the second section, different types of constraints aggregations are explored in order to improve the overall efficiency, which leads to several model variants. Section three introduces a post-optimization approach needed to correct infeasible solutions that may be obtained when using inner or partial circle coverings. The fourth section covers the computational experiments performed to compare and evaluate the circle coverings, model formulations and their variants. The last section concludes the chapter with a set of final remarks and comments.

4.1 Non-Linear Mathematical Model

Non-Linear Programming models can be used to formulate mathematically the Nesting problem, taking into account the representation of the pieces by circles and the container outline, including the side that is to be minimized. Through the use of NLP models, two different mathematical models were formulated, one which has its main focus on the circles of the piece representation and the other which has its focus on the pieces (each piece described by a set of circles). In order to be able to represent the concept of pieces, using circles, a piece must be defined as a collection of circles with fixed relative positions between each other.

Both NLP models are defined through a set of variables, constraints and objective function, which conceptually are very similar. Considering the model that is based on circles, each piece is defined indirectly through the variables assigned to its circles, while considering the model based on pieces, each piece is defined through the variables assigned to its reference point. The variables are continuous, where some define the orthogonal position (i.e., the layout position) and others the orientation of the pieces. Both models require constraints that prevent the overlap between pieces (*Non-Overlapping Constraints*), and that maintain the pieces inside the container (*Containment Constraints*). Due to different formulations, only the model based on circles requires the use of a third type of constraints that maintain the relative positions between the circles of a piece (*Piece Integrity Constraints*).

The objective function focuses on the minimization of the container length, being the same for both mathematical models. This function is designed to minimize the distance of the circles, plus their individual radius, to the minimum position in the orthogonal dimension that represents the layout length.

4.1.1 Model based on Circles

The Circle Covering representation, when used to solve the Nesting problem with continuous rotations, naturally leads to a mathematical model where the main components are the circles. This model based on circles (M1) considers each circle as an independent structure, with its own set of assigned variables. This model has two translation variables (x, y) for each circle, a rotation variable for each piece and an additional variable to keep track of the maximum x coordinate (i.e., the layout length). Concerning the constraints, it needs to have a *Non-Overlapping Constraint* for each pair of circles belonging to different pieces, four *Containment Constraints* for each circle

$$\text{minimize} \quad l \tag{4.1}$$

$$\text{subject to:} \quad (x_{k_i} - x_{h_j})^2 + (y_{k_i} - y_{h_j})^2 \geq (R_{k_i} + R_{h_j})^2, \quad \forall i \in C_k,\ \forall j \in C_h,\ \forall k,h \in \mathbb{N}, k \neq h \tag{4.2}$$

$$x_{k_i} + R_{k_i} - l \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} \tag{4.3}$$

$$-x_{k_i} + R_{k_i} \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} \tag{4.4}$$

$$y_{k_i} + R_{k_i} - Wd \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} \tag{4.5}$$

$$-y_{k_i} + R_{k_i} \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} \tag{4.6}$$

$$x_{k_i} - x_{k_0} = \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}}, \qquad \forall i \in C_k \wedge i \neq 0,\ \forall k \in \mathbb{N} \tag{4.7}$$

$$y_{k_i} - y_{k_0} = \sin(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}}, \qquad \forall i \in C_k \wedge i \neq 0,\ \forall k \in \mathbb{N} \tag{4.8}$$

$$x_{k_i}, y_{k_i}, \theta_k, l \in \mathbb{R} \tag{4.9}$$

Figure 4.1: Mathematical Model based on Circles (M1)

(one constraint per side of the strip, including the length) and, for each piece, one *Piece Integrity Constraint* for each circle except the circle that is used as the reference point of the piece. The number of *Non-Overlapping Constraints* has a factorial growth with the number of circles from different pieces, while the number of *Containment* and *Piece Integrity Constraints* grow linearly with the total number of circles. The complete mathematical model M1 is presented in Fig. 4.1.

The NLP model based on circles consists of a formulation based on packing circles that are aggregated into distinct sets, representing pieces. For each circle, two variables are defined: x_{k_i} is the variable that defines the position of circle i of piece k on the x-axis, and y_{k_i} is the variable that defines the position of circle i of piece k on the y-axis. Each piece k (i.e., a set of circles) has an associated variable, θ_k, that defines its orientation. Each piece k is composed by a set of C_k circles. So the total number of variables is $2 \times \#\text{Circles} + \#\text{Pieces} + 1$, all of them continuous. The objective function of the model, as seen in Eq. 4.1, is represented by the variable l, corresponding to minimization of the layout length (i.e., reducing the used length of the strip). The *Non-Overlapping Constraints*, defined in equation 4.2, represent a comparison of the distance between each pair of circles i, j from pieces k, h and the sum of the radius of both circles R_{k_i} and R_{h_j}. The type of these constraints is quadratic. The *Containment Constraints*, defined in equations 4.3-4.6 ensure that each circle do not exceed the admissible placement area. The type of these constraints is linear. Equations 4.7-4.8 are used to maintain the relative positions between circles of the same piece, ensuring the piece structural integrity. This means that when the piece is translated or rotated, the relative positions between all circles from the same piece are maintained. The position of each circle i is fixed relative to the circle 0 of the piece k. Constants $A_{k_{0,i}}$ and $D_{k_{0,i}}$ stand for, respectively, the angle and distance between the centers of circle i and circle 0 of piece k. Constant Wd, in equation 4.5, defines the width of the strip. *Piece Integrity constraints* are non-linear constraints due to the presence of the trigonometric functions. Finally, equation 4.9 defines the domains of the variables.

The M1 model is a constrained non-linear programming model with multiple local minima. Models of this type are known to be very hard to solve to optimality, as (Jones, 2013) has shown recently on a similar problem. Furthermore, the complexity of the model increases both with the number of pieces and the number of circles, specially the non-overlapping constraints that have a factorial growth on number of circles.

4.1.2 Model based on Pieces

An alternative way to model the Nesting problem with continuous rotations is to base it on the pieces instead of the circles. This idea led to the development of model M2, where placement and rotation variables are defined for the reference point of each piece. Naturally, this model does not require *Piece Integrity Constraints*, since the circle positions are described directly in the *Non-Overlapping and Containment Constraints*. The *Non-Overlapping and Containment Constraints* for model M2 are obtained by transforming the equivalent constraints in model M1 by replacing the circle placement variables (x_{k_i} and y_{k_i}) with the equivalent expressions obtained from the *Piece Integrity Constraints* (equations 4.7-4.8), which are equality constraints.

In model M2 each piece has three variables assigned to it, being two of them the placement position on the layout x and y, and the third one the orientation of the piece θ. This formulation reduces both the number of variables and constraints. The number of variables is $3 \times \#\text{Pieces} + 1$, where the additional variable keeps track of the layout length. The number of variables grows linearly with the number of pieces. Regarding the constraints, since the variables are assigned to the reference point of the piece, the positions of the circles that define the piece must be computed from the reference point. This requirement increases the complexity of the constraints that compute the non-overlapping between the pieces and their containment inside the layout outline. The number of *Non-Overlapping and Containment Constraints* are the same as in model M1, while *Piece Integrity Constraints* are discarded. The *Non-Overlapping Constraints* still have a factorial growth and the *Containment Constraints* a linear growth. The complete mathematical model M2 is presented in Fig. 4.2.

The *Objective Function*, as seen in equation 4.10, is represented by the variable l and is the same as the Model M1 where the aim is to minimize the length of the strip. Each piece k is composed by a set of C_k circles. For each piece, three variables are defined: x_k is a variable that defines the position of the reference point of piece k on the x-axis, y_k is a variable that defines the position of the reference point of piece k on the y-axis and θ_k is a variable that defines the orientation of piece k, around its reference point. The *Non-Overlapping Constraints*, equations 4.11, are a comparison of the distance between each pair of circles i, j relative to the reference point of each piece k, h and the radius of both circles R_{k_i} and R_{h_j}. The *Containment Constraints*, equations sets 4.12-4.15, ensure that the positions of each circle, obtained from the reference point x and y, do not exceed the admissible placement region. Finally, the domain of the variables are defined in equation 4.16. Constants $A_{k_{0,i}}$ represent the initial angle between the center of circle i of piece k and the reference point of the same piece where the constant $D_{k_{0,i}}$ represents their distance. The constant Wd defines the width of the strip.

$$\text{minimize} \quad l \tag{4.10}$$

$$\begin{aligned}
\text{subject to:} \quad & (R_{k_i} + R_{h_j})^2 - (x_k + \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} - x_h - \\
& - \cos(A_{h_{0,j}} + \theta_h) \times D_{h_{0,j}})^2 - (y_k + \sin(A_{k_{0,i}} + \theta_k) \times \\
& \times D_{k_{0,i}} - y_h - \sin(A_{h_{0,j}} + \theta_h) \times D_{h_{0,j}})^2 \leq 0, \\
& \qquad \forall i \in C_k,\ \forall j \in C_h,\ \forall k,h \in \mathbb{N}, k \neq h && (4.11) \\
& x_k + \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} + R_k - l \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} && (4.12) \\
& R_k - x_k - \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} && (4.13) \\
& y_k + \sin(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} + R_k - Wd \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} && (4.14) \\
& R_k - y_k - \sin(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} \leq 0, \qquad \forall i \in C_k,\ \forall k \in \mathbb{N} && (4.15) \\
& x_k, y_k, \theta_k, l \in R && (4.16)
\end{aligned}$$

Figure 4.2: Mathematical Model based on Pieces (M2)

Nevertheless, like the previous model, model M2 is also a constrained non-linear programming model with multiple local minimums, and also very hard to solve to optimality. The complexity of model M2 also increases both with the number of pieces and the number of circles, specially the non-overlapping constraints that has a factorial growth on number of circles. The most important benefits of M2 formulation is the reduction in the number of variables and constraints. However, this benefit comes with a price: an increased complexity of the *Non-Overlapping and Containment Constraints*. This increased complexity is due to the presence of trigonometric functions (sin and cos) needed to compute the positions of each circle.

4.2 Constraints Aggregation

Both NLP models presented in the previous section, M1 and M2, have difficulties in tackling large instances due to the exponential increase in the number of *Non-Overlapping Constraints*, as the number of pieces and circles increases. Real-world instances can have a large number of pieces, up to a few hundreds, and require a large number of circles to achieve good quality approximations. So, in order to tackle real world instances the models proposed in the previous section need further improvements of their computational efficiency.

One possibility to reduce the computational cost is to avoid the computation of *Non-Overlapping Constraints* between circles from pieces that are far apart in the layout. This can be achieved through the use of an hierarchical overlap computation, to determine which pairs of pieces really require comparison. Another possibility is to aggregate all *Non-Overlapping Constraints* between circles from a pair of pieces in a single summation constraint. This aggregation can be further extended to aggregate all *Non-Overlapping Constraints* in a single non-overlapping constraint and also to aggregate other types of constraints in a single constraint. This way, the number of

constraints is reduced to a maximum of three, one constraint of each type. The aggregation is particularly efficient when used together with the hierarchical overlap method. Another benefit of reducing multiple constraints to a single one, even without the hierarchical overlap method, is to reduce the internal computational cost of the solver when compared to the individual computation of different constraints.

In the following subsections the details on how to aggregate each type of constraints are presented, followed by the details on the hierarchical overlap method used. Combining the different aggregations types leads to different model variants derived from models M1 and M2. This variants are presented in the last subsection.

4.2.1 Aggregating Non-Overlapping Constraints

The *Non-Overlapping Constraints* of model M1, as defined in equation 4.2, are now aggregated in a single summation expression that returns the sum of all overlaps between each pair of circles belonging to different pieces (equation 4.17). Each summation term represents the difference between the squared sum of the radius of both circles R_{k_i} and R_{h_j} and the squared distance between each pair of circles centers i, j from pieces k, h (equation 4.18). Since the *Non-Overlapping Constraints* are to be minimized, and being valid if zero or negative, one component of the constraint with negative value could cancel another component with positive value. For this reason, components that return values below zero are ignored, by considering the maximum between the value of $\text{NOVLP\_M1}_{i,j,k,h}$ and zero. This makes the function become non-differentiable, which is solved by squaring the whole component, after verification and correction for negative values.

$$\sum_{i=0}^{C_k}\sum_{j=0}^{C_h}\sum_{k=0}^{N}\sum_{\substack{h=0\\k\neq h}}^{N}\left[\max(0,\text{NOVLP\_M1}_{i,j,k,h})\right]^2 \leq 0 \tag{4.17}$$

$$\text{NOVLP\_M1}_{i,j,k,h} = (R_{k_i}+R_{h_j})^2 - [(x_{k_i}-x_{h_j})^2+(y_{k_i}-y_{h_j})^2] \tag{4.18}$$

The same idea is applied to the *Non-Overlapping Constraints* of model M2 (equation 4.11), given a single summation expression (equation 4.19). Again, each summation term represents the difference between the squared sum of the radius of both circles R_{k_i} and R_{h_j} (equation 4.20) and the squared distance between each pair of circles centers i, j from pieces k, h (equations 4.21-4.22). The same mechanisms are used to avoid negative values and to make the function differentiable.

$$\sum_{i=0}^{C_k}\sum_{j=0}^{C_h}\sum_{k=0}^{N}\sum_{\substack{h=0\\k\neq h}}^{N}\left\{\max\left[0,\text{NOVLP\_R}_{i,j,k,h}-(\text{NOVLP\_X}_{i,j,k,h}+\text{NOVLP\_Y}_{i,j,k,h})\right]\right\}^2 \leq 0 \tag{4.19}$$

$$\text{NOVLP\_R}_{i,j,k,h} = (R_{k_i} + R_{h_j})^2 \tag{4.20}$$

$$\text{NOVLP\_X}_{i,j,k,h} = [x_k + \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} - x_h - \cos(A_{h_{0,j}} + \theta_h) \times D_{h_{0,j}}]^2 \tag{4.21}$$

$$\text{NOVLP\_Y}_{i,j,k,h} = [y_k + \sin(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} - y_h - \sin(A_{h_{0,j}} + \theta_h) \times D_{h_{0,j}}]^2 \tag{4.22}$$

The main advantage in aggregating *Non-Overlapping Constraints* is to discard the computation of a large set of summation terms when pieces are far apart. In fact, the terms are not discarded but replaced by 0, since we already know in advance that equation 4.18 will be negative. The hierarchical overlap method, which will be presented in section 4.2.4 is used to detect these situations.

Aggregating *Non-Overlapping Constraints* becomes more effective as the number of pieces to place increases, since it allows the hierarchical overlap method to discard more summation terms as pieces are naturally more spread. Also, when the number of pieces increases, the corresponding number of summation terms (and the number of *Non-Overlapping Constraints*) has a factorial growth.

4.2.2 Aggregating Containment Constraints

Aggregating *Containment Constraints* follows the same general idea of aggregating *Non-Overlapping Constraints*. So, the *Containment Constraints* of model M1, defined in equations sets 4.3-4.6, are now aggregated in a single summation expression that returns the sum of distances that protrude the container (equation 4.23). This equation has four terms (equations 4.24-4.27), one for each of the four container sides (including one for the layout length). The same mechanisms are used to avoid negative values and to make the function differentiable.

$$\sum_{i=0}^{C_k} \sum_{k=0}^{N} \{[\max(0, \text{CNTM\_XMAX}_{i,k})]^2 + [\max(0, \text{CNTM\_XMIN}_{i,k})]^2 + \\ + [\max(0, \text{CNTM\_YMAX}_{i,k})]^2 + [\max(0, \text{CNTM\_YMAX}_{i,k})]^2\} \leq 0, \tag{4.23}$$

$$\text{CNTM\_XMAX}_{i,k} = (R_{k_i} + x_{k_i} - l), \tag{4.24}$$

$$\text{CNTM\_XMIN}_{i,k} = (R_{k_i} - x_{k_i}), \tag{4.25}$$

$$\text{CNTM\_YMAX}_{i,k} = (R_{k_i} + y_{k_i} - Wd), \tag{4.26}$$

$$\text{CNTM\_YMIN}_{i,k} = (R_{k_i} - y_{k_i}), \tag{4.27}$$

Model M2 follows the same scheme, by aggregating constraints sets 4.12-4.15 in a single summation expression (equation 4.23). The same four terms appear (equations 4.29-4.32) and it is also necessary to use the same mechanisms to avoid negative values and to make the function differentiable.

$$\sum_{i=0}^{C_k}\sum_{k=0}^{N}[\max(0,\text{CNTM\_XMAX}_{i,k})]^2+[\max(0,\text{CNTM\_XMIN}_{i,k})]^2+ \\ +[\max(0,\text{CNTM\_YMAX}_{i,k})]^2+[\max(0,\text{CNTM\_YMIN}_{i,k})]^2\}\leq 0, \tag{4.28}$$

$$\text{CNTM\_XMAX}_{i,k}=(x_k+\cos(A_{k_{0,i}}+\theta_k)\times D_{k_{0,i}}+R_k-l), \tag{4.29}$$
$$\text{CNTM\_XMIN}_{i,k}=(R_k-x_k-\cos(A_{k_{0,i}}+\theta_k)\times D_{k_{0,i}}), \tag{4.30}$$
$$\text{CNTM\_YMAX}_{i,k}=(y_k+\sin(A_{k_{0,i}}+\theta_k)\times D_{k_{0,i}}+R_k-Wd), \tag{4.31}$$
$$\text{CNTM\_YMIN}_{i,k}=(R_k-y_k-\sin(A_{k_{0,i}}+\theta_k)\times D_{k_{0,i}}), \tag{4.32}$$

Containment Constraints aggregations are used without any hierarchical overlap method in both models, which means that all the terms are always computed. This option was taken because the expressions in this case are easier to compute when compared to the non-overlapping constraints case. Additionally, the number of terms only increases linearly with the number of circles.

4.2.3 Aggregating Piece Integrity Constraints

Finally, one comes to the aggregation of the *Piece Integrity Constraints*. These constraints only appear in model M1 and are intrinsically different from the previous constraints, since they are equality constraints (equations sets 4.7-4.8) instead of inequality ones. Being equality constraints, it is necessary to considerer both positive and negative deviations and to avoid positive deviations cancellations due to negative deviations, and *vice-versa*. This is achieved by only squaring each summation term, where equation 4.34 measures x deviations and equation 4.35 measures y deviations.

$$\sum_{i=0}^{C_k}\sum_{j=0}^{C_h}\sum_{k=0}^{N}\sum_{\substack{h=0\\k\neq h}}^{N}\{\text{PINT\_X}_{i,j,k,h}+\text{PINT\_Y}_{i,j,k,h}\}\leq 0 \tag{4.33}$$

$$\text{PINT\_X}_{i,j,k,h}=(x_{k_i}-x_{h_j}-\cos(A_{k_{0,i}}+\theta_k)\times D_{k_{0,i}})^2 \tag{4.34}$$
$$\text{PINT\_Y}_{i,j,k,h}=(x_{k_i}-x_{h_j}-\sin(A_{k_{0,i}}+\theta_k)\times D_{k_{0,i}})^2 \tag{4.35}$$

Unlike in the previous constraint aggregations, using hierarchical overlap methods does not make sense when aggregating *Piece Integrity Constraints* since the constraints only deal with circles of the same piece.

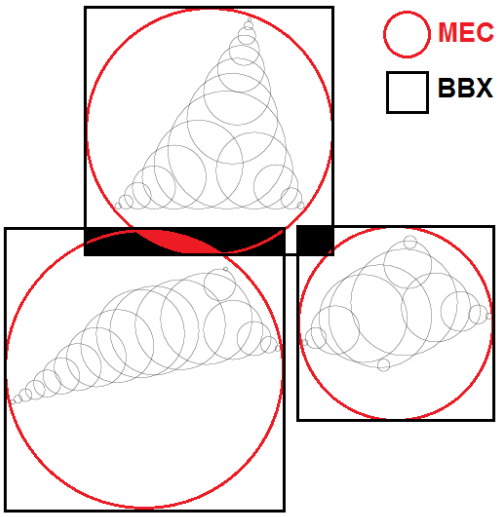


Figure 4.3: Hierarchical Overlap example.

4.2.4 Hierarchical Overlap Method

The hierarchical overlap method is used together with the *Non-Overlapping Constraints* aggregation of both models. This method consists in discarding overlap computation between distant pieces. It uses a hierarchical approach to the verification of overlaps, defined as a Bounding Volume Hierarchy, with a tree structure, starting with a basic approximation to the piece (bounding box), and using more refined representations until reaching the most accurate, based on circle covering. In this hierarchical overlap method the overlapping between two pieces can be defined as not occurring when any of their representations is not overlapping.

Pieces are defined with three levels of representation in the hierarchical structure, where the first level contains its bounding box, the second level a minimum enclosing circle (MEC), and finally the circle covering. The verifications only proceed to the next hierarchical level when a possible overlap situation is detected, as shown in Fig. 4.3. The overlap detection starts by comparing the bounding boxes of each piece, and if positive the overlap detection proceeds to compare the MEC. If the overlap cannot be discarded in the first two levels, the overlap detection uses the Circle Covering of the pieces.

Models M1 and M2 have different behaviors and characteristics that require some modifications to the implementation of the hierarchical overlap verification method. Since the reference point of each piece is not exactly on the center of the piece, but on the center of the closest circle, the representation through bounding box and MEC must take into account the distance between this point and the most distant circle plus radius of the circle covering. Since the rotation of the piece around its reference point generates a circle (considering the radius equal to the most distant point of the circle covering), the MEC will enclose it, and the bounding box will enclose the MEC. This leads to a slight increase in the size of the bounding box and MEC, and also determined that their centers are always on the current position of the piece reference point. The main difficulty in

Table 4.1: Model variants.

Model Variant	Non-overlapping Constraints	Containment Constraints	Piece Integrity Constraints
M1S	No	No	No
M1E	Yes	No	No
M1T	Yes	Yes	Yes
M2S	No	No	—
M2E	Yes	No	—
M2T	Yes	Yes	—

implementing this method lays on the characteristics of model M1, where the positions of the circles are defined through the piece integrity constraints. Considering that the Circle Covering must always be contained by the MEC, the problem arises when the solver is running the model, since the solver method allows constraints to be infeasible during its operation, some of those being the *Piece Integrity Constraints*, causing circles of a given covering to lag behind the current placement of the piece, and being outside the MEC. This causes convergence problems for the solver, since the *Non-Overlapping Constraints* start returning false positive and false negative detections. This can be minimized by increasing the size of the bounding box and MEC, but with their increase, the efficiency of the hierarchical overlap method is reduced. Model M2 does not have this type of problem due to the formulation of its constraints, by having the circle positions computed directly from the reference point.

4.2.5 Model Variants

Different models can be created based on these constraint aggregations, by using different combinations of constraint aggregations. The behavior of the model will change, together with its computational efficiency and the quality of solutions. Since the *Non-Overlapping Constraints* has a factorial growth while the *Containment and Piece Integrity Constraints* grow linearly, there is a greater advantage in aggregating the *Non-Overlapping Constraints* together since their decrease will provide a greater impact.

The downside of aggregating constraints is that it can lead to an increase in the sensitivity during the search. This increased sensitivity causes difficulties for the solver to decide the path that minimizes the objective function. Another effect that also needs to be considered is that as the complexity of the constraint increases, the numerical precision error gets larger, thus resulting in cases where the constraint will be considered valid when it is not, and *vice-versa*. Table 4.1 shows the different variants of models M1 and M2 that were considered to solve the Nesting problem with continuous rotations.

All variants starting with M1 are the ones derived from the model based on circles (M1), and models starting with M2 are the ones derived from the model based on pieces (M2). The last letter on the model variant name (S, E and T) indicates the constraint aggregations used in each

variant, where: S means no aggregations used, E only aggregates *Non-Overlapping Constraints*, and T aggregates all constraints types. The aggregations are done by type of constraint, so when all constraints types are aggregated, the total number of constraints is, depending on the model, 3 for variant *M1T* (1 non-overlapping constraint, 1 containment constraint and 1 piece integrity constraint) or 2 for variant *M2T* (1 non-overlapping constraint and 1 containment constraint).

4.3 Layout Post-Optimization

The post-optimization phase is used to address the infeasible layouts created by CC that do not completely cover the pieces, such as the *PCC* and *ICC*, and also to reduce the excess waste derived from *LR* and *HR* coverings when using the *CCC*. The produced results can only be compared to results in the literature, and between different types of covering, if their layouts have no infeasibilities. In order to address the layout infeasibility, the compacted layouts are re-compacted using a *CCC* with an higher resolution than the one used when compacting the layout, such as High Resolution Plus (*HRP*) or Very High Resolution (*VHR*). This is done in order to allow pieces to more easily adjust to each other, since it uses a covering representation closer to the original polygonal outline of the piece. Normally, the resolutions used in solving Nesting instances are the *LR* and *HR*, due to their reasonable number of circles, since solving the same instances with an higher resolution (*HRP* and *VHR*) in a single step is impractical, due to the significant increase in computational cost. The main downside of the post-optimization phase is that it increases the total computational cost required to achieve a feasible solution, while not being able to guarantee that the correction of the layout will be successful. The computational cost increases with an increase in the total number of circles of the used CC.

The use of *CCC* allows avoiding infeasible solutions due to its excess covering of the piece, which is beneficial for certain industrial applications where pieces must be placed with a minimum distance separating them. This is due to technological constraints, related to the cutting precision and also due to cutting process (if cutting with laser or torch, the surrounding area might suffer deformations, which requires pieces to be placed with a minimum distance from the cutting region).

A situation where infeasibilities exist can be seen in Fig. 4.4. The two polygonal outlines overlap each other due to the incomplete covering of the pieces, while the circle covering does not contain overlaps between circles from different pieces. The example shown in this figure has been exaggerated, using a *LR* covering, in order to show the types of infeasible situations that may arise when using *PCC* or *ICC*.

An example of the detection process for infeasibilities can be seen in Fig. 4.5. If the polygonal representation of a piece has any vertex inside another piece, or if any edge of the current piece intersects any edge of another piece, an infeasibility is defined. This example was made with the purpose to explain all cases of infeasibilities that are detected, considering only the polygonal representation.

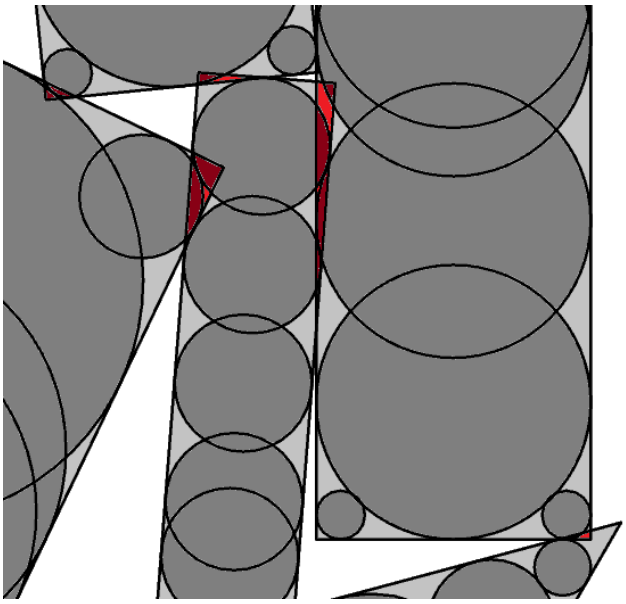


Figure 4.4: Example of the creation of infeasibilities, considering a feasible circle covering.

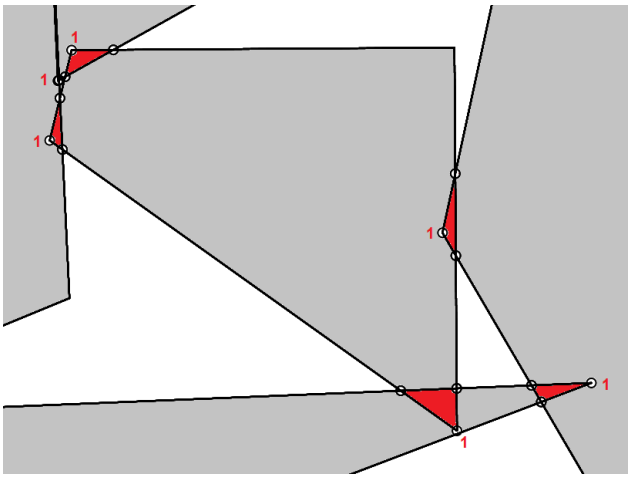


Figure 4.5: Detection of infeasibilities.

Table 4.2: Scaling of *CCC* resolutions.

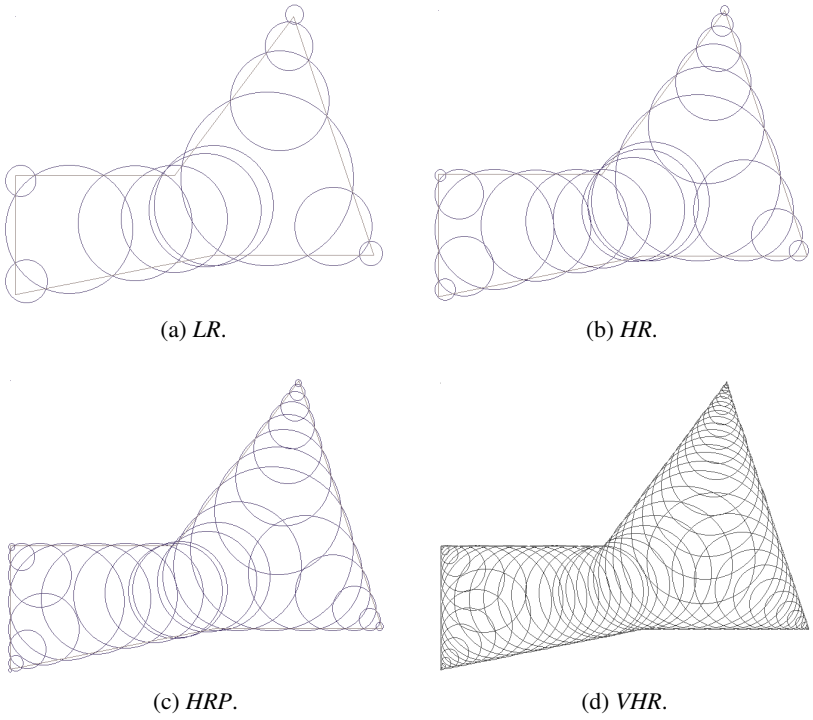
Resolution	Threshold	# Circles	Area	Area %
LR	0.25	13	33.61	117.9
HR	0.10	20	30.41	106.7
HRP	0.05	29	29.44	103.3
VHR	0.01	66	28.68	100.6
Polygonal	0.00	–	28.50	100.0

The differences between the coverings of the various resolutions can be seen in Fig. 4.6 where a piece is covered using *CCC* with *LR*, *HR*, *HRP* and *VHR* resolution coverings. These figures clearly show the increase in quality of representation through circle covering that is achieved by using higher resolutions. The increase in the number of circles and the effect in the reduction of the excess area can be seen in Table 4.2. In this table, the polygonal area of the piece is also presented, to allow comparison with each type of covering. The polygonal representation has no additional area, since the circle covering is derived from it, considering it the ”real" representation of the piece. The benefit of using higher resolutions as a post-optimization method derives from it being applied to a layout that has already been compacted, thus the pieces are not required to move substantially to reach a stable position. This effect is able to compensate the increase in the computational cost derived from the higher number of circles. However, in situations where the overlaps present in the layout cannot be removed the computational cost increases, since the solver will try to solve them. The computational cost is fully wasted on layouts that cannot be corrected. For this reason, this method may be adequated to be applied only to the most promising solutions.

The Post-Optimization may not be able to correct infeasible layouts due to lack of space to remove overlaps, such as when using *ICC* or *PCC* and being limited by the fixed width of the strip. This may also occur when the layouts are so tightly packed that concavities of a piece prevent two pieces that are overlapping from moving in opposite directions. Difficulties such as these are difficult to solve. An alternative to assist in correcting infeasible layouts consists of compacting the pieces using a container with reduced width, by as much as 2%, in order to allow pieces also to adjust in that direction, and later using the Post-Optimization with the full width of the container. This post-optimization approach can also be applied to feasible layouts derived from *CCC*, using *HRP* or *VHR* in order to further improve the packing of the layout.

4.4 Results and Discussion

In this section, the proposed models and algorithms are tested, aiming to solve the Nesting problem with continuous rotations, and having their results presented and discussed. This section focuses on analyzing the behavior and effectiveness of the solution approach based on NLP models together with Circle Covering. Due to the interdependence between the different model variants,



(a) *LR*. (b) *HR*.

(c) *HRP*. (d) *VHR*.

Figure 4.6: Scaling of *CCC* resolutions.

resolution coverings and types of covering, each one of these parameters will be tested independently, while fixing the others, in order to analyze the impact that each has on the final solution, considering solution quality and computational cost.

In the first sub-section, the Nesting instances used in the tests are presented. Not all instances will be used in all tests, due to their characteristics. The most appropriate instances for each test will be selected in order to present the computational results in a clear way. The instances can be separated into three distinct groups, by their size (small, medium and large instances). The small and medium instances are used to test the configuration parameters, such as model type, resolution and covering type, that is used to solve the Nesting problems with continuous rotations. Since this chapter focuses on the definition and evaluation of the NLP model variants, different resolutions and types of covering, it will use only the small and medium size instances. The large instances will be used on the next chapter, where the approaches designed to tackle larger and more complex instances are introduced and explored, in order to verify their efficiency dealing with real world problems. The overall setup configuration is also shown in this sub-section, including the approach that is used to obtain the initial solutions.

The second sub-section evaluates the impact that the different formulations of the mathematical models have solving a Nesting problem with continuous rotations, considering also the variants of each model depending on the type of constraints that are aggregated. This test allows to verify which model and what variants are better suited to deal with specific characteristics of a given instance, considering computational performance and quality of the solution. Since only the mod-

els are being evaluated, the resolution and type of covering were selected with a focus on having reduced computational cost and admissible solutions, thus selecting a low resolution (*LR*) version of the instances, with the *CCC*.

The third sub-section addresses the impact caused by different resolutions, in terms of the quality of layout compaction solution and computational cost. Two different resolutions are tested, one with a lower resolution (*LR*) and the other with a higher resolution (*HR*). The models and types of covering are selected in order to allow comparing the impact of both resolutions, with reduced computational cost, while being able to achieve an admissible solution with good quality.

The fourth sub-section contains the computational experiments that were done in order to select the most appropriate resolution for the post optimization method (that attempts to correct infeasible layout results). A comparison between the impact of different resolutions, regarding quality of solution and computational cost, is done.

The fifth sub-section focuses on the impact on the quality of the layout from the different types of covering: *CCC*, *PCC* and *ICC*. These tests show the impact that each covering have on the quality of the solution, after using the post-optimization method that addresses infeasibilities and improves the compaction quality.

The final sub-section uses the most promising combination of parameters (model and constraint aggregation, resolution and covering) to obtain the best layout compaction solutions. These results are then compared to the current literature results.

4.4.1 Nesting Instances, Setup Configuration and Initial Solutions

A set of Nesting instances were selected from the ESICUP[1] website to evaluate and test the computational efficiency of the proposed models and circle coverings. The selected instances are presented in Table 4.3, where the data is shown to give some sort of measurement about the relative geometric complexity of each instance. The instances *poly1a* and *poly2a* have the least complexity due to their low number of pieces and the reduced number of vertices and concavities. The instance *swim* is, by far, the most complex one, with 960 vertices, 48 pieces and multiple concavities per piece. This is a real-world instance and solving it remains the biggest challenge in this area. It should also be noted that instances *poly2a* to *poly5a* are obtained from instance *poly1a* by replicating it 2, 3, 4 and 5 times the original set of pieces from *poly1a*. Instances *poly2b* to *poly5b* follow a similar scheme of *poly2a* to *poly5a*, but 15 new pieces are added each time, instead of replicating the pieces from *poly1a*.

The instances *poly1a*, *poly2a*, *poly2b* and *jakobs1* are considered small instances, while *poly3a* and *poly3b* are considered medium instances. This is due to the combination of their number of pieces, but also derived from the number of circles used in each Circle Covering. The instances *poly4a*, *poly4b*, *poly5a*, *poly5b* and *swim* are considered large instances, due either to the high number of pieces (more than 50) or the total number of vertices (almost 1000).

[1] EURO Special Interest Group on Cutting and Packing (http://www.fe.up.pt/esicup)

Table 4.3: Geometric characteristics of the Nesting instances used.

Instance	# Pieces	# Vertices	Avg. # Vertices	Total # Conc.	# Pieces with Conc.	Width	Best length [a]
jakobs1	25	150	6.0	22	10	40	11.82
poly1a	15	69	4.6	6	5	40	13.21
poly2a[b]	30	138	4.6	12	10	40	25.71
poly2b	30	148	4.9	19	11	40	29.07
poly3a[b]	45	207	4.6	18	15	40	38.81
poly3b	45	222	4.9	39	17	40	38.79
poly4a[b]	60	276	4.6	24	20	40	52.08
poly4b	60	296	4.9	66	22	40	49.45
poly5a[b]	75	345	4.6	30	25	40	63.81
poly5b	75	363	4.8	96	27	40	58.09
poly10a[b]	150	690	4.6	60	50	40	— [c]
poly20a[b]	300	1360	4.6	120	100	40	— [c]
swim	48	960	20.0	99	42	5752	— [c]
swim4[d]	192	3840	20.0	396	168	5752	— [c]

[a] Best published results obtained by ϕ-functions and mathematical model (Stoyan et al., 2012)
[b] *poly2a*, *poly3a*, *poly4a*, *poly5a*, *poly10a* and *poly20a* instances are multiple of *poly1a*
[c] Not currently solved with free continuous rotations
[d] *swim4* instance is a multiple of *swim*

The instances *poly1a* and its multiples have pieces with few vertices (4.6 per piece) and few concavities, and without compatible angles between pieces (i.e., the concave angles of a piece are not compatible with the convex angles of another). Instance *jakobs1* also contains pieces with few vertices (6.0 per piece) and few concavities, but the pieces in this instance have compatible angles between each other, due to the nearly perfect fits and square angles that the pieces contain. The *swim* instance has a huge number of vertices (20.0 per piece) and concavities, without having pairs of pieces with compatible angles.

The selection of these instances was based on their different geometric characteristics, taking into account the variations in pieces size and their structure (convex and concave), while also considering their number of pieces in order to have a variation in computational cost. Using these instances has the benefit of making computational experiments with different characteristics, thus better understanding the efficiency of the approaches, and also allowing a later comparison with other results in the literature.

The computational experiments were processed on a computer with two Intel Xeon E5-5690 processors at 3.46Ghz, with 48Gb Ram at 1333Mhz and with the operating system Ubuntu 12.04 LTS x86-64. The selected Non-Linear solver is Algencan[2] v2.37 as in (Andreani et al., 2007) and (Andreani et al., 2008), which is a non-linear solver based on the Augmented Lagrangian multipliers method. This solver is based in a single-thread execution. In order to reduce the total computational cost required for a given instance, multiple executions of the solver are done in

[2] http://www.ime.usp.br/~egbirgin/tango/

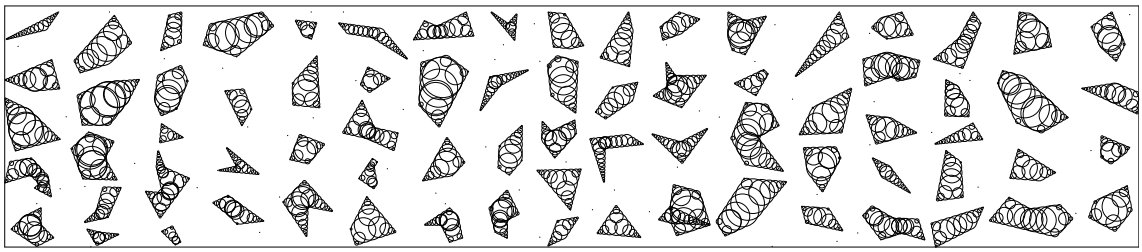


Figure 4.7: Initial solution example for instance *poly5b*.

parallel. This solver does not guarantee convergence to the global optimum. While there are solvers able to converge to an optimal solution, as Jones (2013) shown, they can only be applied to instances up to four pieces, and even in those cases, the computational cost is very high.

The selected solver converges to a local minimum, requiring multiple starting points to explore different regions of the solution space. Since starting from a specific solution will return the same local minimum, having different starting positions allows to explore the solution space for better local minimum, and maybe find the global minimum. This search strategy employed is very basic, where the initial positions of the pieces are defined by placing them into a grid, in a non-overlapping configuration, with random rotations. An example of an initial solution for the instance *poly5b* can be seen in Fig. 4.7. For each nesting instance, 30 different initial solutions were generated and used afterwards in the computational tests. This allows a fair comparison of the results obtained with different approaches and coverings, since they all share the same configuration.

4.4.2 Testing the Model Variants

This set of tests concerns the comparison of the two proposed mathematical models (M1 and M2) and the several variants derived from those models, obtained through constraint aggregation. The objective is to compare the two models and evaluate the constraints aggregation strategy. For this set of tests, the circle covering type used was *CCC* with an *LR* resolution, with the instances *jakobs1*, *poly1a*, *poly2a*, *poly2b*, *poly3a* and *poly3b*. This configuration was selected in order to focus on the model variants, not having concerns about feasibility of the results neither the impact that using a much higher resolution has on the computational cost. Tests were done using all the 30 initial solutions.

Both NLP models share the same objective function that aims to minimize the length l of the strip. This objective function forces the pieces to gradually compact until they reach a stable position, where the solver cannot find any movement that would improve the solution. Table 4.4 summarizes the results obtained for the 6 model variants (*M1S*, *M1E*, *M1T*, *M2S*, *M2E* and *M2T*).

The results indicate that the NLP models and their variants show a consistent behavior, with model M2 achieving best average layouts and usually also the best one, although the computational times are much higher than model M1. The reason for these bigger computational times is due to

Table 4.4: Model variants computational results (obtained using *CCC* and *LR*).

Instance[a]	Model Variant	Obj. Function (l) Min.	Avg.	Max.	Avg. Time (s)	# Runs
jakobs1	*M1S*	16.36	17.77	20.92	14.47	30
(11.82)	*M1E*	15.98	17.96	25.37	1.70	30
	M1T	15.82	18.01	20.89	1.77	30
	M2S	15.33	17.13	19.89	79.47	30
	M2E	15.73	17.14	20.17	18.07	30
	M2T	15.85	17.38	19.69	4.37	30
poly1a	*M1S*	16.61	17.68	18.75	4.60	30
(13.21)	*M1E*	16.28	17.71	21.55	0.97	30
	M1T	15.83	17.66	19.94	1.17	30
	M2S	16.39	17.33	18.57	24.47	30
	M2E	16.04	17.63	19.09	9.00	30
	M2T	15.90	17.64	18.99	3.60	30
poly2a	*M1S*	30.61	32.70	35.26	95.73	30
(25.71)	*M1E*	31.60	33.40	35.44	8.60	30
	M1T	32.00	33.30	36.58	11.63	30
	M2S	30.30	32.25	35.26	635.70	30
	M2E	30.94	32.84	34.58	111.10	30
	M2T	30.83	33.24	35.60	37.97	30
poly2b	*M1S*	34.15	36.27	38.05	106.80	30
(29.07)	*M1E*	35.23	36.70	38.28	9.77	30
	M1T	34.53	37.10	40.83	10.83	30
	M2S	33.83	35.64	38.05	709.17	30
	M2E	33.64	35.78	38.51	121.40	30
	M2T	34.82	36.47	38.72	41.77	30
poly3a	*M1S*	45.94	48.42	50.99	526.93	30
(38.81)	*M1E*	46.22	48.50	51.00	36.80	30
	M1T	46.26	48.79	52.65	38.50	30
	M2S	45.70	47.33	48.99	3104.67	30
	M2E	46.67	47.96	49.65	249.57	30
	M2T	46.39	48.17	49.87	122.27	30
poly3b	*M1S*	46.37	48.43	50.78	528.23	30
(38.79)	*M1E*	46.50	48.46	52.40	29.53	30
	M1T	45.36	48.46	51.62	36.27	30
	M2S	45.00	47.02	50.01	2736.03	30
	M2E	46.37	48.03	55.86	266.47	30
	M2T	46.27	48.44	50.88	118.90	30

[a] Reference value: best result in literature, considering length

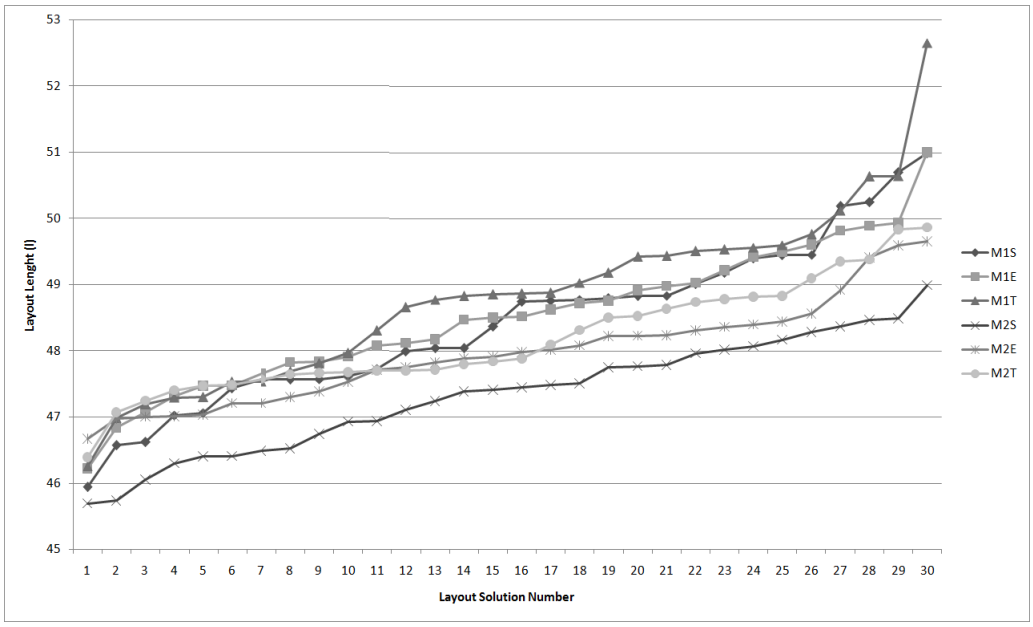
the complexity of the constraints of model M2, specially in the last iterations of the convergence to the local minimum.

Considering the aggregation of constraints, the variants without constraint aggregation, *M1S* and *M2S*, return the best average solutions, but also with the higher computational cost. Aggregating the *Non-Overlapping Constraints* greatly reduces the computational time with a slight decrease in the solution quality. The models with all the constraints aggregated, *M1T* and *M2T*, usually return the worst average solutions. Regarding their computational cost, small differences are noticeable. While on model M2 the computational cost is reduced as the constraints are aggregated (due to constraint exclusion), model M1 shows a different behavior. When the model M1 has all constraints aggregated, the *Piece Integrity Constraints* create difficulties in the convergence to a local minimum, which increases the computational time. These behaviors in the quality of the average solutions and computational time, can be explained by the increase of sensitivity of the solver to numerical precision problems, which reduce its capability to distinguish small changes in the value of the aggregated constraints during the convergence to local minimum.

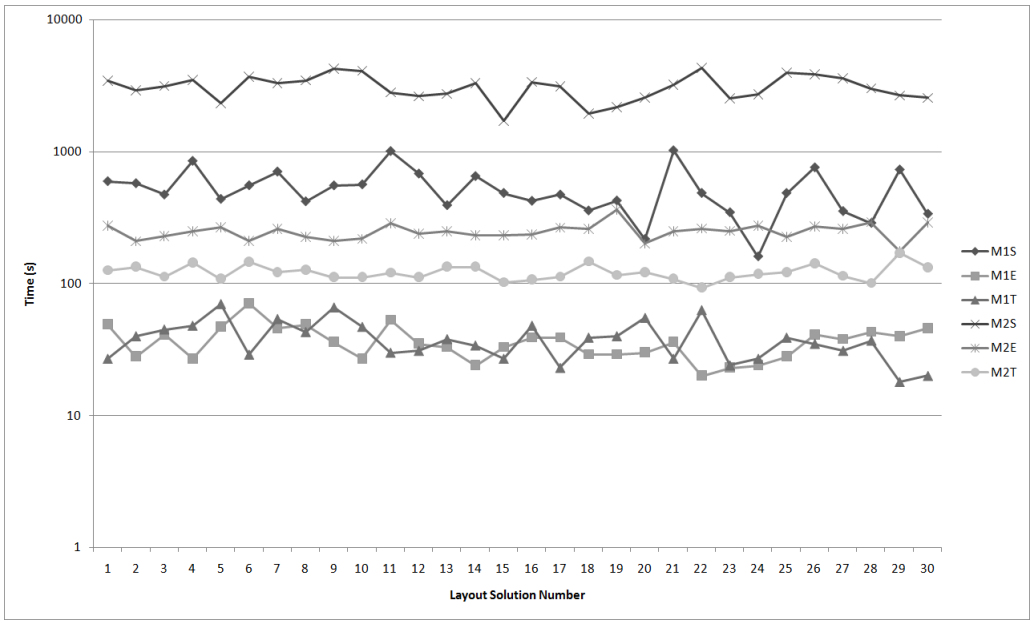
To better compare and analyze the behavior of the different models, the results obtained for instance *poly3a* will be presented in two graphs, one for the compaction layout result, and the other for the computational time. This instance was selected due to its complexity, since less complex instances do not show significant differences between the models. It is hard to compare the two models based on their results in a random sequence, due to one model getting the best result in a certain run but not on other runs, so the results will be presented for each model variant, with each individual variant ordered by their solution quality. When presenting the details about the computational time, the same ordering is used, for each individual model variant.

Considering the compaction results from instance *poly3a*, as seen in Figure 4.8a, the variant *M2S* is clearly seen as the one that provides the best solutions. All of the other variants are very close regarding the best compaction results, but their differences become more significant as the number of initial solutions increases. Variants *M2E* and *M2T* have similar results, although not as good results as *M2S*, while all variants of model M1 (*M1S*, *M1E* and *M1T*) provide inferior compaction results, being *M1T* the variant with worse results. The trend in this, and the other instances, is for the M2 variants to return better compaction results than the variants of M1. Occasionally, the models are not able to fully compact a set of pieces, ending with a bad result, but that may be also feasible (as can be seen in the worst value for the layout length in Figure 4.8a). This problem is due to numerical precision problems that arise from the small variations in the value of the constraints, which reduce the capability of the solver to define the best path to improve the current solution. This problem causes an increase to the computational cost, and may produce very bad solutions, if the solver is not able to converge to a local minimum, due to reaching its maximum number of iterations. When comparing the variants using their computational time, as seem in Figure 4.8b, presented in logarithmic scale, their differences are very noticeable. Variant *M2S* is the most computationally expensive, followed by *M1S* and *M2E*. The variant *M2T* is immediately bellow, and above both *M1E* and *M1T*, which are the least computational expensive.

The behavior of the models in instance *poly3a* is similar to the behavior presented in other



(a) Layout length *l*.



(b) Computational time.

Figure 4.8: Comparison of model variants, using instance *poly3a* with *LR* and *CCC*.

instances, where the trade-off that exists between different NLP formulations for the Nesting problem with continuous rotations is noticeable, considering the quality of the compact solution, and the computational cost required. Model M2 usually returns better compaction results than M1. The variants *M1S* and *M2S* return better results than the variants *M1E* and *M2E* which have their *Non-Overlapping Constraints* aggregated. These variants are also usually better than the variants *M1T* and *M2T* that have all constraints aggregated. When comparing the computational cost, the model M1 has better performance than M2, where the variants that have aggregated constraints (*M1T* and *M2T*) are much faster than have just the *Non-Overlapping Constraints* aggregated (*M1E* and *M2E*) or the variants that do not (*M1S* and M2S).

The results show that the model variants have different behaviors, when comparing the variants based on M1 to the variants based on M2, where the same configuration allows M1 variants to have lower computational cost, while M2 variants provide better solution quality. When the comparison is made considering the change in constraint aggregation, the behavior is similar for the model variants from both models M1 and M2, where the variant S usually provides high quality solutions with high computational cost, and T provides low quality with low computational cost, being the variant E in the middle. Aggregating the *Non-Overlapping Constraints* allows the computational times to be more controllable, which allows the variants *M1E* and *M2E* scale well to instances with more pieces. If the objective is to obtain the layout with the best compaction, the best model is the *M2S*, however, if a solution is required, but with a reduced computational cost, the best is model *M1E*.

4.4.3 Testing the High and Low Resolution Coverings

Another important parameter that affects the computational cost and solution quality is the CC resolution used, i.e., the number of circles used to approximate the pieces. To evaluate the impact of different resolutions the *HR* and the *LR* resolutions are used. The covering type used was the *CCC* in order to obtain admissible solutions, with the model variants *M1E* and *M2E*. These models were selected based on their trade-off between computational cost and quality of layout compaction.

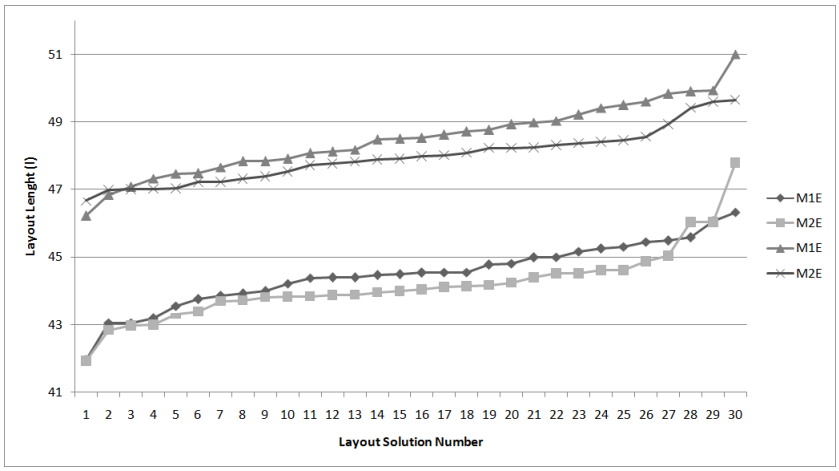
Table 4.5 summarizes the results obtained with 30 initial solutions, for each set of CC resolution, model variant and instances. From the results shown in Table 4.5, the use of *HR* covering shows significant improvements in the quality of the compaction of the layouts. The average value is clearly lower, but the greatest difference can be noticed between the minimum values achieved for each instance, which is significantly lower when using *HR* covering. Another noticeable aspect is that the improvement in the compaction of the layout increases with the complexity of the instance, i.e., when the instances have a larger number of pieces. The main downside in using a *HR* covering is the significant impact on the computational cost, which is about 2 to 3 times the computational cost of using *LR* covering, for the tested instances.

In order to better visualize the impact that a different resolution has on this problem, *poly3a* instance and its results are presented in Figure 4.9. Considering this instance, the graph presented in Figure 4.9a, shows that when using a *HR* covering, the layout compaction results are clearly

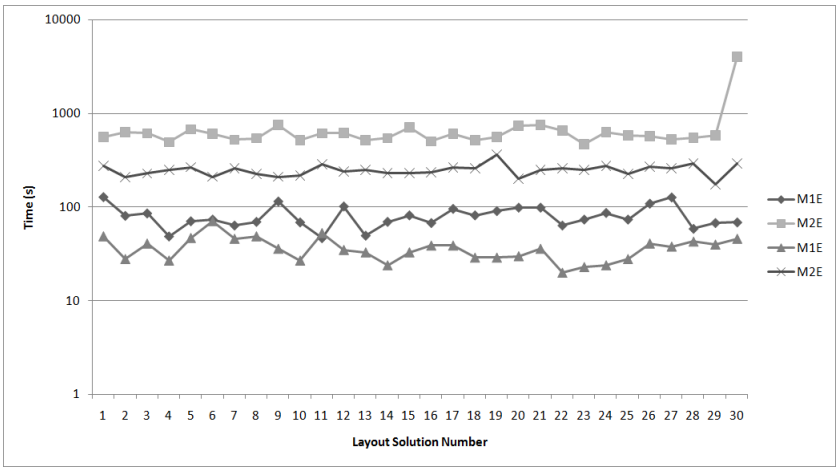
Table 4.5: Circle Covering Resolutions (*LR* and *HR*) computational results (using *CCC*).

Instance[a]	Model Variant	Resolution	Obj. Function (l) Min.	Avg.	Max.	Avg. Time (s)	# Runs
jakobs1	*M1E*	*LR*	15.98	17.96	25.37	1.70	30
(11.82)		*HR*	14.32	16.77	21.79	3.50	30
	M2E	*LR*	15.73	17.14	20.17	18.07	30
		HR	13.85	16.15	20.54	56.00	30
poly1a	*M1E*	*LR*	16.28	17.71	21.55	0.97	30
(13.21)		*HR*	15.20	16.55	18.44	2.27	30
	M2E	*LR*	16.04	17.63	19.09	9.00	30
		HR	14.76	16.10	17.82	29.23	30
poly2a	*M1E*	*LR*	31.60	33.40	35.44	8.60	30
(25.71)		*HR*	27.66	34.58	91.15	22.93	30
	M2E	*LR*	30.94	32.84	34.58	111.10	30
		HR	28.74	29.88	31.69	286.03	30
poly2b	*M1E*	*LR*	35.23	36.70	38.28	9.77	30
(29.07)		*HR*	31.88	33.70	36.74	26.37	30
	M2E	*LR*	33.64	35.78	38.51	121.40	30
		HR	31.26	33.18	35.53	295.77	30
poly3a	*M1E*	*LR*	46.22	48.50	51.00	36.80	30
(38.81)		*HR*	41.95	44.48	46.32	80.80	30
	M2E	*LR*	46.67	47.96	49.65	249.57	30
		HR	41.93	44.17	47.79	710.83	30
poly3b	*M1E*	*LR*	46.50	48.46	52.40	29.53	30
(38.79)		*HR*	42.38	44.07	46.22	70.53	30
	M2E	*LR*	46.37	48.03	55.86	266.47	30
		HR	42.39	44.02	54.15	611.70	30

[a] Reference value: best result in literature, considering length



(a) Layout length *l*.



(b) Computational time.

Figure 4.9: Comparison between different covering resolutions (*LR* and *HR*), using instance *poly3a* with *CCC*.

better than those achieved with *LR*. Comparing the results achieved by both model variants, the variant *M2E* returns better layout solutions than *M1E*. When comparing the computational time, presented in Figure 4.9b considering logarithmic scale, the computational time of the *HR* covering is always significantly higher than the *LR* covering. In this figure, the differences in computational time between the model variants are also clearly observed. Variant *M1E* using either resolution is faster than the variant *M2E*. Again, the occasional event of a compaction with convergence problems due to numerical precision issues can be seen in Figure 4.9a and Figure 4.9b, where one of the model variants is not able to fully compact a set of pieces, ending with a bad, but still feasible, result (as can be seen in the worst value for the total length in Figure 4.9a) and using too much computational time in the process (worst value for the total length in Figure 4.9b). This is due to the maximum number iterations being reached for the solver, due to difficulties in converging to the current local minimum.

In order to have a visual representation of the impact that using *LR* covering and *HR* covering have on the quality of the layout compaction, Figure 4.10 presents two solutions that have been compacted with both resolutions. This figure clearly demonstrates that when using a *HR* resolution instead of a *LR* one, together with *CCC*, the pieces have a higher probability to achieve tighter layouts, due to the smaller distance between pieces, when compared to the *LR* covering. The *CCC* was used to facilitate the comparison between the model variants and different resolution coverings, by allowing to analyze their results with feasible solutions. If other types of coverings had been used, the solutions would not be directly comparable, since the other types of covering (*PCC* and *ICC*) generate infeasible solutions.

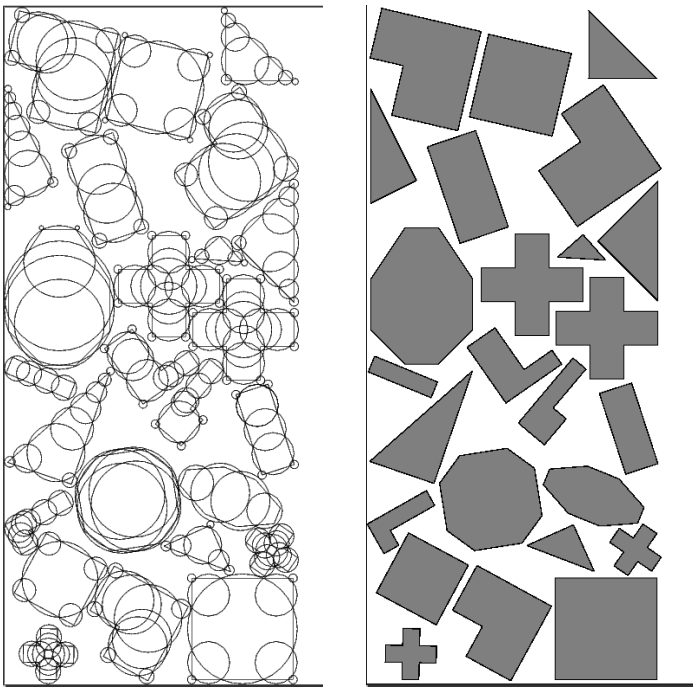
The use of different circle covering resolutions has a strong impact on the quality of the layout compaction and in the computational time. Using a *LR* covering allows to compact the layout much faster, but the quality of the results of the layout compaction are not satisfactory. The *HR* covering returns tight layouts but with a much higher computational cost. The increase in computational cost due to different resolutions indicates that the cost of using higher resolutions than *HR* becomes too high to return solutions in an acceptable amount of time.

4.4.4 Layout Post-Optimization Computational Experiments

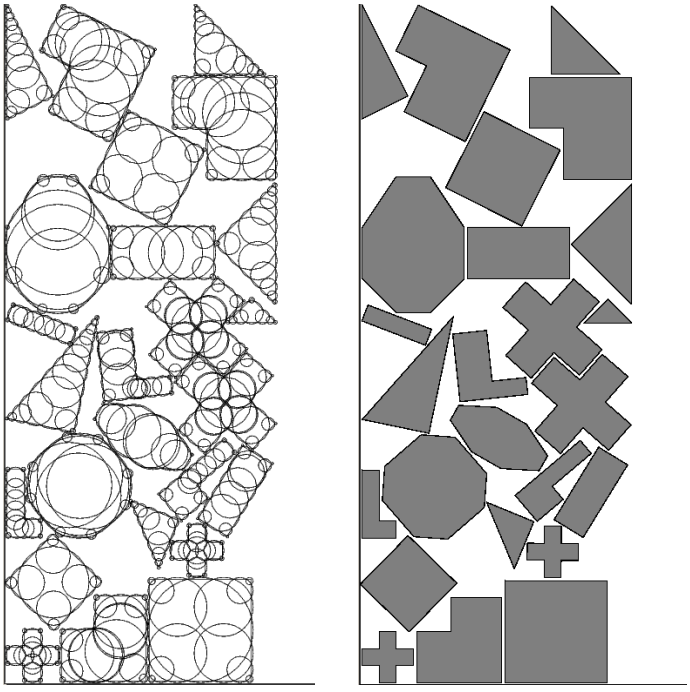
The Post-Optimization phase is used to address the infeasible layouts created by CC that do not completely cover the pieces, such as the *PCC* and *ICC*, and also reducing the excess waste derived from *LR* and *HR* coverings when using the *CCC*. In order to allow pieces to more easily adjust to each other, a higher resolution is used, such as High Resolution Plus (*HRP*) or Very High Resolution (*VHR*), since it allows a covering representation closer to the original polygonal outline of the piece. The main downside to this Post-Optimization phase is that it increases the total computational cost required to achieve a feasible solution, while not being able to guarantee that the correction of the layout will be successful. In this section the computational experiments regarding the Post-Optimization approach will be shown, in order to analyze the results and determine which is the best configuration that allows correcting the highest number of infeasible solutions, and also producing the highest solution quality possible. The presented results show a combination of high resolution such as *HRP* or *VHR* with *CCC*, and several model variants.

A visual representation of the different levels of resolution that exist, for some of the instances used, can be found in Table 4.6. Since the objective is to correct infeasible layouts, without increasing its length significantly, using identical or lower resolutions in the post-optimization phase does not bring any advantage. The *LR* and *HR* coverings are present in Table 4.6 in order to allow comparison with the higher resolution coverings *HRP* and *VHR*. As seen in Table 4.6, the increase in the quality of approximation leads to a significant growth in the number of circles. The columns in the right side of Table 4.6 show the average number of circles for each piece which allows to determine the relative difficulty between the instances, when also considering their total number of circles.

Each instance was tested with 30 initial solutions which were compacted (during the normal optimization process) with the model variant (*M2E*), using *CCC*, *PCC* and *ICC*, and *HR* covering,



(a) *LR-CCC* layout. (b) Piece layout for the *LR-CCC*.



(c) *HR-CCC* layout. (d) Piece layout for the *HR-CCC*.

Figure 4.10: Layouts obtained for instance *jakobs1* with *HR* and *LR* coverings.

Table 4.6: Number of circles for Post-Optimization.

	# Circles (total)				# Circles per Piece			
Instance	*LR*	*HR*	*HRP*	*VHR*	*LR*	*HR*	*HRP*	*VHR*
jakobs1	208	368	521	—	8.3	14.7	20.8	—
poly1a	154	260	370	853	10.3	17.3	24.7	56.9
poly2a	308	520	740	1706	10.3	17.3	24.7	56.9
poly2b	336	534	778	1799	11.2	17.8	25.9	60.0
poly3a	462	780	1110	2559	10.3	17.3	24.7	56.9
poly3b	476	755	1107	2546	10.6	16.8	24.6	56.6
poly4a	616	1040	1480	3412	10.3	17.3	24.7	56.9
poly4b	624	989	1448	3342	10.4	16.5	24.1	55.7
poly5a	770	1300	1850	4265	10.3	17.3	24.7	56.9
poly5b	770	1227	1795	4145	10.3	16.4	23.9	55.3
swim	921	1179	1362	2610	19.2	24.6	28.4	54.4

in order to be able to analyze the Post-Optimization approach using different characteristics. The Post-Optimization method was applied to the results obtained, and its results analyzed considering different resolutions (*HRP* and *VHR*) and different model variants (*M1E* and *M2E*). The last set of computational experiments analyzed the influence that using different covering types on the normal optimization phase has on the results returned by the post-optimization phase.

To verify the impact that both *HRP* and *VHR* coverings have in the Post-Optimization phase, regarding computational cost and quality of compaction, several instances were selected (*poly1a*, *poly2a*, *poly2b*, *poly3a* and *poly3b*) and tested using the model variant *M2E*. These instances were selected due to their size and complexity, while the choice of the model variant used was selected due to the reasons previously stated (able to achieve good quality solutions, within an acceptable computational time). Considering the results shown in Table 4.7, increasing the resolution from *HRP* to *VHR* causes a very small decrease in the length of the layout, at the expense of a significant increase in the computation time. The same increase in resolution does not create a significant difference in the number of feasible solutions, being both able to produce feasible solutions almost every time. However, when infeasible solutions occur, they may be derived from numerical precision errors, or lack of space to allow pieces to correct overlaps (due to fixed width of the container, and pieces with concavities that prevent other pieces from separating), or another reason. The computational tests for the post-optimization consider only values of the solution quality and computational cost that correspond to feasible layout solutions.

The computational cost of the Post-Optimization phase can be reduced by using a different model variant, such as *M1E*, however, this variant usually returns worse quality solutions than the previously used *M2E* variant. Table 4.8 shows computational experiments regarding the impact that the *M1E* variant has when used in the post-optimization procedure. As expected, the *M1E* variant returns layouts with lower quality than *M2*, but at a significantly lower computation time. The total number of feasible solutions is lower than *M2E* when using the the *M1E* variant. This is due to numerical precision problems that arise with the non-overlapping constraint aggregation,

Table 4.7: Impact of resolution in the Post-Optimization, with *CCC* and *M2E*.

	Avg.O.F.[a](*l*)		Avg.Time[a](s)		# Feas. Solutions		#
Instance	*HRP*	*VHR*	*HRP*	*VHR*	*HRP*	*VHR*	Runs
poly1a	15.48	15.07	4.23	37.55	30	29	30
poly2a	28.59	27.73	46.70	488.14	27	29	30
poly2b	31.96	30.90	49.38	640.83	29	30	30
poly3a	42.38	40.93	132.61	2003.58	28	26	30
poly3b	42.16	41.07	161.96	1824.25	27	28	30

[a] Avg. values considering only feasible solutions

Table 4.8: Impact of model variants *M1E* and *M2E* in the Post-Optimization, with *CCC* and *VHR*.

	Avg.O.F.[a](*l*)		Avg.Time[a](s)		# Feas. Solutions		#
Instance	*M1E*	*M2E*	*M1E*	*M2E*	*M1E*	*M2E*	Runs
poly1a	15.31	15.07	6.48	37.55	25	29	30
poly2a	28.18	27.73	33.11	488.14	27	29	30
poly2b	31.29	30.90	46.48	640.83	23	30	30
poly3a	41.47	40.93	86.44	2003.58	27	26	30
poly3b	41.11	41.07	88.89	1824.25	19	28	30

[a] Avg. values considering only feasible solutions

since the smallest circles that compose the pieces can easily overlap circles of other pieces, without a significant effect on the aggregated non-overlapping constraint. The constraint aggregation causes the effects of these small circles to be barely noticeable, which may be considered an approximation error. This model variant also has difficulties removing overlaps from pairs of pieces that have circles on both sides of the other piece (i.e. when the skeletons of the pieces intersect each other). This causes circles from the same piece to move in different directions to minimize the overlap, but without success since doing so would violate the integrity of the piece. The *M2E* model variant has a different formulation, which explains its different behavior. Since the reference point of the piece is considered, and not its circles, when the overlap is to be minimized between two pieces, in the same condition as explained before, the piece will be forced to move in one direction only, being able to remove overlaps much more efficiently.

Up to this point the Post-Optimization processed layouts compacted with *CCC*, which were feasible, in order to isolate the influence of both resolutions and model variants. The main purpose of using the Post-Optimization is to allow correcting infeasible layouts generated by *PCC* and *ICC*, and for this reason, the next set of computational experiments, shown in Table 4.9 will analyze the impact that they have on the Post-Optimization phase, when used during the normal optimization phase. As seen in this table, using *CCC* returns layouts with lower quality, but almost all of them being feasible. The difference in compaction quality between the *PCC* and *ICC*, after the Post-Optimization phase, is very small. Most of their layouts start tightly compacted, and have their length increased in the post-optimization phase when being corrected. The greatest difference can

Table 4.9: Impact of covering types *CCC*, *PCC* and *ICC* in the Post-Optimization, with *HRP* and *M2E*.

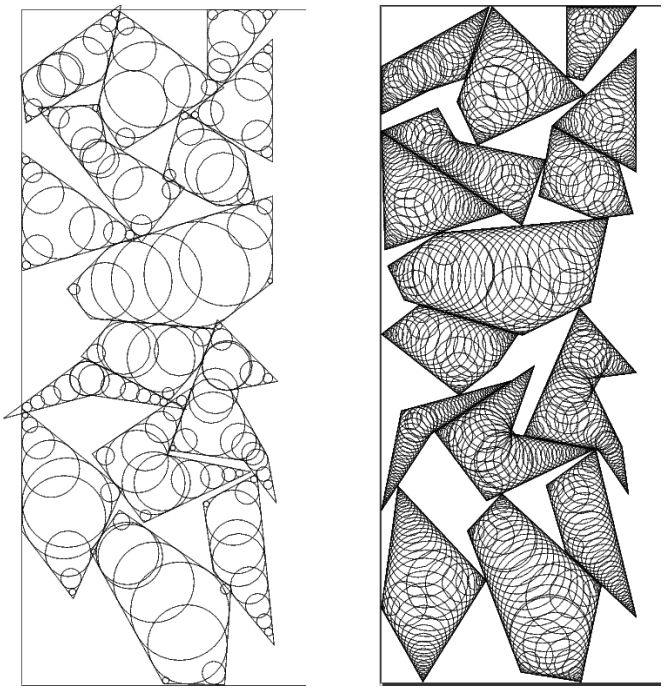
	Avg.O.F.[a](*l*)			Avg.Time[a](s)			# Feas. Solutions			#
Instance	*CCC*	*PCC*	*ICC*	*CCC*	*PCC*	*ICC*	*CCC*	*PCC*	*ICC*	Runs
poly1a	15.48	15.30	15.35	4.23	19.35	20.00	30	23	11	30
poly2a	28.59	28.64	27.55	46.70	139.00	195.29	27	15	7	30
poly2b	31.96	31.44	31.84	49.38	109.43	504.69	29	23	29	30
poly3a	42.38	41.87	40.99	132.61	89.56	791.00	28	18	5	30
poly3b	42.16	41.90	42.50	161.96	217.47	1285.38	27	19	26	30

[a] Avg. values considering only feasible solutions

be seen in their computational time, and number of admissible solutions.

An abnormally high computational time indicates that the solver is having difficulty in achieving a local minimum in the layout, which can be caused by several factors, such as numerical precision problems, which cause difficulties converging to local minimum, also due to lack of space to move pieces in order to correct overlaps, among others. It is natural that the type of covering that generates the tightest layouts (*ICC*) is the most difficult to correct the overlaps between its pieces. This large increase in computational cost is not shown in Table 4.9 since only the values for the instances that were successfully corrected by the Post-Optimization Phase are shown. The excluded infeasible layouts wasted their computational time completely, and their average is significantly higher than the feasible results, because the solver continues trying to reach a feasible local minimum until it succeeds, or the number of iterations reaches the maximum. An example of the Post-Optimization phase can be seen in Fig. 4.11 where an infeasible solution, produced during the normal optimization phase with *ICC-LR*, is shown, together with the corrected layout, using the Post-Optimization phase, with *CCC-VHR*.

Due to the significant impact that the number of circles has on the computational cost, the use of *HRP* is preferable, compared to *VHR*. When the layouts have been compacted with *PCC* or *ICC*, the use of the post-optimization is necessary, however, using it on admissible layouts initially compacted using *CCC* can provide a slight improvement in the quality of the layout compaction. The use of *VHR* covering can be justified to further improve a small set of very promising layouts, but with a large computational cost. The model variant should be selected based on the expected computational cost. The most balanced variants are *M1E* for low computational cost, and *M2E* for high compaction quality. The post-optimization does not guarantee that the layouts will return admissible solutions when they have been created by *PCC* or *ICC* due to difficulties removing overlaps caused by numerical precision problems, and also due to limited space available for the pieces to move (such as when limited by the fixed width of the container, when a concavity from one piece prevents other pieces to move in opposite directions to correct their overlaps, among other examples).



(a) Infeasible solution produced with *ICC-LR*.

(b) Feasible solution due to Post-Optimization using *CCC-VHR*.

Figure 4.11: Example considering *ICC-LR* during the normal optimization phase and *CCC-VHR* durign the Post-Optimization phase

.

4.4.5 Testing the Circle Covering Types

In order to determine the impact that the type of Circle Covering has when solving Nesting problems with continuous rotations, several covering types were tested. The model variants used were *M1E* and *M2E*. The aim of this test is to complement the tests done previously, by evaluating the computational cost and quality of the layouts obtained with the different circle covering types. The circle covering types compared in this tests have already been introduced in chapter 3, and are defined as *CCC*, *PCC* and *ICC*. The circle coverings *PCC* and *ICC* have their coverings improved by the tip covering algorithm, in order to achieve a covering that minimizes the inadmissibilities that might occur in the layout. For this computational experiment the instances that were selected are *jakobs1*, *poly1a*, *poly2a*, *poly2b*, *poly3a* and *poly3b*, i.e., the ones defined as small or medium instances.

Table 4.10 summarizes the results obtained from a run with 30 initial solutions, showing only the admissible solutions (after applying the post-optimization phase to solve inadmissible solutions and improve the ones already admissible). The minimum, average and maximum values for the layout compaction are shown, together with the average computational time.

Taking into account the minimum and average values of the layout compaction, it can be seen that the *CCC* is able to return, almost always, admissible solutions (it fails on some cases due to numerical precision errors) but rarely achieves the best quality. The *ICC* has many layout solutions that cannot be turned into admissible solutions, but the tendency is to achieve better results than the coverings *CCC* and *PCC*. This is due to the *ICC* covering having a smaller covering of the piece (since the circles are contained inside the piece outline). The disadvantage of this type of covering is the high number of infeasible layouts, which means that the resulting layout has a high number of occurrences of overlaps between pairs of polygons. An infeasible layout is troublesome to resolve without degrading the quality of the final layout solution, and depending on the size and tightness of the compacted layout, the pieces may lack sufficient space to be able to adjust and eliminate the overlaps. The covering type *PCC* has the best trade-off between admissible solutions and the quality of the solution, by having a number of admissible solutions close to *CCC* and a quality of solution comparable to *ICC*.

The main downside in using *PCC* and *ICC* types of coverings is their possibility to generate infeasible layouts, which require post-optimization to correct, and even then, may not be able to successfully eliminate the overlaps. This Post-Optimization method increases the total computational time to solve the Nesting problem, which depends on the number of circles of the *CCC* resolution used, *HRP* or *UHR*. This can be seen in Table 4.11, where the effect of using *CCC*, *PCC* or *ICC* (including post-optimization) is analyzed, together with both model variants *M1E* and *M2E*. The columns named *Avg. O.F. Nrm.* and *Avg. O.F. P-Opt.* refer to average values of the layout compaction obtained in the normal compaction, and the post-optimization re-compaction, respectively. Regarding the columns that refer to *Avg. Time Nrm.* and *Avg. Time P-Opt.*, the same applies.

Table 4.10: Comparing Circle Coverings Types, with *HR* (including Post-Optimization using *HRP*).

Instance[a]	Model Variant	Covering Type	Obj. Function[b](l) Min.	Avg.	Max.	Avg. Time[b](s)	# Feas. Solutions
jakobs	*M1E*	*CCC*	13.62	16.1	21.34	4.40	29
(11.82)		*PCC*	13.87	15.7	21.35	4.57	25
		ICC	13.16	14.8	19.36	8.61	19
	M2E	*CCC*	13.28	15.2	20.26	73.60	26
		PCC	13.26	15.6	25.01	61.57	25
		ICC	13.08	14.5	19.10	96.97	10
poly1a	*M1E*	*CCC*	14.70	15.8	17.90	2.59	29
(13.21)		*PCC*	14.52	15.6	17.00	3.38	26
		ICC	14.47	15.5	16.82	3.23	13
	M2E	*CCC*	14.18	15.5	17.24	32.43	30
		PCC	14.42	15.3	18.52	47.18	23
		ICC	14.46	15.3	16.17	45.20	11
poly2a	*M1E*	*CCC*	26.79	33.6	91.10	23.64	28
(25.71)		*PCC*	27.49	34.3	91.10	25.00	23
		ICC	27.61	28.9	30.75	54.22	8
	M2E	*CCC*	27.18	28.6	30.15	299.80	27
		PCC	26.98	28.6	30.43	375.43	15
		ICC	26.77	27.5	29.57	420.32	7
poly2b	*M1E*	*CCC*	30.32	32.3	34.52	28.26	29
(29.07)		*PCC*	30.37	31.8	33.49	32.14	25
		ICC	30.96	32.6	34.92	41.77	30
	M2E	*CCC*	30.55	32.0	33.76	324.81	29
		PCC	29.55	31.4	33.35	374.47	23
		ICC	29.68	31.8	34.13	504.69	29
poly3a	*M1E*	*CCC*	40.84	42.7	44.11	83.00	30
(38.81)		*PCC*	40.92	42.3	44.48	92.07	21
		ICC	40.22	41.5	42.35	131.81	9
	M2E	*CCC*	40.53	42.4	44.53	805.61	28
		PCC	40.46	41.9	44.52	608.26	19
		ICC	40.20	41.0	41.63	1400.90	5
poly3b	*M1E*	*CCC*	41.12	42.6	44.94	77.03	25
(38.79)		*PCC*	40.46	42.2	44.03	102.14	21
		ICC	40.25	42.6	45.29	124.54	28
	M2E	*CCC*	41.10	42.2	43.53	759.06	27
		PCC	40.20	41.9	45.14	830.04	19
		ICC	40.30	42.5	51.16	1285.38	26

[a] Reference value: best result in literature

[b] Values considering only feasible solutions

The Post-Optimization method usually is able to improve the solution quality of layouts generated by *CCC* and *PCC*, while the layouts generated by *ICC* are usually degraded. Considering the average solutions, it can be seen that *ICC* is able to return better but very few feasible solutions. The *CCC* is able to return feasible solutions in almost all cases, although their solution quality is lower than the other types of coverings. The infeasible solutions that arise with this covering are created by the numerical precision errors, which prevent the solver from reaching a local minimum, and forcing it to stop when the maximum number of iterations is reached. The *PCC* achieves better quality solutions than *CCC* due to its circle outline being closer to the polygonal outline of the piece. However, the *PCC* produces less feasible solutions than *CCC* but more than the *ICC*. The *ICC* generates a very low number of feasible solutions, due to the large number of overlaps between pieces caused by its very tight layouts, and the lack of space to move the pieces. The pieces may be constrained by the fixed width of the container, or due to concavities of a piece that prevents other pieces from moving apart from each other. These difficulties also arise in the *PCC* but in a lesser degree, since it is also an incomplete covering of the polygonal representation of the piece. Considering these types of covering, the *PCC* is the one that provides the most balanced results considering solution quality and computational cost.

To illustrate visually the impact that each of these types of coverings have on a layout, Figure 4.12 shows three examples obtained with the *CCC* (Figure 4.12a), *PCC* (Figure 4.12b) and *ICC* (Figure 4.12c), together with their respective polygonal representation. This allows to see the distance separating the pieces and the overlapping created between them, depending on their covering type. The *PCC* and *ICC* are infeasible solutions, while the *CCC* is feasible.

Figure 4.12d shows the layout obtained from *CCC* compaction, where the pieces have a small separating distance among them. When comparing it to the distance in Figure 4.12e, with *PCC*, the difference is noticeable since the pieces are closer together, but very few significant occurrences of overlap can be seen. In comparison to Figure 4.12f, that uses *ICC*, nearly all corners of the pieces are overlapping other pieces. This is due to the value of the allowed penetration depth in *ICC*. Using a Post-Optimization approach allows correcting the overlaps and directly comparing the layouts.

4.4.6 Non-Linear Programming Approach Evaluation

Taking into account all the approaches presented in this chapter, the selection of the best configuration parameters for each one may enable achieving comparable results to the currently presented in the literature. The computational experiments regarding the model, resolution and covering type can be defined considering that the objective is to obtain the best solution, within an acceptable computational time. For this reason, we propose using the model *M2E* for achieving a good solution quality with a reasonable computational cost, and *M1E* for a reduced computational cost, although with a reduced solution quality. Considering the resolutions, using *HR* has significant advantages over *LR* where using a higher resolution increases the computational cost by a large

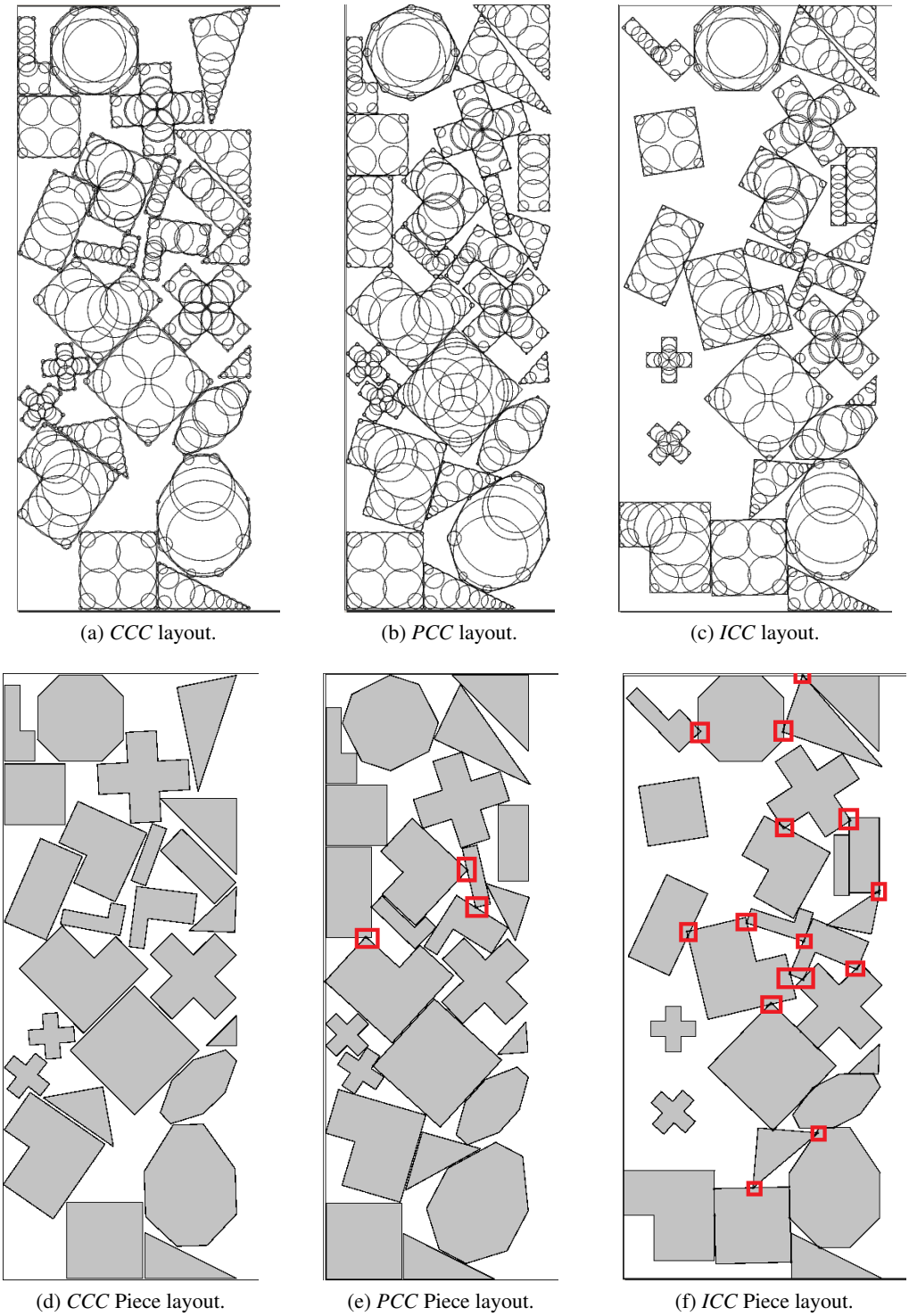
Table 4.11: Impact of Circle Covering Types (with *HR*) in the Post-Optimization procedure (with *HRP*).

Instance[c]	Model Variant	Covering Type	Avg. O. F. (*l*)		Avg. Time (s)		# Feas. Sol.	
			Nrm.[a]	P-Opt.[b]	Nrm.[a]	P-Opt.[b]	Nrm.	P-Opt.
jakobs1	*M1E*	*CCC*	16.77	16.09	3.33	1.07	30	29
(11.82)		*PCC*	15.95	15.74	3.33	1.24	0	25
		ICC	15.32	14.79	2.67	5.95	0	19
	M2E	*CCC*	16.15	15.21	55.60	18.00	30	26
		PCC	15.90	15.64	51.53	10.04	0	25
		ICC	14.46	14.50	48.67	48.30	0	10
poly1a	*M1E*	*CCC*	16.55	15.84	1.87	0.72	30	29
(13.21)		*PCC*	15.83	15.64	2.00	1.38	0	26
		ICC	15.13	15.53	2.00	1.23	0	13
	M2E	*CCC*	16.10	15.48	28.20	4.23	30	30
		PCC	15.49	15.30	27.83	19.35	0	23
		ICC	15.42	15.35	25.20	20.00	0	11
poly2a	*M1E*	*CCC*	34.58	33.64	18.00	5.64	30	28
(25.71)		*PCC*	33.68	34.31	14.13	10.87	0	23
		ICC	32.28	28.86	14.97	39.25	0	8
	M2E	*CCC*	29.88	28.59	253.10	46.70	30	27
		PCC	28.72	28.64	236.43	139.00	0	15
		ICC	27.45	27.55	225.03	195.29	0	7
poly3a	*M1E*	*CCC*	44.48	42.70	70.23	12.77	30	30
(38.81)		*PCC*	42.70	42.28	64.73	27.33	0	21
		ICC	41.13	41.46	63.70	68.11	0	9
	M2E	*CCC*	44.17	42.38	673.00	132.61	30	28
		PCC	42.12	41.90	522.37	85.89	0	19
		ICC	39.92	40.99	609.90	791.00	0	5

[a] Avg. values considering all solutions

[b] Avg. values considering only feasible solutions

[c] Reference value: best result in literature



(a) *CCC* layout. (b) *PCC* layout. (c) *ICC* layout.

(d) *CCC* Piece layout. (e) *PCC* Piece layout. (f) *ICC* Piece layout.

Figure 4.12: Instance *jakobs1* with *CCC*, *PCC* and *ICC*.

Table 4.12: Current approaches compared with best in literature, using *HRP-CCC* covering.

	(Stoyan et al., 2012)[b]		*M1E*		*M2E*	
Instance	Min. O.F. (l)	Time-to Best (s)	Min. O.F. (l)[a]	Avg. Time (s)[a]	Min. O.F. (l)[a]	Avg. Time (s)[a]
jakobs1	11.82	495	13.16	2.75	13.08	25.45
poly1a	13.21	150	14.47	1.11	14.18	14.53
poly2a	25.71	280	26.79	18.59	26.77	127.00
poly2b	29.07	958	30.23	19.78	29.55	221.17
poly3a	38.81	390	40.22	36.07	40.20	337.72
poly3b	38.79	1156	40.25	56.84	40.20	554.94

[a] Values considering only feasible solutions

[b] Results obtained using ϕ-functions and a mathematical model.

amount. Regarding the selection of covering types, the *PCC* has the best trade-off between solution quality, feasibility of the solutions and their computational cost (also derived from using the post-optimization method).

The best results produced by *M1E* and *M2E* can be seen in Table 4.12, side by side, with the current best results available in the literature, considering continuous rotations. Our computational time is expressed in average values, considering both model variants *M1E* and *M2E* using *HR* covering with *CCC*, including the compaction with Post-Optimization. This is not directly compared to the computational cost presented in the literature since it is only presented the Time-to-Best, and not the overall optimization time.

The best results from the literature are not configured to simulate the conditions of a real-world problem, since they have the gap between pieces defined as zero. In real-world problems this value is never zero, due to technological constraints imposed by the cutting process. These types of constraints can be easily simulated by the Circle Covering representation, where the most favorable types of covering are the ones that have a positive protrusion distance between the pieces, such as *CCC* and *PCC*. Comparing the CC approach to other approaches (that can accurately represent the polygonal representation of the piece) places it at a disadvantage when tackling problems that do not require a separating gap.

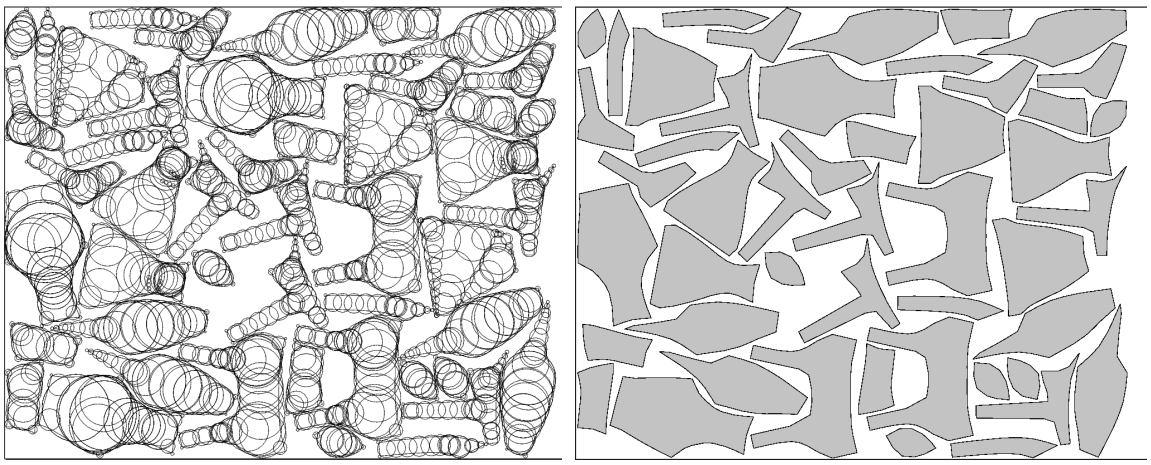
Regarding the remaining instances, the larger instances *poly4a*, *poly4b*, *poly5a* and *poly5b* were not solved due to their computational cost. However, the instance *swim* was solved, in order to verify that the developed approach is able to tackle this instance. It should be noticed that the *swim* instance has not yet been solved using continuous rotations, except with our approach. Its results, using several model variants, can be seen in Table 4.13. This is a particularly challenging instance due the geometric complexity of the pieces, which contains a high number of vertices and many concavities. Solving this challenging instance validates the proposed approach.

Taking into consideration the values presented in Table 4.13, our best result is 7023 units, with model *M2S* with its average computational time being 36242 seconds. This table also shows the difference in solution quality and computational time between model variants, confirming that the

Table 4.13: Comparison of model variants for instance *swim* with *CCC* and *HR*.

Model Variant	Obj. Function[a] (l)			Avg. Time (s)[a]
	Min.	Avg.	Max.	
M1S	7285	7647	8086	5531.32
M1E	7209	7685	8507	423.05
M1T	7263	7620	8595	420.14
M2S	7023	7293	7639	36 241.71
M2E	7267	8985	20671	4806.59
M2T	7294	7693	8500	1127.29

[a] Values considering only feasible solutions



(a) Compaction with Circle Covering. (b) Compaction with only the pieces visible.

Figure 4.13: Example of *Swim* compaction, *HR*, *CCC*.

best solutions are usually obtained by *M2S* variant, but with a prohibitive computational cost. The best solution can be seen in Figure 4.13. This figure shows the layout with the circle covering containing a large number of circles and the layout with the polygonal representations of the pieces.

4.5 Concluding Remarks

In this chapter several non-linear programming (NLP) model formulations have been presented. All formulations are derived from two basic models, which focus on solving the Nesting problem with continuous rotations, taking into account its specific characteristics, by exploring different formulations. One NLP model focused on the description of pieces (represented by Circle Covering) by considering them as composed by a set of individual circles (Circle Based Model *M1*), while the other NLP model described the pieces as a single element (its reference point) from where the circle covering outline is derived (Piece Based Model *M2*). These two alternative

formulations of the Nesting problem have different intrinsic characteristics, allowing *M1* to use simpler equations, at the expense of a higher number of variables and constraints than *M2*. In both models, the number of variables and constraints increases with a higher number of pieces and also their number of circles, with the number of constraints having a factorial growth. In order to control the growth of the constraints (and thus, limiting also the increase in computational cost for larger instances), a method based on constraint aggregation (by type of constraint) was implemented, for each type of constraint. This gave rise to three model variants (*S*, *E* and *T*), for each model (*M1* and *M2*), where the *S* variant did not use any constraint aggregation, while the *E* variant aggregated only the non-overlapping constraints (they have exponential growth, while the others have linear growth), and the *T* variant aggregated all types of constraints into individual constraints.

The advantage of constraint aggregation is based on two effects, being the first one on reduction of the internal overhead in the solver derived from the selection of the individual constraints to compute their value, and the second being the possibility to use a hierarchical overlap detection approach to discard overlapping constraints between pieces that are far apart. The hierarchical overlap detection method starts by comparing the bounding box of the pieces, then the minimum enclosing circle, and only then the real circle covering. The solver allows some constraints to be infeasible during its optimization process, leading to difficulties in converging to local minimums. This problem is aggravated due to numerical precision errors. The downside of using constraint aggregation is that the solver loses sensitivity to small variations produced from individual constraints, since the values of all positive overlaps are added together, causing an increase in the total numerical precision error. This increases the difficulty of the solver in converging to a better local minimum, due to losing capability to distinguish between small variations in the quality of the result.

The selection of the best model variant depends on the computational cost limit, for which one of them is able to provide the best results. The *M1* model usually produces faster compactions, but with lower quality than *M2*, which can be explained by its simpler equations. Considering their variants, the variant without aggregated constraints (*S*) returns the best quality but being the slowest, while the variant with all constraints aggregated (*T*) is usually the fastest, but with lowest quality. The variant with only the non-overlapping constraints aggregated (*E*) is a balanced approach, considering the others, returning average quality with an average computational time. The best quality solution is usually obtained by the model variant *M2S*. Depending on its computational cost, other variants can be selected, in order to chose the model variant that leads to the best quality, without exceeding the desired limit in computational cost. The variants with more aggregated constraints are faster, however, their solution quality is inferior. Variant *M1S* is better than *M1E*, which is better than *M1T* considering solution quality. Regarding computational cost, their behavior is reversed. This is the same for variants *M2S*, *M2E* and *M2T*. When comparing models, *M1* is faster than *M2*, but it usually returns worse compaction solutions.

Instead of adjusting the computational cost through the model variant, another option is defining a circle covering representation with higher or lower quality of approximation. Coverings with

distinct approximation quality have different number of circles, which impact the computational cost through the number of constraints generated from them. Higher approximation quality leads to circle coverings with more circles which increase the number of non-overlapping constraints exponentially, which leads to a significant increase in the computational cost. The computational costs have shown that using a High Resolution (*HR*) covering returns tighter compactions than Low Resolution (*LR*) covering but with a significant increase in computational time. Using higher resolutions than *HR* for the compaction process is prohibitive, while using resolutions lower than *LR* produces layouts with very bad quality.

Another important aspect is the type of the selected circle covering. This issue does not have a direct impact on the complexity of the *NLP* models since their total number of circles is nearly identical, so the impact on the computational cost of the model is very reduced. The different types of resolutions have mostly influence on the quality of the final solution, considering its feasibility and tightness of the layouts. Since the Complete Circle Covering (*CCC*) produces a circle covering with excess covering, with a specific protrusion distance, it naturally leads to feasible layouts at the expense of the pieces not touching. The other covering types, Partial (*PCC*) and Inner Circle Covering (*ICC*) exhibit a different behavior, since these two coverings have a positive penetration depth, i.e., the circle covering does not completely cover the polygonal representation of the piece leading to infeasible solutions. The main difference between the two is that the *ICC* only has positive penetration depth (with zero protrusion distance), which allows the pieces to be placed in configurations with perfect fits, while the *PCC* has both positive protrusion distance and penetration depth. The computational experiments show that using a *CCC* returns feasible solutions but plenty of wasted space on the layout, while the *ICC* achieves the tightest layout compactions, although with very few feasible solutions. The *PCC* is shown to produce the best trade-off considering the quality of the layout compaction, and the total number of feasible solutions.

The feasibility of the layouts was addressed using an approach based on a *CCC* together with a resolution higher than *LR* and *HR*. The NLP model is used to re-compact the pieces of the infeasible layout, aiming to produce a feasible one. This approach has shown to have some limitations, regarding achieving feasible solutions, caused by numerical precision problems, the efficiency of the selected model variant, and the type of covering used. The computational cost grows when these problems appear, and is completely wasted when no feasible solution is produced. For this reason, choosing the right set of configuration parameters (model variant, resolution, and type of covering) is important to achieve feasible layouts with high quality, with a reasonable computational cost.

Regarding the configuration parameters, one important component that has a huge impact on the final result, is the method used to generate the initial placement positions of the pieces. Since the solver used starts with the initial solution and converges to a local minimum, the relative positions of the pieces, as exist in the initial solution, will not change significantly. The current way to produce a good initial solution is to place the pieces into a grid structure, in a random sequential order, with random orientations.

These approaches are able to solve the Nesting problem with continuous rotations, but demand an extensive knowledge about the behavior of the models and which configuration parameters are better suited to provide the best solutions. There is also the possibility of improving the approaches by using a multi-step approach to the problem, where at each step, the configuration parameters change, depending on the objective, and the solution obtained in the previous step. Modifications include changing the objective function, and trying to solve smaller parts of the problem, which are explored in the next chapter.

Some of the approaches presented in this chapter can also be found in (Rocha et al., 2013b). In it, the Nesting problem with continuous rotations is addressed, but using only the model variant *M1S* in order to introduce the NLP model formulation, and its implementation regarding the multiple covering types and resolutions. The instances presented are also a subset of the instances used in this chapter.

Chapter 5

Extended Non-Linear Approaches

The previous chapter presented a solution approach to the Nesting problem with continuous rotations based in alternative Non-Linear Programming formulations. The solution approach uses one of the formulations to create a NLP model for a given instance. This model is then solved by a standard NLP solver, starting from multiple initial solutions and producing compacted layouts. Although previous solutions were interesting, there is a need for higher performance and better quality layouts. In order to achieve that, extended approaches were studied and explored, building up on strengths of the previous solution approach and addressing its weaknesses. These approaches will be presented in the following sections, with their description and computational experiments.

The first approach is named Multi-Resolution approach, in the first section of this chapter. It is based on using a two-step process for the normal optimization process, where the main characteristic is the use of different resolutions in the first and second step, having the first step a lower quality resolution, and the second step, a higher quality resolution. This approach takes advantage of the low computational cost of the *LR* covering to achieve a fast compaction, and using the *HR* covering to re-compact the layout, to improve the compaction quality. Since the pieces are already pre-compacted from the first step, the computational cost is reduced.

The second approach is the Two-Step approach (in the second section) which separates the normal optimization process into two distinct phases. The first phase compacts a set of large pieces, trying to adjust them well together but leaving sufficient space between them to position the small pieces. The second phase assigns the remaining small pieces to the holes generated in the first phase (between the large pieces), and compacts all pieces together. In order to tackle the infeasibilities generated in this problem, due to the overlapping that is caused by the placement of small pieces, the Post-Optimization phase (presented in the previous chapter) is tweaked to improve the number of feasible solutions generated.

The last approach is the Multi-Layer approach (in the third section), where pieces are divided (or distributed) into separate groups, and compacted sequentially into the layout, creating multiple

compaction layers. Each layer is addressed as an independent compaction problem, while considering the previously compacted pieces in the layout as fixed pieces, and with the current pieces being the only ones allowed to move.

The fourth section of this chapter presents the best results achieved by all presented approaches, for each instance. These results are compared to the best results in literature.

The final section contains the concluding remarks.

5.1 Multi-Resolution Approach

The Multi-Resolution approach was developed to address the high computational times obtained with the NLP approaches presented in the previous chapter. The computational cost was managed by selecting the right resolution considering its associated level of quality and number of circles, that would achieve a reasonable computational time. Since reducing the computational cost implies reducing the number of circles, and therefore, also reducing the quality of the approximation, this strategy imposes a limitation on the quality of the layouts that could be produced. Using a higher-quality representation would increase significantly the computational cost, but would allow a tighter layout, since the pieces would be able to adjust better to each other.

The main idea of this approach consists in using a multi-step approach that combines the strengths of the *LR* covering (achieving a compact layout very fast) and the *HR* covering (very good adjustment of pieces between each other), to overcome their disadvantages when used independently and gather their individual strengths.

5.1.1 Multi-Resolution Algorithm

The Multi-Resolution approach (denoted *LR+HR*) uses a *LR* covering initially to achieve a fast compaction of the layout, and after it, it uses a *HR* covering to adjust the pieces to each other. The type of covering is the same in both first and second steps of the normal optimization process. Since a great amount of time is used in the compaction process, the *LR* covering should be selected in order to be as fast as possible, while allowing the pieces to achieve a good relative position between each other. The *HR* covering must also be carefully selected in order to adjust the pieces to achieve an acceptable compaction quality, but without a significant impact on the computational cost. In order to improve this effect, one model variant may be selected in order to also provide the lowest computational cost in the first step, and another to provide the highest quality in the second step (while also considering the impact in the computational cost). The objective of this approach is either producing the same quality of layout compaction as the single use of *HR*, but with much less computational cost, or producing a layout compaction with higher quality but with similar computational cost of the independent use of *LR* covering.

The change in resolution from a lower quality to a higher quality can produce overlaps between circles of adjacent pieces. This situation is more frequent when using the *ICC* and *PCC*, since additional circles are used to better cover the piece, but it can also happen when using *CCC* in rare situations. When using *ICC*, the re-compaction with higher resolution will have to spread out

the pieces, thus increasing the length of the container. This is due to the large penetration depth allowed by the *LR ICC*, which is significantly reduced when using *HR ICC*. Using this approach with the *PCC* may increase or decrease its length of the layout (since it reduces both the protrusion distance and penetration depth), while using *CCC* usually reduces the length (due to the reduction in protrusion distance).

This approach is based on the same concept of the Post-Optimization approach used to address infeasible layouts, but focusing on the reduction of the computational cost while maintaining the high layout quality. This approach uses a different configuration, where the types of coverings are the same in both steps, the resolutions have a lower quality than those used in the Post-Optimization phase, and the model variants may be the same, or different in both steps, depending on the requirements.

5.1.2 Results and Discussion

The behavior of the Multi-Resolution approach results (*LR+HR*) is compared to the single-step NLP model compaction using both *LR* and *HR* resolutions. The results are shown in Table 5.1. Table 5.1 used *CCC* together with model variant *M2E* to test the *LR+HR* approach, in order to isolate the influences of each parameter (model variant, resolution and type of covering). Using *PCC* and *ICC* would produce infeasible layouts, preventing their direct comparison, and requiring using a Post-Optimization phase which would influence the results. For this reason, the *CCC* is used. Using a model variant for the first step, and another one for the second step would not allow comparing directly the independent *LR* compaction to the *HR* compaction, and would also difficult their comparison to the Multi-Resolution compaction (using *LR+HR*). Using the same model variant allows verifying the impact of each independent resolution compaction, and also comparing them to the compaction of both resolutions sequentially, in the Multi-Resolution Approach. The results in Table 5.1, regarding the *LR* and *HR* coverings compaction with model variant *M2E* and *CCC*, were also presented in the previous chapter, in section 4.4.3.

The layout compaction quality achieved by the *HR* covering is better, on average, than the compaction using *LR* covering, due to the lower waste of the former, but at the expense of a significantly higher computational cost. When comparing the *LR+HR* approach to the independent *LR* and *HR*, Table 5.1 shows that the *LR+HR* covering is able to achieve better average results, and also able to achieve the best solutions occasionally. The biggest advantage of *LR+HR* over the single *HR* compaction is the significant reduction in computational cost, while being able to maintain very close results, regarding the compaction quality, or even improve them. The *LR+HR* is able to achieve the best solution in instances *poly1a*, *poly2b*, *poly2b* and *poly3b*, while *HR* achieves the best in *jakobs1* and *poly3a*. This effect is also noticeable considering the average values. The improvement in the quality of the compaction can be explained by using two sequential compactions of the layout and allowing a better adjustment of the pieces. The solver compacts using *LR* on the first run, stopping at a local minimum. When the piece coverings are replaced by other piece coverings with higher approximation quality, the pieces are forced to re-adjust to each other, and the solver is able to search for a better local minimum than the one previously found.

Table 5.1: Circle Covering Multi-Resolution approach (*LR*, *HR* and *LR+HR*) (with *CCC* and *M2E*, 30 runs each).

		Obj. Function (l)			Avg.
Instance[a]	Resolution	Min.	Avg.	Max.	Time (s)
jakobs1	*LR*	15.73	17.14	20.17	18.07
(11.82)	*HR*	13.85	16.15	20.54	56.00
	LR+HR	14.12	15.62	19.95	30.63
poly1a	*LR*	16.04	17.63	19.09	9.00
(13.21)	*HR*	14.76	16.10	17.82	29.23
	LR+HR	14.66	16.24	18.22	15.17
poly2a	*LR*	30.94	32.84	34.58	111.10
(25.71)	*HR*	28.74	29.88	31.69	286.03
	LR+HR	28.09	29.69	31.43	172.47
poly2b	*LR*	33.64	35.78	38.51	121.40
(29.07)	*HR*	31.26	33.18	35.53	295.77
	LR+HR	30.85	32.72	34.56	187.13
poly3a	*LR*	46.67	47.96	49.65	249.57
(38.81)	*HR*	41.93	44.17	47.79	710.83
	LR+HR	42.31	43.62	45.09	461.20
poly3b	*LR*	46.37	48.03	55.86	266.47
(38.79)	*HR*	42.39	44.02	54.15	611.70
	LR+HR	41.99	43.68	45.78	449.47

[a] Reference value: best result in literature, considering length

By analyzing these results, this approach *LR+HR* shows the gains that can be achieved by starting the compaction using a *LR* approach and then using a *HR*. This has the additional benefit of being used on computationally expensive instances which can only be solved with *LR*, where the most promising solutions can later be improved using this approach.

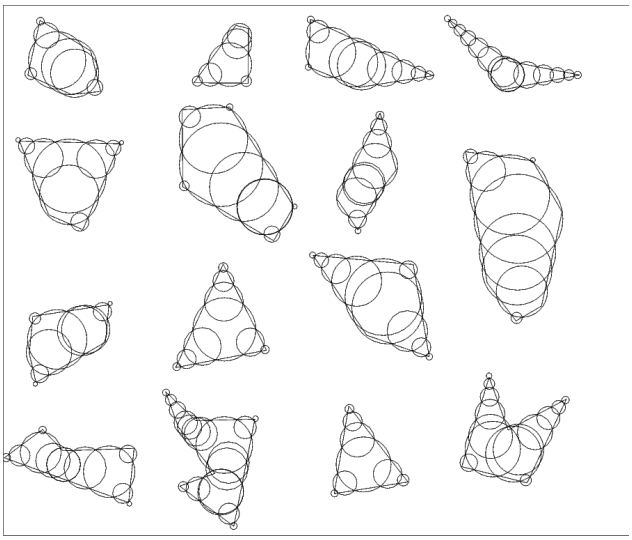
The average computational times presented in Table 5.1 show that using a *LR* covering for a first compaction, and using *HR* for the second compaction, is able to reduce significantly the total computational cost, and produce an equivalent or better compaction quality compared to the *HR*. Since the relative positions between the pieces have already been quickly determined in the first step of the *LR+HR* approach, the solver only needs to focus on adjusting the pieces in the second step. The penalty for using a higher resolution is significantly reduced since only minor adjustments to the pieces are required, thus being able to achieve a local minimum much faster. Considering the results presented in Table 5.1, if the *LR+HR* approach had been used with a model variant that focused on the computational speed, for the first step, the computational time could be further reduced. The difference between the computational time of the independent *LR* compaction and the *LR+HR* approach is the additional cost used by the second step with *HR* covering, since the independent *LR* compaction and the first step of the *LR+HR* are the same. It clearly shows the significant increase in the compaction quality by using a second step with *HR* and its impact in the computation time.

An illustration of the compaction process using the Multi-Resolution approach can be seen in Fig.5.1, with instance *poly1a* and *CCC*. In it is shown the impact caused by the selection of different resolutions for each step, and their influence on the final quality of the layout. The initial solution used can be seen in Figure 5.1a, with the *LR* covering. The compaction of the first step produces the layout seen in Figure 5.1b. The pieces are then replaced by a higher quality covering, which can be seen in Figure 5.1c, and their compaction in the second step produces the layout shown in Figure 5.1d. In this example, the *LR* layout produced in the first step achieves a length of 16.73 units and is improved in the second step to 15.22 units with *HR* covering. The reduction in the length of the layout is due to the higher quality covering, which reduces the waste of its covering when resolution is increased.

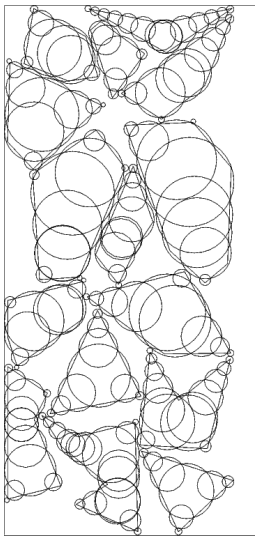
The results show that using a Multi-Resolution approach, combined with an appropriate type of covering and model variant can produce improved results when addressing the Nesting problem with continuous rotations, relative to the approaches presented in the previous chapter (single-step approaches). The selection of the appropriate configuration parameters is very important since it allows increasing the quality of the results, and reducing computational cost, which enables tackling problems with greater efficiency.

5.2 Two-Phase Approach

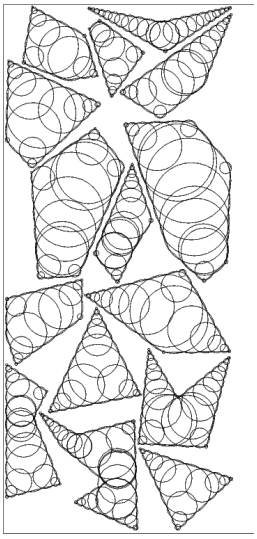
The Two-Phase approach follows a common compaction strategy to achieve layouts with good compaction quality which is based on the observation that the overall layout length is defined by the positions of the large pieces. For this reason, this approach starts by initially compacting



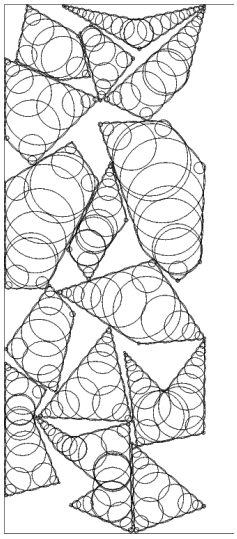
(a) *LR-CCC* initial solution.



(b) Compaction of *LR-CCC*.



(c) *LR CCC* substituted by *HR-CCC*.



(d) Compaction of *HR-CCC*.

Figure 5.1: Multi-Resolution approach *LR+HR*.

only the larger pieces, followed by the placement and compaction of the smaller ones among the already compacted larger pieces.

This approach is used to achieve better quality solutions, compared to the other results presented in this and previous chapters. By addressing the compaction process, using a Two-Phase approach, the main compaction process becomes divided into two simpler components, where the main structure of the layout is defined in the first phase. By focusing in the generation of high quality layouts in the first phase, through an extension of the objective function of the NLP models presented in chapter 4, the quality of the final layout is expected to increase, being fully feasible and with a very reduced length.

In the first phase, the compaction process tries to adjust the larger pieces between each other and also between the outline of the container, while compacting them until they create a good layout, but still with sufficient space. In the second phase, the remaining unplaced smaller pieces are assigned to the holes created in the first phase (empty space between larger pieces) and all pieces are compacted together.

The distinction between big and small pieces is done through their area, although using a combination of area, perimeter, and other geometrical characteristics may prove beneficial.

In the following subsections the overall description about the Two-Phase approach, and its NLP model extension, will be discussed. The computational experiments will follow, by allowing to analyze the results and discuss the impact that different configurations have on the final solution quality.

5.2.1 Two-Phase Approach Overall Description

The objective of the first phase is to generate a layout, consisting only of the larger pieces, with a good adjustment between each other and the container sides, while still having space between them to place small pieces. In order to control the layout length, the adjustment between pieces, and also to the container, a set of parameters are used, where *target* defines the minimum length L of the layout (allowing the layout to be compacted until it reaches the *target* value), P controls the adjustment between different pairs of pieces (how close pieces are to each other), and W controls the adjustment between pieces and the container outline (how close are pieces to the outline). These parameters were introduced into the objective function of the NLP models, presented in chapter 4, producing an extended NLP model that is used in this approach.

An illustration about the first parameter can be seen in Figure 5.2, where the pieces are forced to be compacted until they are all under the *Target* value. Since the pieces that are defining the current length of the layout (with the highest x coordinate) are pushed, it is normal that they are closer to the side that is pushing them (the same side of the *target* line) than the opposite side of the layout. The compaction shows a different behavior when the other parameters are being used, since they will increase or decrease this effect.

When the *target* value is too low, the pieces will be packed very tightly, thus reducing the empty space available (that forms the holes). If the holes are very small, the remaining pieces will not be able to be positioned inside any hole, and will have to be placed in the empty region of

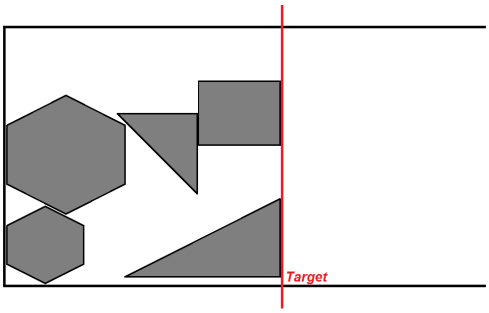
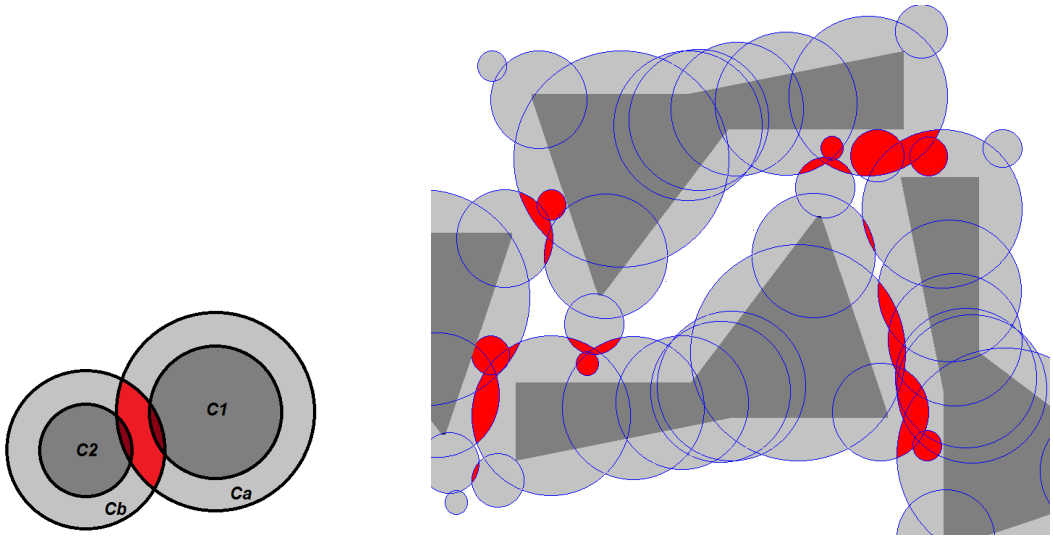


Figure 5.2: Layout derived from pushing pieces below *target* value.

the strip. When the *target* value is too high, the larger pieces will not be able to adjust well to each other and to the container outline (therefore not creating a good initial layout) and preventing further improvement of the final solution quality compared to the normally obtained (since the pieces placement positions resemble a random initial solution).

The other two components (P and W) deal with the minimization of the distance between pieces, and between the pieces and the container outline. This is achieved by simulating an attraction effect between pieces, by extending the circles of each one of the pieces circle coverings, by a given distance, and maximizing their overlap. An illustration of this effect can be seen in Figure 5.3a, where the extended circles (Ca and Cb) obtained from their respective pieces circle covering ($C1$ and $C2$) show the overlapping region that is to be maximized, while preventing the overlap between both original circles. Figure 5.3b shows the multiple expanded circles from different pieces where some are overlapping. By maximizing the overlap of these circles, the pieces are drawn together, forcing them to adjust to each other, in order to minimize their total separating distance. This effect is the same when considering the attraction between the pieces and the container outline. This approach was inspired on one approach described in (Alves et al., 2012), where the No-Fit-Polygons of the pieces are expanded and their overlap is used to better adjust the pieces. In (Alves et al., 2012) many different constructive procedures are presented, all based on the No-Fit-Polygon, where many strategies for sorting selection, placement and evaluation of the placement of pieces are proposed and discussed.

The component P deals with the minimization of the distance between pieces, for pieces that are within a specific range from each other. Figure 5.4 contains an illustration about the effect of this component, where the expanded representation that is overlapping is maximized, bringing the pieces together. This effect only works on pieces that have their extended representations currently overlapping, being blind to pieces that are not close to each other. Figure 5.4a shows the initial solution of a layout, where the extended representations of the pieces overlap, but only between certain pairs of pieces. Their overlap represent the current attraction between the pieces. Figure 5.4b clearly shows the effect of the attraction between pieces on the final layout, where the pieces have been pulled together. This component allows the pieces to adjust properly to each other, minimizing their separating distances, and creating holes (empty regions) near the outline of the container. This component is used after the first component (minimize layout below *target*



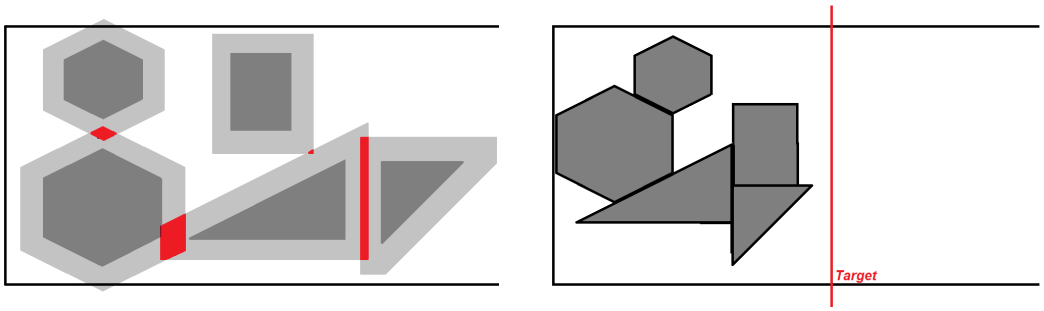
(a) Attraction between expanded circles.

(b) Attraction between multiple expanded circles.

Figure 5.3: Example of attraction effect.

value) is achieved.

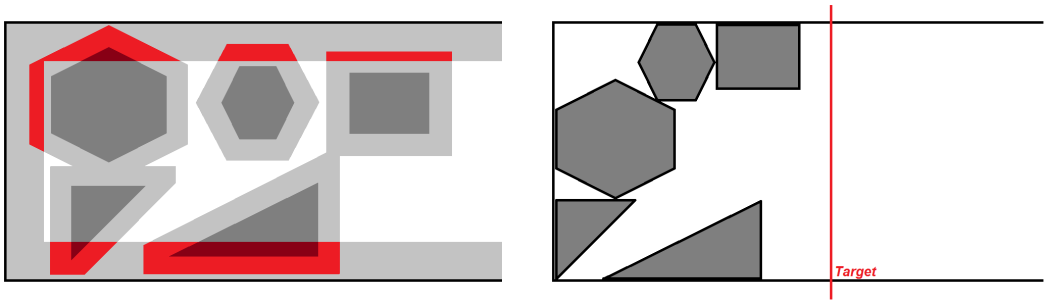
The component W deals with the attraction effect between pieces and the container outline, for pieces that are within a certain range from the outline. Figure 5.5 presents this effect, where the expanded representation of both the pieces and the container outline is maximized (if overlapping), and pulling the pieces toward the outline. Figure 5.5a shows the overlap between the expanded representation of both pieces and the container outline, which is to be maximized. With the maximization of their overlap the pieces will move to the external regions of the layout (but remaining inside the container) and adjust themselves to the outline. An illustration of a layout produced with this component can be seen in Figure 5.5b where the pieces are placed near the outline of the piece and the inner regions of the container have empty space. This component is



(a) Attraction between all pieces (approximated representation).

(b) Layout derived from attraction between all pieces.

Figure 5.4: Objective function piece adjustment component.



(a) Attraction between pieces and container sides (approximated representation).

(b) Layout derived from attraction between pieces and container sides.

Figure 5.5: Objective function container sides adjustment component.

used after the first component (minimize layout below *target* value) is achieved.

The three components (L, P and W) are only active in the first phase of the Two-Phase approach, being used with the extended NLP model. Achieving a balanced trade-off between all these components allows to produce a layout with the larger pieces well adjusted, and enough space to place the smaller pieces without significant changes to the relative positions of the pieces.

The second phase consists in assigning the remaining smaller pieces to the holes created in the first phase, followed by their compaction together with the previously placed larger pieces, while using the normal NLP model presented in chapter 4. The hole assignment process starts by analyzing the size of the pieces, considering their area. In order for a piece to fit into a hole, the first thing to determine is the area of the piece relative to the area of the hole. If the piece area is bigger than the hole area, it allows discarding this combination and if the piece area is smaller, it still does not ensure that the piece fits inside the hole, since they may have incompatible shapes.

The assignment between pieces and holes is based on a greedy approach, being the assignment of small pieces to smallest holes available. Finding if a piece fits inside the hole, and if affirmative, computing the best placement position and orientation to do so is a very difficult task.

This approach distinguishes pieces based on a single criteria, their area. After the assignment of the pieces to holes, the pieces are placed in the center of the hole, in a random orientation. This does not ensure that the piece will not overlap the surrounding pieces, but it is expected that the amount of overlap will be reduced. If the area of the piece is big enough to support another piece or pieces, they are placed on the remaining space of the hole, positioned in a way also to reduce overlap with the previously placed pieces into the same hole.

5.2.2 Non-Linear Programming Model Extension for the First Phase

The generation of the first phase layout, with only the larger pieces, requires a different model than the ones previously presented in chapter 4, since they aim to achieve the best layout compaction possible. In the first phase of this approach, the big pieces are positioned in order to achieve a good adjustment between them, creating the main structure of the layout. In order to produce a

good quality layout, three different parameters were included in the objective function, which must be controlled during the first phase: *target* allows to define the minimum length that the layout will be compacted, in order to leave enough empty space to position the small pieces; P allows controlling the adjustment between pairs of pieces and W allows adjusting the pieces to the outline of the container. The objective function of the normal NLP model as presented in chapter 4 only considered the parameter l which represented the length of the container.

The original objective function of the NLP model consists only in minimizing the length l of the strip. This objective function acts upon the circle with the highest value (placement position plus radius) on the x-axis, forcing its x-axis value to be reduced. This effect pushes the circle that is currently limiting the reduction of the length of the strip. Considering the extended NLP model objective function, each one of the parameters has a specified weight assigned in order to control them. The extended objective function is defined as minimizing Eq. 5.2, where α_L, α_P and α_W are the weights assigned to the components L, P, W, respectively. L represents the length l of the strip minus a given *target* value, P represents the attraction between pairs of pieces, and W represents the attraction between the pieces and the container. The values that these components have assigned to them have a strong influence on the behavior of the model.

Figure 5.6 contains the formulation of the extended NLP model, used in the first phase of the Two-Phase approach, while sharing some constraints of models M1 and M2.

$$\text{GRAVP\_P}_{i,j,k,h} = \text{GRAVP\_R}^2_{i,j,k,h} - \text{GRAVP\_X}^2_{i,j,k,h} - \text{GRAVP\_Y}^2_{i,j,k,h} \tag{5.10}$$

$$\text{GRAVP\_R} = R_{k_i} + R_{h_j} + vxp \tag{5.11}$$

$$\text{GRAVP\_X}_{i,j,k,h} = (x_{k_i} - x_{h_j})^2 \tag{5.12}$$

$$\text{GRAVP\_Y}_{i,j,k,h} = (y_{k_i} - y_{h_j})^2 \tag{5.13}$$

$$\text{GRAVW\_XMIN}_{i,k} = (R_{k_i} + vxp - x_{k_i}) \tag{5.14}$$

$$\text{GRAVW\_YMAX}_{i,k} = (R_{k_i} + vxp - (Wd - y_{k_i})) \tag{5.15}$$

$$\text{GRAVW\_YMIN}_{i,k} = (R_{k_i} + vxp - y_{k_i}) \tag{5.16}$$

$$\text{GRAVP\_P}_{i,j,k,h} = \text{GRAVP\_R}^2_{i,j,k,h} - \text{GRAVP\_X}^2_{i,j,k,h} - \text{GRAVP\_Y}^2_{i,j,k,h} \tag{5.17}$$

$$\text{GRAVP\_R} = R_{k_i} + R_{h_j} + vxp \tag{5.18}$$

$$\text{GRAVP\_X}_{i,j,k,h} = x_{k_i} + cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} - x_{h_j} - cos(A_{h_{0,j}} + \theta_h) \times D_{h_{0,j}} \tag{5.19}$$

$$\text{GRAVP\_Y}_{i,j,k,h} = y_{k_i} + sin(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} - y_{h_j} - sin(A_{h_{0,j}} + \theta_h) \times D_{h_{0,j}} \tag{5.20}$$

$$\text{GRAVW\_XMIN}_{i,k} = (R_k + vxp - x_k - \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}}) \tag{5.21}$$

$$\text{GRAVW\_YMAX}_{i,k} = (R_k + vxp + y_k + \cos(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}} - Wd) \tag{5.22}$$

$$\text{GRAVW\_YMIN}_{i,k} = (R_k + vxp - y_k - \sin(A_{k_{0,i}} + \theta_k) \times D_{k_{0,i}}) \tag{5.23}$$

$$\text{minimize} \quad F_{2P} = \alpha_L \times L + \alpha_P \times P + \alpha_W \times W \tag{5.1}$$

$$\text{subject to:} \quad L = \max(0, target - l), \tag{5.2}$$

$$P = \sum_{i=0}^{C_k} \sum_{j=0}^{C_h} \sum_{k=0}^{N} \sum_{\substack{h=0 \\ k \neq h}}^{N} \left\{ \left(\max \left[0, \text{GRAVP\_P}_{i,j,k,h} \right] \right) \right\}^2 \tag{5.3}$$

$$W = \sum_{i=0}^{C_k} \sum_{j=0}^{C_h} \sum_{k=0}^{N} \sum_{\substack{h=0 \\ k \neq h}}^{N} \{ (\max \left[0, \text{GRAVW\_XMIN}_{i,j,k,h} \right])^2 + \\ + (\max \left[0, \text{GRAVW\_YMAX}_{i,j,k,h} \right])^2 + (\max \left[0, \text{GRAVW\_YMIN}_{i,j,k,h} \right])^2 \} \tag{5.4}$$

Constants Definition: (5.5)

equations 5.10 to 5.16 for model M1 or equations 5.17 to 5.23 for model M2

Non-Overlapping Constraints: (5.6)

equations 4.2 for model M1 or equations 4.11 for model M2

Containment Constraints: (5.7)

equations 4.3 to 4.6 for model M1 or equations 4.12 to 4.15 for model M2

Piece Integrity Constraints: (5.8)

equations 4.7 to 4.8 only for model M1

$$x_{k_i} \in R, y_{k_i} \in R, \theta_k \in R, l \geq 0 \tag{5.9}$$

Figure 5.6: Non-Linear Programming Model Extension for the First Phase.

The objective function presented in Eq. 5.1 contains three components with weights assigned to them, where the aim is to minimize their sum. The first component, L (Eq. 5.2) contains the current value for the length l, where a *target* value is subtracted in order to disable this component when l is below *target*. The second component P (Eq. 5.3) contains the distance that separates pairs of pieces that are not separated by more than the defined vxp value (as seen in Eq. 5.11). The last component, W (Eq. 5.4), contains the distance that separates the pieces from the container outline, also only distances not bigger than the specified value vxp. The variables α_L, α_P and α_W assign weights to each component, to control the impact of the L, P and W components on the objective function. The effects from P and W are only active for pairs of pieces closer than vxp and for pieces closer to the outline than vxp. Balancing these three components is the key to achieve good quality solutions when solving the Nesting problem with continuous rotations.

In order to use the variables α_L, α_P and α_W with the L, P and W components, the components had to be normalized (i.e., their values should be variable only on the interval from 0 to 10). In order to achieve this, the lower and upper bounds for each of the components were determined, and based on their values, the normalization was done through a linear function. The computation of the lower bound for the L component considered the container length for the perfect fit of the pieces, while the upper bound used the initial container length. The other two components, P and W, had their lower bound determined (zero, which is when no pieces are within range of one another), while an approximation to their upper bound was determined by trial and error (since it is difficult to compute the total of multiple overlaps).

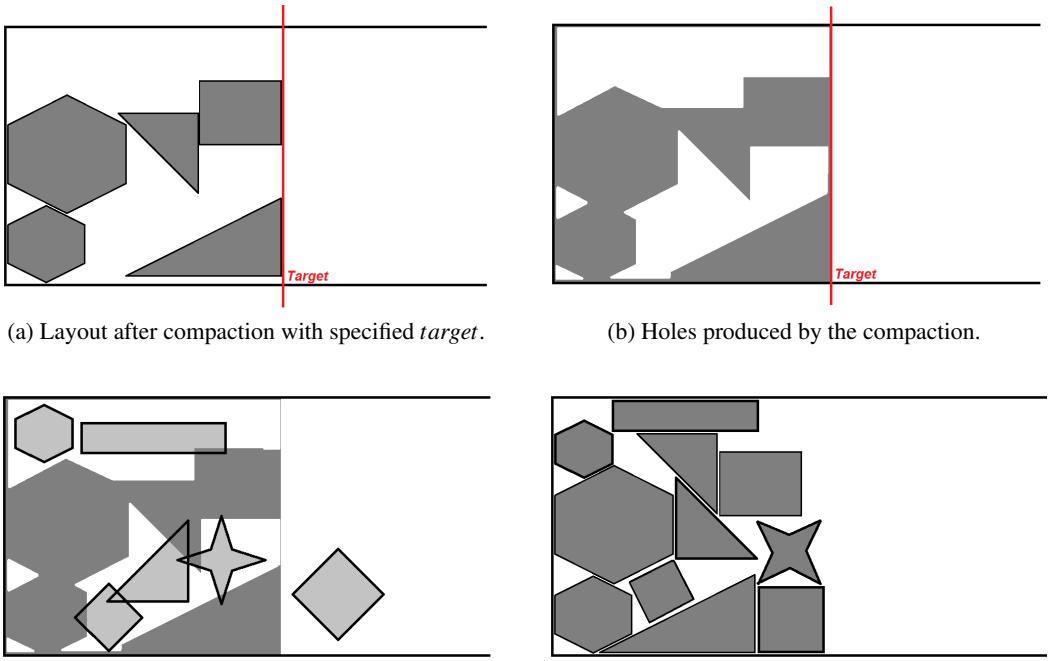
The Eq. 5.6, Eq. 5.7 and Eq. 5.8 are the natural constraints of the NLP model. Depending on the model, M1 or M2, they have a different formulation, since the models are either based on circles or based on pieces, respectively. The terms present in Eq. 5.3 are obtained from the Eq. 5.10. Depending on the model, M1 or M2, different equations are replaced in the terms of Eq. 5.4. The equations used with model M1 are Eq. 5.10 to Eq. 5.16 and with model M2 are Eq. 5.17 to Eq. 5.23.

5.2.3 Second Phase Hole Fill and Layout Compaction

The entire Two-Phase approach can be summarized in Figure 5.7. The first phase, where the largest pieces are compacted and the main structure of the layout is defined (considering the *target* value, the adjustment between pieces and the adjustment between pieces and the container), is shown in Figure 5.7a. The holes produced during the first phase can be seen in Figure 5.7b, with the empty region in the strip available at the right side of the *target* line.

In the second phase, a simplified illustration about the assignment process (placement of pieces into holes) is shown in Figure 5.7c, considering the previously placed pieces and the remaining empty space.

The hole assignment and piece placement process is done with the assistance of a grid that represents a discretization of the layout, considering a certain approximation quality, which contains the information about the available and unavailable placement positions. This process starts by defining the regions in the grid that are empty and those that are non-empty, using 0 for the



(a) Layout after compaction with specified *target*.

(b) Holes produced by the compaction.

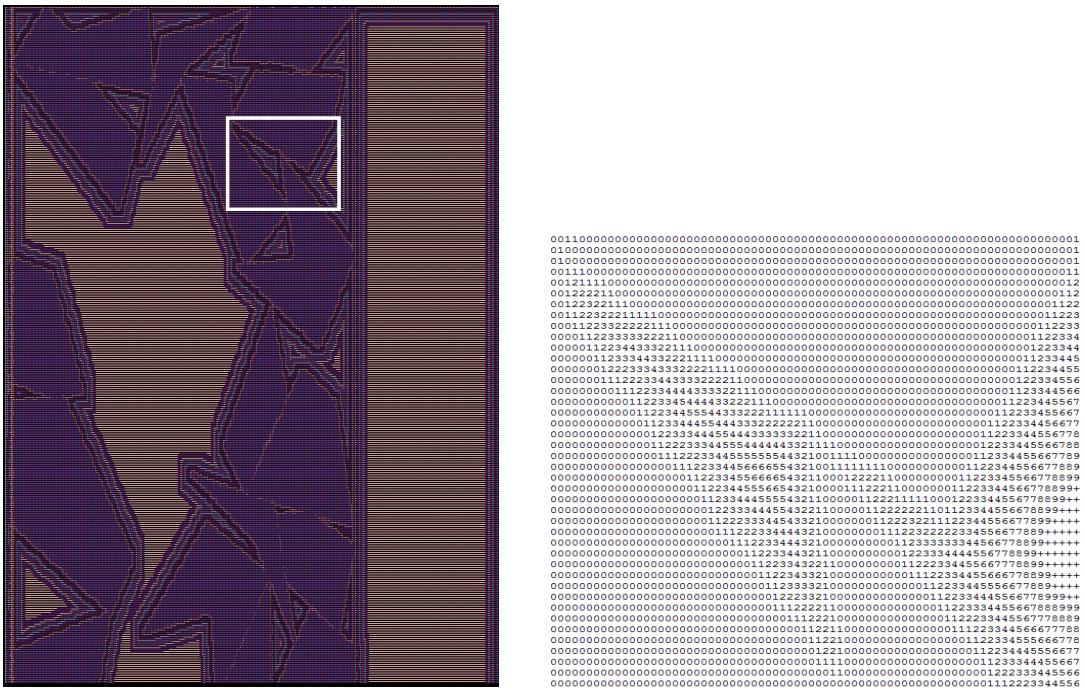
(c) Assignment of small pieces to available holes.

(d) Final layout.

Figure 5.7: Assignment procedure of small pieces to holes.

first case, and 1 for the second. The second step uses a distance transform algorithm to attribute to each empty block the value of its closest distance to a non-empty block (as illustrated in Figure 5.8, where 5.8a contains the grid and 5.8b its zoomed region). The following step removes all grid values above a given distance, which allows excluding small holes and defining distinct separate empty regions, and the last step identifies the individual empty regions while determining their block with highest distance to a non-empty block, in order to place pieces in them. The placement of pieces in the points most distant from the other pieces (which can be considered the center of the holes) reduces the chance of generating severe overlaps. After determining the individual empty regions, the pieces are assigned based on a rule that defines that the smaller pieces will be placed in the smallest holes where they can fit (considering their different areas). After a piece is placed into the hole (in a random orientation), the grid is recalculated, in order to update the information about the holes. This process continues until all pieces are placed.

The remainder of the second phase consists in the compaction of all pieces using the normal NLP model, producing the final layout solution, as seen in Figure 5.7d. The configuration of the model is the same as during the first phase (considering resolution, type of covering and model variant) except for the introduction of a value for the upper bound of the objective function. This value (the upper bound of the objective function) refers to the length of the layout l, and is defined as equal or slightly bigger than the length of the layout achieved in the first phase. The specification of this value is important due to the natural behavior of the solver with the normal NLP model. When compacting a layout with overlaps, the solver initially gives preference to the correction



(a) Grid with nearest distance to non-empty block. (b) Zoomed region.

Figure 5.8: Hole detection procedure.

of overlaps, by addressing the feasibility of the *Non-Overlapping constraints*, while the objective function takes a secondary role. This behavior increases the length of the layout while the overlaps are being corrected and later, when the solution is feasible, the objective function gains relevance and its length starts decreasing with the minimization of the layout length. Since the NLP model converges to local minima, after the expansion of the layout and its following contraction, the model stops and returns a solution as soon as it reaches the first feasible local minima. In order to force the solver to intensify its search for a feasible local minimum solution of better quality, the layout length is prevented from expanding beyond the defined upper bound limit. The pieces are forced to find feasible placement positions within the limited space that they have, generating good solutions when they succeed, but at a cost of an increased number of infeasible solutions, comparatively to when no such limit is imposed. If the maximum layout length is set too high, the layout will expand too much and possibly generate bad solutions. If it is set too low, the overlaps might not be corrected, leading to infeasible solutions. Finding the right value for this parameter is a difficult task, due to the dependence on the layout obtained in the first phase.

5.2.4 Results and Discussion

The initial configuration of the parameters, used in the Two-Phase approach, considered the results of previous computational experiments, when testing the model variants with different resolutions and types of circle coverings. In the first phase, the *target* value was defined as equal to the length of the perfect fit of all pieces in the layout, taking into account the specified width (even

if no perfect fit exists for the layout). This was done by computing the total area of the pieces, which allows to determine the length of a perfect fit of the pieces in the strip when considering the width of the strip. The objective function components were configured with the weights $\alpha_L = 100$, $\alpha_P = 10$ and $\alpha_W = 1$, to give preference to the piece compaction until it reaches the desired *target* length. When this happens, this component is discarded, while the other two components gain preference, with the adjustment between pieces being preferred to the adjustment between pieces and the container.

The selection of the values for the parameters *target*, α_L, α_P and α_W was based on preliminary tests that indicated which values could allow achieving good solutions. The tests were not exhaustive, since this approach is currently being developed, but their results allowed drawing some conclusions. The *target* value is the main influence in the area available for the holes that are created in the first phase (also due to α_L parameter being given priority over the others), while the variation between α_P and α_W produce different types of holes. A value higher in α_P leads to pieces that are closer together leaving the bigger holes concentrated near the outline of the container, while a bigger α_W concentrates the pieces near the outline of the container, leaving the bigger holes at the center. This type of behavior was observed in most preliminary experiments, some of them reflecting it more than others. This behavior is heavily dependent on the initial position and orientation of the big pieces. Due to the difficulty selecting the exact values that could produce the best results for a given instance, the values for these parameters were configured in order to express the desired behavior: big pieces adjusted to each other with a layout length closer to value of the perfect fit with all pieces, while giving preference to the adjustment between all pieces instead of the adjustment to the container.

For the second phase, the *target* value is defined as zero, and the objective function weights remain with the same values. This configuration still gives preference to the minimization of the length of the layout, but still produces a reduced effect for the adjustment of the pieces, and an even smaller effect for the adjustment between pieces and the outline of the container. The assignment of the remaining pieces to the holes is based on the rule of assigning the smallest pieces to the smallest available holes, with a tolerance of 5% between the area of the piece relative to the hole. Orientations of the placed pieces are arbitrary, and the reference points of the pieces are placed in the positions inside the holes that are further away from any unavailable placement position. Since the small piece placement into holes is a greedy approach, and the exact placement method is very basic, it is normal that most small pieces have overlaps with surrounding pieces.

The determination of the hole assignment rule was also based on preliminary tests. Several possibilities were considered, such as assigning the largest pieces to the largest holes, assigning the smaller pieces to the smallest available holes, or assigning them at random. The problem of determining if a piece fits on a certain hole was also considered (it is addressed by comparing their area), and if pieces can be placed into holes where they do not fit (since the compaction procedure will adjust the pieces, the overlaps may be removed). There was also the difficulty in determining the best orientation and position, inside the selected hole, in order to minimize overlap. Since it is hard to deduce which configuration will produce the best results, due to the many configuration

parameters that can be defined, and due to the significant impact that the initial solution has on the resulting layout, the preference was given to the placement of small pieces into the small holes, while the remaining pieces are placed on remaining space in the strip, when no more valid holes exist.

The second phase compacts all pieces together, trying to remove the overlaps between each other, and while also limiting the maximum increase in the strip length caused by the correction of overlaps and adjusting pieces. If the layout length is fixed to the value obtained in the first phase, the pieces are forced to adjust to each other (in order to find a non-overlapping configuration), without increasing the length of the layout (which creates greater difficulties for the solver, and may end up producing an infeasible layout). In order to intensify the focus of the solver in achieving a feasible solution without degrading the quality of the compaction significantly, the maximum length of the layout was allowed to expand up to a maximum of 120% of the perfect fit. If the produced layout contains overlaps, a Post-Optimization phase is used to address them, using the *CCC* with a higher resolution.

The selected preferences do not ensure that the best results will be achieved, due to the lack of exhaustive testing for each one. Better results can be achieved with further development of this approach and selection of appropriate configuration parameters. The current configurations were selected just to demonstrate how the approach works.

The first set of tests for both phases was done using *HR* and *PCC* together with *M1E*. The Post-Optimization phase uses *CCC* with the model variant *M1E* to correct infeasible layouts and also improving the feasible ones. The impact that two types of resolution have when correcting the layouts with the Post-Optimization phase is shown in Table 5.2. The selection of model variant *M1E* is due to its expected lower computational cost.

The results produced by the Two-Phase approach with Post-Optimization (as seen in Table 5.2) show that the used configuration is not able to consistently return feasible solutions. When the layouts are re-compacted using the Post-Optimization phase to correct the infeasibilites, only a reduced number of them are able to become feasible layouts, which causes great difficulty in achieving good results. This difficulty in solving overlaps is derived from the selected configuration parameters, being the first the type of covering used in the Two-Phases (*PCC*) which allows the placement of pieces very close to each other while allowing for some small regions of the pieces to overlap. The second parameter is derived from the limitation of the expansion of the container length l, which forces the pieces to adjust to each other, inside a limited container. This creates overlaps between pieces that the model variant *M1E* cannot correct. The use of the model variant *M1E* in the Post-Optimization phase confirms that it is not very efficient in removing the overlaps. Considering the resolutions used in the Post-Optimization phase, it can be seen that when using *VHR* over *HRP* there is an improvement in the average compaction quality and its best solution, but with a significant impact on the computational time and a reduced number of feasible layouts. The difference in compaction quality is due to the reduction in excess covering between the pieces in the layout, while the difference in the number of feasible solutions is due to numerical precision errors created by using very small circles.

Table 5.2: Two-Phase Compaction using *HR-PCC* and Post-Optimization Phase using *HRP-CCC* and *VHR-CCC*, 20 runs each.

					Obj. Function (F)			Avg.	# Feas.
Instance[c]	Phase	Model	Res.	Cov.	Min.	Avg.	Max.	Time (s)	Layouts
poly1a	2PH[a]	*M1E*	*HR*	*PCC*	14.07	14.81	17.15	6	0
(13.21)	P-OPT[b]	*M1E*	*HRP*	*CCC*	13.88	14.70	15.62	3	7
			VHR		13.57	14.41	15.58	4	3
poly2a	2PH [a]	*M1E*	*HR*	*PCC*	24.16	26.15	28.07	70	0
(25.71)	P-OPT[b]	*M1E*	*HRP*	*CCC*	26.60	27.32	28.04	13	2
			VHR		26.00	26.00	26.00	10	1
poly2b	2PH[a]	*M1E*	*HR*	*PCC*	26.13	28.41	31.11	93	0
(29.07)	P-OPT[b]	*M1E*	*HRP*	*CCC*	29.72	29.72	29.72	45	1
			VHR		29.41	29.41	29.41	36	1
poly3a	2PH[a]	*M1E*	*HR*	*PCC*	36.17	37.94	40.40	192	0
(38.81)	P-OPT[b]	*M1E*	*HRP*	*CCC*	39.03	39.94	40.85	22	2
			VHR		40.82	40.82	40.82	104	1
poly3b	2PH[a]	*M1E*	*HR*	*PCC*	36.32	38.08	40.79	176	0
(38.79)	P-OPT[b]	*M1E*	*HRP*	*CCC*	39.08	39.08	39.08	52	1
			VHR		38.94	38.94	38.94	22	1

[a] Values considering all solutions
[b] Values considering only feasible solutions
[c] Reference value: best result in literature, considering length

The use of *PCC* leads to very tight layouts, in the two phases of the approach, which are very difficult to correct (by removing overlaps) in the Post-Optimization phase due to the limited movement of the pieces (which is restricted by the width of the container). Since the Two-Phase approach is based on placing pieces with small overlaps, to force them to adjust to each other, using coverings as the *PCC* or *ICC* creates layouts that are too compact to be successfully corrected.

In order to improve results, another configuration was selected, with the Two-Phase approach using *CCC* with *M1E* model variant and *HR* resolution, and the Post-Optimization phase using *CCC* but with model variant *M2S* and *HRP*. The *CCC* in the Two-Phase approach and the *M2S* model variant with *HRP* in the Post-Optimization phase were chosen to produce a higher number of feasible solutions. The results of the computational experiments with this configuration can be seen in Table 5.3.

Table 5.3 shows that the number of feasible solutions increased significantly, although only a few are produced in the Two-Phase approach, even with *CCC* (due to the overlaps between pieces). Only a few layouts are unable to be corrected on the Post-Optimization phase, due to numerical precision errors, and also due to significant overlap between pieces (solver unable to determine the best path that minimizes the overlap, within its maximum number of iterations). Comparing the quality of the layouts, the change from *PCC* to *CCC* reduced the quality of the feasible layouts, in both the best and average solutions, while also increasing the computational cost of the Post-Optimization phase. While this configuration produces many more feasible solutions, using

Table 5.3: Two-Phase Compaction using *HR-CCC* and *M1E* and Post-Optimization Phase using *HRP-CCC*, with *M2S*, 30 runs each.

					Obj. Function (F)			Avg.	# Feas.
Instance[c]	Phase	Model	Res.	Cov.	Min.	Avg.	Max.	Time (s)	Layouts
poly1a	2PH[a]	*M1E*	*HR*	*CCC*	14.59	15.20	15.92	6.70	10
(13.21)	P-OPT[b]	*M2S*	*HRP*	*CCC*	14.10	14.70	15.40	12.32	28
poly2a	2PH[a]	*M1E*	*HR*	*CCC*	28.05	28.36	28.73	38.50	4
(25.71)	P-OPT[b]	*M2S*	*HRP*	*CCC*	26.83	27.83	29.60	214.30	25
poly2b	2PH[a]	*M1E*	*HR*	*CCC*	29.93	31.23	32.13	61.00	5
(29.07)	P-OPT[b]	*M2S*	*HRP*	*CCC*	29.35	30.68	32.33	206.25	27
poly3a	2PH[a]	*M1E*	*HR*	*CCC*	40.65	42.25	43.13	100.00	5
(38.81)	P-OPT[b]	*M2S*	*HRP*	*CCC*	39.52	41.10	42.72	718.65	21
poly3b	2PH[a]	*M1E*	*HR*	*CCC*	41.75	41.75	41.75	90.00	1
(38.79)	P-OPT[b]	*M2S*	*HRP*	*CCC*	40.07	40.77	41.39	666.14	22

[a] Values considering all solutions
[b] Values considering only feasible solutions
[c] Reference value: best result in literature, considering length

another model variant may be able to produce better quality solutions while using the same covering type and resolutions. To address this, the same configuration was run, but using model variant *M2E* on the Two-Phase approach, with *HR* and *CCC*. This configuration was expected to assist in producing better layouts that also produce many feasible solutions with the Post-Optimization approach, with an acceptable computational cost. The model variant *M2E* is more computationally expensive than the variant *M1E* but it also returns better average solutions. Table 5.4 contains the results for the computational experiments with this configuration. The Post-Optimization has the same configuration, using model variant *M2S*, together with *HRP*.

The results in Table 5.4 show that using the model variant *M2E* over *M1E* does not bring significant benefits, since one model variant produces better results on some instances, while the other model produces better results on other instances. The results from both model variants are similar, except when comparing their computational cost and feasible solutions. There is a very large increase in the computational cost by using *M2E* instead of *M1E* although the number of feasible solutions is significantly higher for the Two-Phase approach. The computational cost used in the Post-Optimization phase is only used to improve most solutions since they are already feasible, which allows a significant reduction in the computational cost of the Post-Optimization phase. In order to improve the quality of the solutions without decreasing the feasibility and increasing the computational cost significantly, a change in the resolution of the Post-Optimization to *VHR* could allow better solutions being achieved. The chance of infeasible solutions (caused by numerical precision problems when comparing very small circles) may be lower due to most layouts used in the Post-Optimization phase already being feasible solutions. The impact of the increased resolution used with model variant *M2S* in the Post-Optimization phase can be seen in Table 5.5.

Table 5.4: Two-Phase Compaction using with Post-Optimization Phase together with model *M2*, 30 runs each.

					Obj. Function[a] (F)			Avg.	# Feas.
Instance[b]	Phase	Model	Res.	Cov.	Min.	Avg.	Max.	Time (s)[a]	Layouts
poly1a	2PH	*M2E*	*HR*	*CCC*	14.21	15.39	17.11	118	26
(13.21)	P-OPT	*M2S*	*HRP*	*CCC*	14.23	15.08	16.70	13	30
poly2a	2PH	*M2E*	*HR*	*CCC*	27.40	29.61	31.97	1251	20
(25.71)	P-OPT	*M2S*	*HRP*	*CCC*	26.64	28.32	30.28	184	30
poly2b	2PH	*M2E*	*HR*	*CCC*	30.86	33.34	37.27	1700	19
(29.07)	P-OPT	*M2S*	*HRP*	*CCC*	30.02	31.59	33.54	269	30
poly3a	2PH	*M2E*	*HR*	*CCC*	42.02	44.74	46.77	4784	12
(38.81)	P-OPT	*M2S*	*HRP*	*CCC*	39.58	41.82	44.32	806	30
poly3b	2PH	*M2E*	*HR*	*CCC*	41.63	45.16	59.07	5790	18
(38.79)	P-OPT	*M2S*	*HRP*	*CCC*	39.65	41.46	43.58	922	29

[a] Values considering only feasible solutions

[b] Reference value: best result in literature, considering length

Table 5.5: Two-Phase Compaction with Post-Optimization Phase using *HRP* and *VHR*, for instance *poly1a*, with all using *CCC*, 30 runs each.

					Obj. Function[a] (F)			Avg.	# Feas.
Instance[b]	Phase	Model	Res.	Cov.	Min.	Avg.	Max.	Time (s)[a]	Layouts
poly1a	2PH	*M2E*	*HR*	*CCC*	14.21	15.39	17.11	118	26
(13.21)	P-OPT	*M2S*	*HRP*	*CCC*	14.23	15.08	16.70	13	30
			VHR		13.59	14.66	16.97	77	26

[a] Values considering only feasible solutions

[b] Reference value: best result in literature, considering length

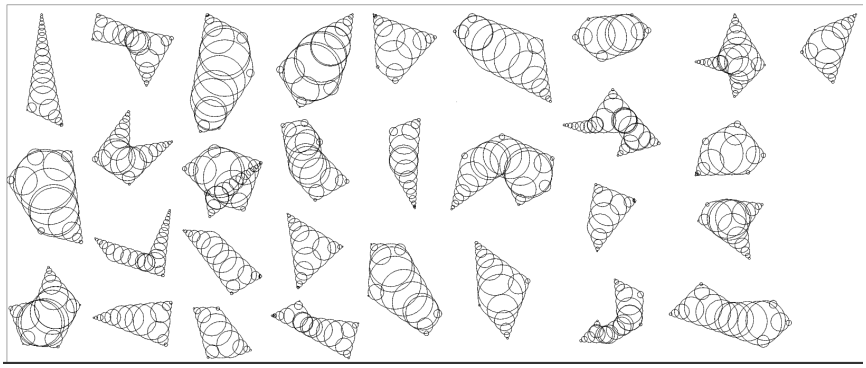
Unfortunately, the *VHR* covering could not be compared with *HRP* using *M2S* for all selected instances due to its large number of constraints, which the solver has problems supporting. The model variant *M2S* with *VHR* covering could only be tested with instance *poly1a*. Table 5.5 shows that the increase in the quality of approximation in the Post-Optimization phase leads to a small reduction on the minimum and average length of the layout, but with a significant increase of the computational cost. The growth in computational cost could be much higher if the layouts used in the Post-Optimization phase were not initially feasible. For the selected configuration, using a *VHR* covering in the Post-Optimization phase seems to produce very good quality solutions with a minor impact on the number of feasible solutions. Unfortunately, this cannot be confirmed for bigger instances due to limitations in the maximum number of constraints of the solver, using this resolution.

An illustration of the Two-Phase approach, using results from these computational experiments, can be seen in Fig. 5.9. This sequence starts by taking an initial solution composed of big pieces (Fig. 5.9a), compacting them until they reach a specified *Target* value (Fig. 5.9b and Fig. 5.9c), and placing the remaining small pieces in the empty spaces between the big pieces (Fig. 5.9d and Fig. 5.9e). The positions where the pieces are placed depend on the hole assignment and placement method, which may produce placement positions where the small pieces overlap the big pieces. The final step is the compaction of all pieces (Fig. 5.9f), and correction of the layout if it is infeasible, or improvement if it is feasible, using the Post-Optimization approach.

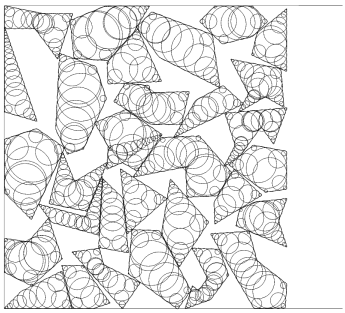
This approach enables achieving layouts with high compaction within a reasonable computational time, depending on the configuration used. This effect is derived from running the model in two steps, by compacting some pieces in the first step, and placing and compacting the remaining on the second. The Post-Optimization phase is also useful to address infeasibilities on the layout, but further increasing the computational cost. More extensive computational experiments need to be done, for each step of this approach, in order to determine the configuration that can lead to the best results.

The first effect that can be noted in this approach is the great impact that the initial solution has on the results. The initial solution, for the first phase, consists of a selection of a certain number of big pieces, which are selected based on their area. Other methods for piece selection may allow determining better pieces to be placed in this phase, which also can have a positive impact on the results of the first phase. The initial position and orientation of the big pieces have a great effect on the layout produced in the first phase. A single piece may prevent the layout to achieve the desired *target* value, which also reduces the effect of the other components of the objective function. The remaining variables that influence the layout produced in the first phase are derived from the chosen *target* value, and the weights assigned to each component of the objective function, such as the length minimization, piece adjustment and container adjustment components. Each one of these parameters, including the initial solution have a huge influence in the shape and number of holes generated.

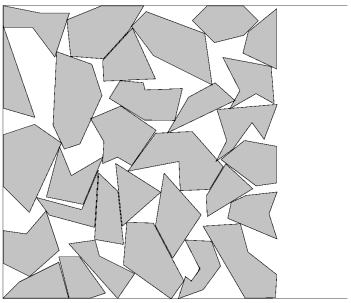
The procedure used to assign pieces to holes is very basic, since it only verifies the difference in area between pieces and holes, while placing the piece in a random orientation on the most far



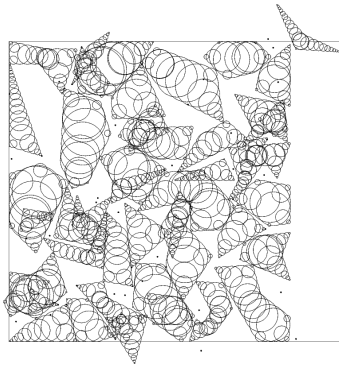
(a) Initial Solution PH1.



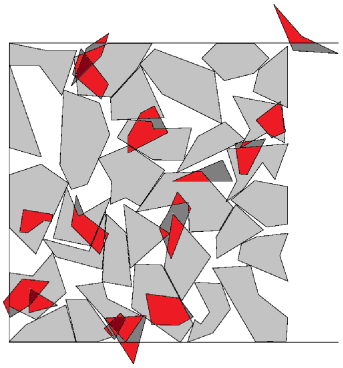
(b) PH1 Compaction with big pieces.



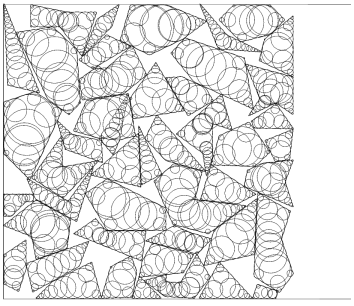
(c) PH1 Layout with only the pieces visible.



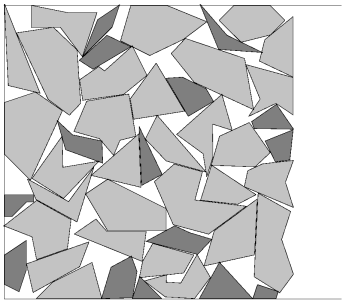
(d) Initial Solution PH2.



(e) PH2 Initial Solution with only the pieces visible.



(f) PH2 Compaction with all pieces.



(g) PH2 layout with only the pieces visible.

Figure 5.9: Two-Phase compaction procedure.

away point from the outline of the hole. The development of this method is expected to improve the final results, by allowing a successful verification if a piece fits in a certain hole, and if it does, determine the best piece placement position and orientation to place the piece inside the hole. The selected placement rule managed to place the pieces near their defined position, but most of them had high overlaps with the surrounding pieces. Due to the lack of exhaustive testing, it was not possible to determine if the used rule (small pieces to small available holes) is the best rule, or if the others would allow better results, and how much improvement would they provide, if they are better.

The piece to hole assignment is the first part of the second phase, with the remaining part dealing with the compaction of the layout in order to remove overlaps while minimizing the total strip length. To successfully produce a feasible layout with a reduced length, the parameters of the objective function must contain a proper configuration. Assigning different weights to the component produces a significant impact on the final layout, even with the pieces pre-compacted and with their relative positions defined. For this reason, the experiments that were done are insufficient to determine their effect in the final quality of the layout.

Since the main objective is compacting the layout to achieve the lowest length possible, the natural choice was selecting only that component, while the overlap correction was done through the model constraints by having the other components of the objective function disabled.

The use of *PCC* has shown to produce many infeasible layouts, even with Post-Optimization. This can be explained, in part, by the difficulties in solving the overlaps on the orthogonal axis limited by the width of the strip. When the pieces are packed too tightly, there is no available space for the pieces to spread-out, preventing the correction of the overlaps. By using the *CCC*, this effect was minimized, but at the cost of a reduced compaction quality of the layout, due to the excess covering by the circles.

Comparing the results produced by the Two-Phase (with Post-Optimization) approach to the results presented in the previous chapter allows determining that this approach produces the best results from all previously presented approaches. The best results produced with the Two-Phase approach are close to the best results presented in the literature. Due to the geometrical representation used, it is not possible to ensure that a layout is completely feasible without having any excessive covering (such as *CCC*), which degrades the quality of the final solution. For this reason, it is not entirely fair to compare the results derived from this approach to the best results in literature, since the best results in literature have been achieved with the use of polygonal covering of the pieces, without any separating distance imposed. Considering that our results are very close to the best in literature, while not surpassing their quality, this validates the Two-Phase approach as a possible approach to solve the Nesting problem with continuous rotations.

5.3 Tweaking Normal Optimization for Layout Feasibility in Post-Optimization

The infeasibilities that occur in the layouts are only partially addressed with the Post-Optimization approach presented in section 4.4.4. Some of the reasons for this partial success are derived from the numerical precision problems, lack of space across the width of the layout to move the pieces, and interlocking pieces. The numerical precision problems increase with higher resolutions, due to the existence of very small size circles, which reduces the ability to remove overlaps between the pieces. The second case occurs when already very tight layouts need to be expanded to remove overlap situations. Expansions across the layout length are always possible by increasing its length, however, the same is not true for expansions across the width due to the fixed width of the container. In the third case, the overlaps between pieces are also very difficult to solve due to the concavities of a piece preventing movement of other pieces.

Due to the great difficulty in solving the numerical precision problems, this problem is addressed by avoiding resolutions with excessive high quality (i.e., that contain circles with very small size).

For these and other reasons a solution was created that consists in reducing the width of the container by a small amount during the normal compaction, and later using a Post-Optimization phase with the full width. This approach is designed to be able to improve the number of feasible solutions, without a significant impact on the quality of the solution, since the pieces are placed near their final relative positions, and only require minor adjustments.

Using this approach with *CCC* was also able to produce a higher number of feasible layouts, due to the additional room required to remove overlaps created by the hole assignment and placement method.

5.3.1 Layout Width Reduction

The use of *CCC* produces more feasible layouts than the *PCC* and *ICC* covering. The difficulty of the *PCC* and *ICC* to generate feasible layouts can derive from the inability of the pieces to spread-out across the width due to the fixed width of the strip. This problem is aggravated when the circle coverings of the pieces are already in overlapping configurations, which produces an increased overlap in the polygonal representation of the pieces (compared to when using *PCC* or *ICC* with their circle coverings without overlaps).

In order to compensate for this, and possibly achieve a feasible solution, a strip with a reduced width is used in the normal compaction phase, together with *PCC*, while using all of the strip width in the Post-Optimization phase. The use of *ICC* has downsides, due to generating layouts with too much overlap, which requires a bigger reduction in the container width in order to correct them. A significant reduction in the container leads to a significant change in the relative position of the pieces when they are processed in the Post-Optimization phase, which discards the result of the compaction in the normal phase. The objective is to be able to adjust the reduction of

the container width while achieving a feasible solution without significant changes in the relative positions between the pieces.

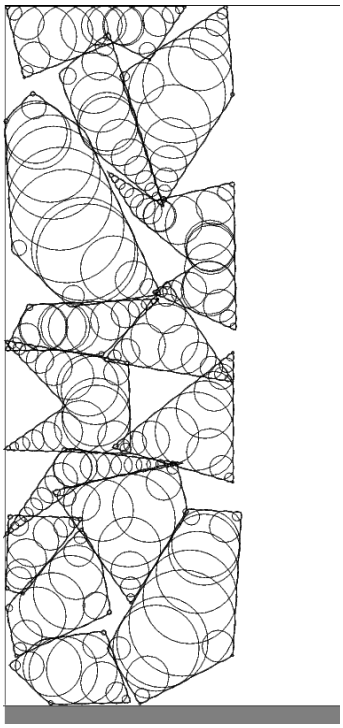
An example of this method can be seen in Fig. 5.10. The normal compaction process starts with the layout having its width reduced by a given amount, and then proceeds by compacting the pieces (which can be seen in Fig. 5.10a, using *HR-PCC*, and in Fig. 5.10b, with its polygonal representation).

The Post-Optimization phase uses the full width of the container, which gives sufficient space across the width for the pieces to adjust, and remove overlaps (this can be seen in Fig. 5.10c, using *HRP-CCC*, and in Fig. 5.10d, with its polygonal representation). This example uses pieces from instance *poly1a* with width = 40. The normal compaction achieves a length of 13.11, with a reduced width of 39 (a reduction of 2.5%). The Post-Optimization uses the full width of 40 and achieves a length of 13.91. This approach is more appropriate to be used with *PCC* and *ICC* coverings in normal approaches, since *CCC* already achieves feasible solutions. If other approaches are found to usually generate infeasible solutions, even with *CCC*, this approach may be able to correct some of those solutions.

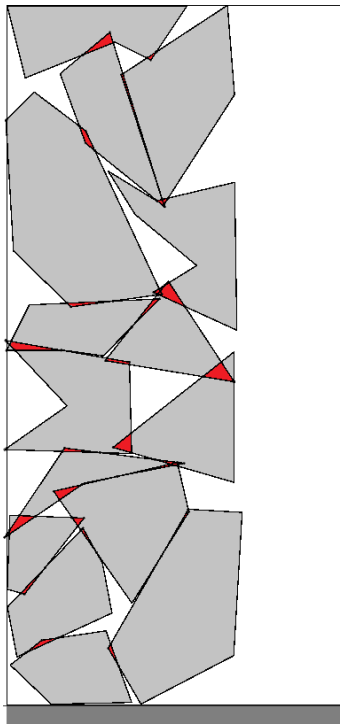
5.3.2 Results and Discussion

These computational experiments aim to verify the impact that the reduction of the layout width has on the number of feasible solutions, and the quality of the final solution. In order to do this, the layouts were compacted using *PCC* with *HR* in the normal optimization phase, and *CCC* with *HRP* in the Post-Optimization, while the width of the container varied between 95.000% and 99.375%. In order to achieve a low computational cost during the normal compaction phase the model variant *M1E* was used, and to achieve a high quality solution the *M2E* model variant was used in the Post-Optimization. The difference between using *PCC* and *CCC* was analyzed through the quality of the final solution, in the computational experiments considering the same configuration with the layout width at 95.000%. The compaction using the full width in both normal and Post-Optimization is also shown to allow comparing the results from this method to the results obtained without it.

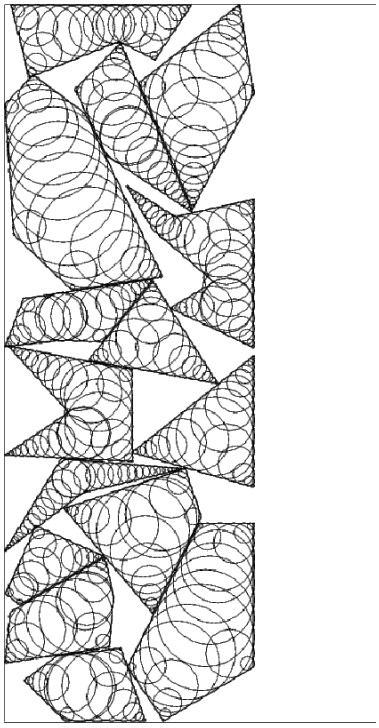
The layouts produced in normal compaction phase were created using the Two-Phase approach specifically to generate tight layouts with pieces that are overlapping, and also with the desired reduction in layout width. The Two-Phase approach allows generating layouts with overlaps in the circle coverings that could not be achieved using a normal compaction process. The normal compaction process starts with a feasible layout, where the initial positions of pieces have a non-overlapping configuration, and the compaction process adjusts the pieces to each other (using continuous translations and rotations), trying to prevent overlaps in their circle covering, while reducing the length of the container. For this reason, a normal compaction process produces almost no overlaps considering the circle covering representation of the pieces, while possibly generating overlaps of their polygonal representation (which is usual when using *PCC* and *ICC* due to their reduced covering). In order to ensure that pieces would adjust well but still produce overlaps in these layouts, the compaction process in the Two-Phase approach allowed an increase



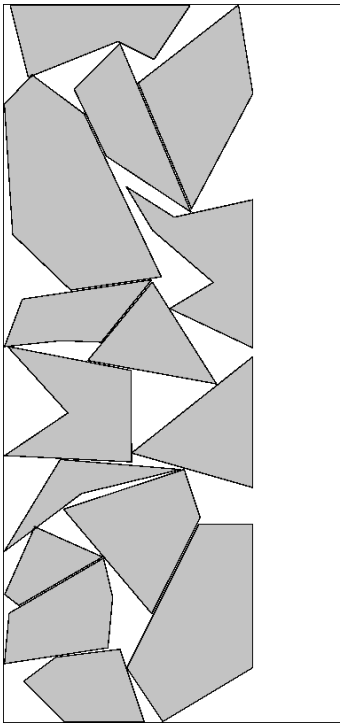
(a) PCC of normal compaction with width reduction .



(b) Polygonal representation of normal compaction with width reduction.



(c) PCC of Post-Optimization compaction with full width.



(d) Polygonal representation of Post-Optimization with full width.

Figure 5.10: Width reduction strategy to improve Post-Optimization feasibility (using *poly1a* with *CCC*).

Table 5.6: Different reductions of the layout size (Width = 40) using Two-Phase approach with instance *poly1a*, and 30 runs each configuration.

Width						Obj. Function (F)			Avg.	#
(%)	(w)	Phase	Res.	Cov.	Model	Min.[c]	Avg.	Max.	Time (s)	F.L.
100.000	40.00	2PH[a]	*HR*	*PCC*	*M1E*	12.95	14.22	15.58	14.77	0
100.000	40.00	P-OPT[b]	*HRP*	*CCC*	*M2E*	14.12	14.82	16.18	6.17	15
99.375	39.75	2PH[a]	*HR*	*PCC*	*M1E*	12.38	13.86	16.02	18.13	0
100.000	40.00	P-OPT[b]	*HRP*	*CCC*	*M2E*	13.71	14.72	16.13	12.17	13
98.750	39.50	2PH [a]	*HR*	*PCC*	*M1E*	12.62	14.33	17.45	13.47	0
100.000	40.00	P-OPT[b]	*HRP*	*CCC*	*M2E*	13.56	14.86	17.50	6.43	18
97.500	39.00	2PH [a]	*HR*	*PCC*	*M1E*	12.72	14.40	15.87	17.00	0
100.000	40.00	P-OPT[b]	*HRP*	*CCC*	*M2E*	13.54	14.76	15.53	9.67	18
95.000	38.00	2PH [a]	*HR*	*PCC*	*M1E*	13.48	14.73	16.12	18.63	0
100.000	40.00	P-OPT[b]	*HRP*	*CCC*	*M2E*	13.43	14.57	15.60	6.60	17
95.000	38.00	2PH[b]	*HR*	*CCC*	*M1E*	13.91	15.50	17.35	15.60	16
100.000	40.00	P-OPT[b]	*HRP*	*CCC*	*M2E*	13.93	14.76	15.92	2.23	22

[a] Values considering all solutions
[b] Values considering only feasible solutions
[c] Reference value 13.21: best result in literature, considering length

in the length of the layout obtained in the first phase by up to 120% for the compaction in the second phase. The results of this approach can be seen in Table 5.6.

Table 5.6 shows that the reduction of the strip width in the Two-Phase approach (from 99.375% to 95.000%) leads to an increase in the final number of feasible solutions. As we reserve more space (by reducing width) to be used in the Post-Optimization phase (by increasing the width reduction parameter), the number of feasible solutions produced also increases, but up to a certain point. Even with more space across the width of the container, the infeasibilities of the layouts may not be completely solved. This can occur due to numerical precision problems, created by using resolutions that contain very small circles, and when certain pieces have overlaps where the bones of their medial axis intersect. This causes circles, on each side of the intersection between both bones, to prevent movements that would solve the overlap of the piece. Any of these relative movements between both pieces would increase the overlap between circles, in order to find a direction to move the pieces so that the overlap is corrected. Since the solver searches for the path that minimizes the overlap, it then becomes stuck in this local minimum regarding the overlap of this pair of pieces. Another reason for the inability to solve infeasibilities is due to interlocking pieces, where the concavities of a piece prevent others or another from moving in directions that correct the overlaps.

Using *CCC* (instead of *PCC*) with this approach and 95% of the width in the normal optimization process limited the number of feasible solutions to 16 out of 30, which is a much higher number of feasible solutions. When comparing the differences while using the Post-Optimization phase, this approach with *CCC* managed to correct 6 infeasible layouts, with a total of 22 feasible

layouts. This total number of feasible layouts produced with *CCC* is clearly superior to the number of layouts achieved when using *PCC*, but it also returns worse layout compaction results. The difference in quality of the compaction is mainly due to the excessive covering of the *CCC*.

The computational experiments indicate that using this approach to correct infeasible solutions, when addressing the layouts generated by the Two-Phase approach, is equivalent to using *CCC* in the normal optimization process with full width, or using *PCC* with a width reduced by 1.25%. This small reduction in the container width allows achieving nearly the same relative positions between the pieces, while still maintaining a small distance available to address infeasibilities in the Post-Optimization phase.

A combination of a type of covering and a specific reserved percentage of the width of the layout may prove useful to correct infeasible layouts, but more extensive tests are required. This approach shows promising results, but it is still in an early stage of its development. The main difficulty consists in determining the best parameter configuration that enables producing good results for any given instance.

5.4 Multi-Layer Approach

The Multi-Layer approach focuses on dealing with problems that cannot be solved using the normal approaches due to their computational cost. This situation arises in problems with a large number of pieces, and also when the geometrical representation of the pieces require too much detail, due to their complexity. This approach follows the idea that in order to successfully compact a layout it is not required that all parts of the layout are improved at the same time. For this reason, dealing with partial regions of the layout, and improving them locally, allows tackling the compaction problem through a set of smaller sub-problems. The pieces are divided into distinct groups, and each group is packed into the current layout, without overlapping pieces from other groups. This packing strategy resembles a placement of pieces in layers. Since the groups are tackled one by one and sequentially, they can be considered almost independent of each other. The compaction strategy adjusts the current pieces to the container, and also to the other previously placed pieces.

This approach classifies the groups by two characteristics, the groups that have pieces able to move (mobile), and groups that have fixed pieces. The group, or groups, that have mobile pieces, allow free movement to their pieces. The other groups, with fixed pieces, do not allow their translations or rotations, but their representation still produces an influence regarding the valid areas for piece placement.

The main benefit of this approach is the reduction in computational time required to pack the total amount of pieces, when compared to compacting all pieces at once. Since the computational cost grows almost exponentially with the number of pieces to place, reducing just a few of pieces can provide significant gains in computational time, which is the main limitation when addressing real world problems (very large and complex instances).

This approach offers a trade-off between the compaction quality and computational time. When the number of pieces being compacted is reduced, for each layer, the quality of the compactions is expected to be reduced, while the computational cost also is reduced.

This section discusses the Multi-Layer approach, considering its geometrical characteristics, together with advantages and disadvantages, and its computational experiments.

5.4.1 Layered Piece Placement Strategy

The Multi-Layer approach basically divides the complete set of pieces into several separate groups. The pieces are attributed randomly to each set, or using a greedy rule. The random distribution of pieces for the different sets can be considered the default placement strategy. The process starts with an empty layout, where the pieces of one set are placed, configured into initial starting positions organized by a grid structure, without overlaps. The set is then compacted, and the pieces from the next set are placed, with their positions also defined by a grid.

Depending on the configuration, one or more adjacent layers may have their pieces defined as mobile pieces, simultaneously, while the remaining placed layers have all their pieces defined as fixed pieces. These layers can be denoted as mobile layers, and fixed layers, respectively.

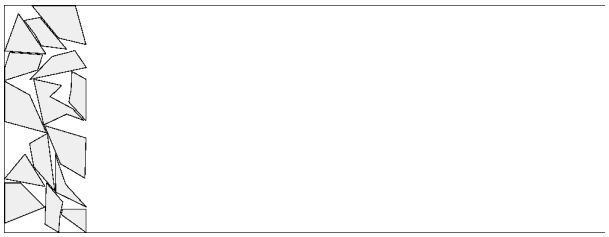
Considering only one mobile layer, it is defined always as the current set being compacted into the layout, while all previously placed sets of pieces are fixed into their compacted position. This enables pieces from the current set to adjust to the pieces in the previous layer. This process is repeated until all sets of pieces are placed into the layout.

The process of compacting one set of pieces independently of the others, by placing the pieces layer-by-layer, is defined as a compaction using one mobile layer (where only the pieces from only one layer are allowed to move freely). An example of this process can be seen in Fig. 5.11, where a single layer is compacted, taking into account the other layers that were previously placed.

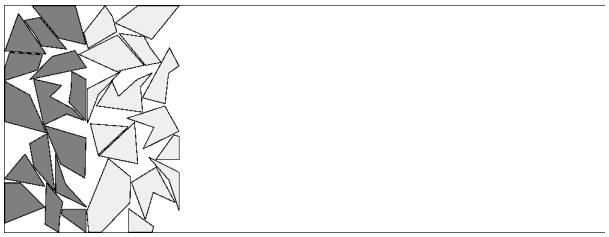
The use of two mobile layers simultaneously has important benefits. The first mobile layer is the current set of pieces being compacted, while the second mobile layer is the previously placed set. Having this configuration allows adjusting the pieces of the previous layer to the pieces of the current layer, thus achieving a better fit between pieces. The main downside is that the number of pieces that are being worked upon doubles, increasing significantly the computational cost. Using the two-layer configuration, instead of just one, allows adjusting the trade-off between the final quality of the layout and the required computational cost. This potential problem is naturally controlled, in part, due to the pieces in the previously placed layer being already packed together, thus not requiring significant computational cost to move, only enough to adjust themselves.

The two-layer configuration allows achieving a smoother compaction since the outlines of each layer are merged, due to the pieces being well adjusted to their neighbors, thus reducing wasted space. An example of a two-layer compaction can be seen in Fig. 5.12, where two sets of pieces are compacted simultaneously, taking into consideration the remaining previously placed pieces. All pieces in the layout that do not belong to the current set are set as fixed.

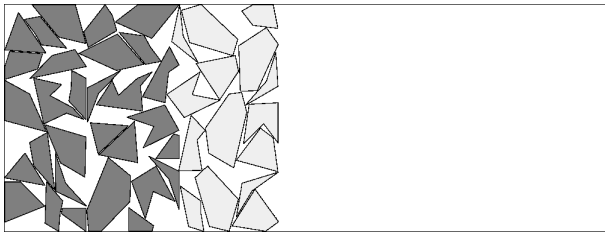
Many parameters influence the behavior of this approach, being difficult to estimate their impact and determining the best configuration for a given problem, particularly since they influence



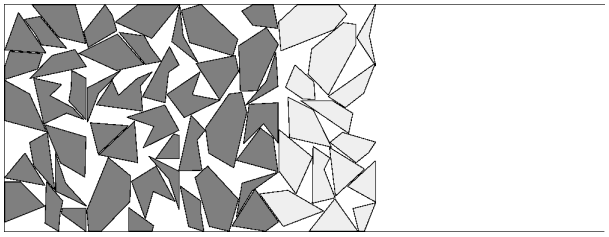
(a) Layer 1 with mobile pieces.



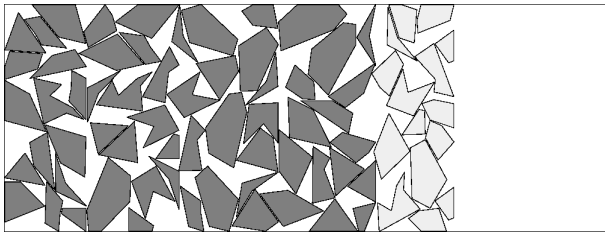
(b) Layer 1 with fixed pieces and layer 2 with mobile pieces.



(c) Layer 1 and 2 with fixed pieces and layer 3 with mobile pieces.

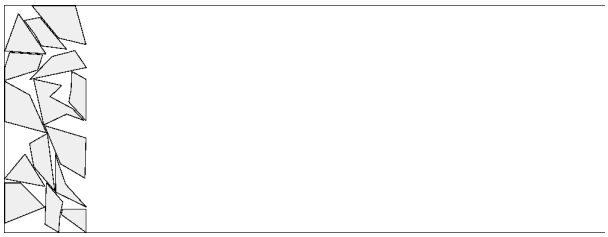


(d) Layer 1 to 3 with fixed pieces and layer 4 with mobile pieces.

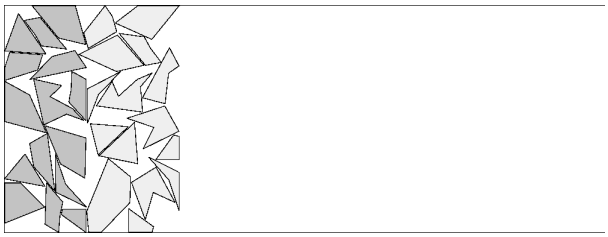


(e) Layer 1 to 4 with fixed pieces and layer 5 with mobile pieces.

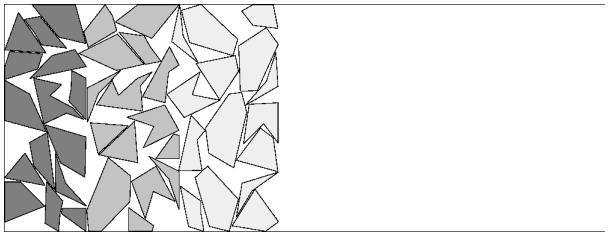
Figure 5.11: Single-layer compaction.



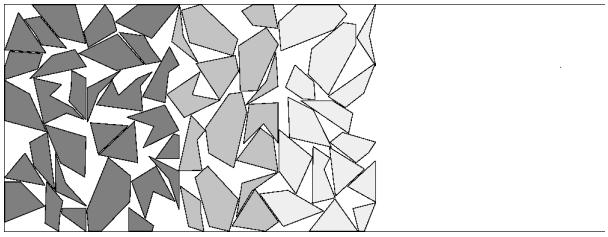
(a) Layer 1 with mobile pieces.



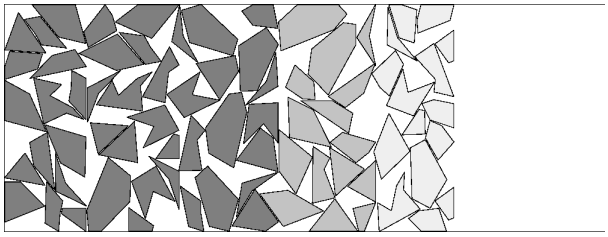
(b) Layer 1 and 2 with mobile pieces.



(c) Layer 1 with fixed pieces and layer 2 and 3 with mobile pieces.



(d) Layer 1 and 2 with fixed pieces and layer 3 and 4 with mobile pieces.



(e) Layer 1 to 3 with fixed pieces and layer 4 and 5 with mobile pieces.

Figure 5.12: Two-layer compaction.

each other. Among those parameters (resolution, coverings, model variants, etc), we can also define the number of pieces that is packed in each layer (which also determines the total number of layers) and impacts the computational cost, the number of mobile layers (which influences the quality of the solution by allowing a better adjustment between pieces from different layers), the compaction strategy used to compact a given mobile layer or set of layers (using normal compaction, Two-Phase approach, multi-resolution, among others), selection of pieces for each set (random, big and small, etc), among others. Selecting the best configuration is a difficult task that cannot be addressed without extensive testing.

Other difficulties may arise due to using multiple layers with large differences between the size of the pieces. Pieces that have a large size relative to the size of the layers being compacted may cause difficulties if one of those pieces has its position fixed while being large enough to occupy a region that belongs to multiple layers. A large piece could prevent some regions from being filled by pieces, degrading the final solution quality. A possible method to address this difficulty requires having a method to detect holes and assign unplaced pieces to the holes generated before.

Besides the size of the pieces, another important aspect that creates difficulties is the complexity of the outline of the pieces. Very complex pieces might require additional effort to adjust to each other, requiring some kind of pre-processing to match compatible pieces.

5.4.2 Results and Discussion

These computational experiments aim to analyze the impact of the multi-layer approach in the compaction of medium, and large size instances. These experiments will also allow verifying the trade-off between the number of mobile pieces selected and the growth in computational cost compared to the compaction using all pieces, and also verifying the trade-off between the number of mobile layers being used and the final layout quality. The instances used will be based on *poly3a*, *poly4a* and *poly5a*, and multiples of instances *poly5a* with 150 and 300 pieces in total.

Table 5.7 presents the compaction values for the instances *poly3a*, with 45 pieces, *poly4a* with 60 pieces and *poly5a* with 75 pieces. The pieces used *HR CCC* coverings and were solved with the model *M1E*. This table allows comparing the Multi-Layer approach when using one and two mobile layers, with the normal compaction with all pieces. The layers are composed by sets of 15 pieces, which are compacted with 3 layers for instance *poly3a*, 4 layers for instance *poly4a* and 5 layers for instance *poly5a*. It can be seen that using the Multi-Layer approach has a reduced computational cost when compared to the compaction done considering all pieces at the same time. The downside of this approach is the impact on the quality of compaction caused by the bad adjustment of the pieces between layers. The use of two mobile layers enables addressing this problem, where both minimum and average compaction values are significantly lower when using two mobile layers. The use of one mobile layer is justified for cases that cannot be solved with an acceptable time interval with two mobile layers, since it has a much lower computational cost. Another aspect that can be noticed in Table 5.7 is that the computational cost of both configurations (one or two mobile layers) does not seem to grow exponentially. This is explained by the fixed number of pieces in each layer, where the only component that grows at each

Table 5.7: Compaction with one and two mobile layers, with 15 pieces for each layer (using *HR-CCC* and model variant *M1E*, 10 runs each).

Instance[b]	# Mobile Layers	Obj. Function: min l			Time[a](s)
		Min.	Avg.	Max.	
poly3a	—	42.14	44.21	46.75	296.10
(38.81)	1	45.06	48.20	51.09	30.60
	2	42.46	44.85	46.47	108.60
poly4a	—	55.69	57.80	59.88	781.50
(52.08)	1	61.02	64.14	67.74	55.05
	2	57.80	59.60	61.92	165.75
poly5a	—	69.03	71.04	73.17	2185.40
(63.81)	1	76.37	80.16	85.32	80.90
	2	71.69	73.38	75.15	275.05

[a] Tests using Core i7-2670QM@2.2Ghz, 8Gb Ram 1.33Ghz
[b] Reference value: best result in literature, considering length

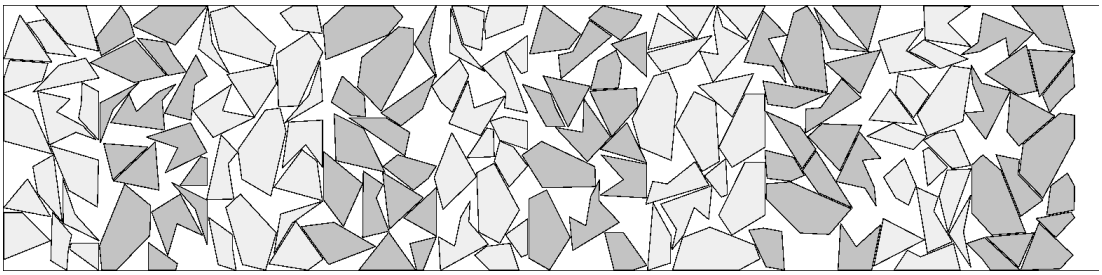


Figure 5.13: Layout produced with one mobile layer, for instance *poly10a*.

step is the number of objects that must not be overlapped (all the previously placed pieces). This limited growth in computational cost allows addressing much larger problems than the previously presented approaches.

An example of a large layout compacted with this multi-layer approach can be seen in Fig. 5.13. This example is a variant of the instance *poly1a*, denoted as *poly10a* due to consist in the repetition of the *poly1a* pieces for about 10 times. This large instance has a total of 150 pieces and is compacted using the Multi-Layer approach using a single mobile layer, with *HR CCC* and model variant *M1E*. The layout, with this configuration, as seen in Fig. 5.13, has a total length of 162.35 achieved within a computational cost of 120 seconds. The different layers are marked with different shades of gray color.

In order to verify if this approach is able to still address larger problems, another example was used, using the same configuration but with more pieces (300 in total), which was also based on a repetition of instance *poly1a*, was denoted as *poly20a*. An example of a layout compaction of this instance can be seen in Fig. 5.14, where it achieved a length of 321.21, in 519 seconds. Preliminary experiments can be seen in Table 5.8 where a a small number of configurations were experimented, with 4 runs each, using model *M1E* with *HR-CCC*.

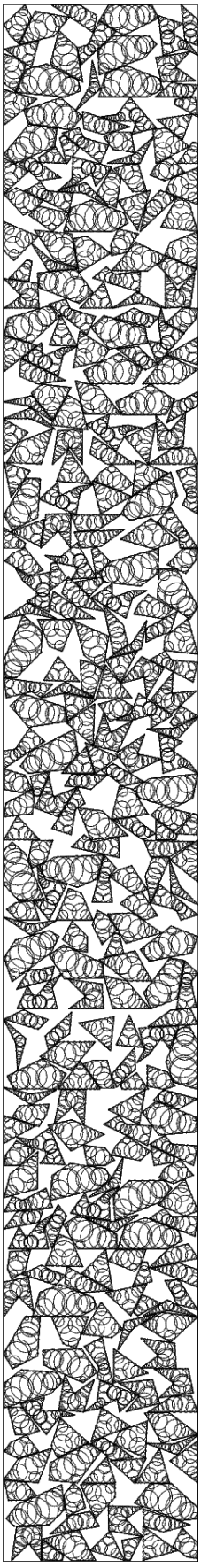


Figure 5.14: Layout produced with one mobile layer, for instance *poly20a*.

Table 5.8: Compaction of *poly20a* with one mobile layer, 4 runs each.

# Pieces per Layer	# Layers	Obj. Function: min l			Avg. Time (s)
		Min.	Avg.	Max.	
8	38	375.30	381.87	385.58	193.25
15	20	319.73	327.05	337.57	500.36
20	15	312.08	316.67	319.55	692.20
30	10	305.70	306.37	307.15	1422.67

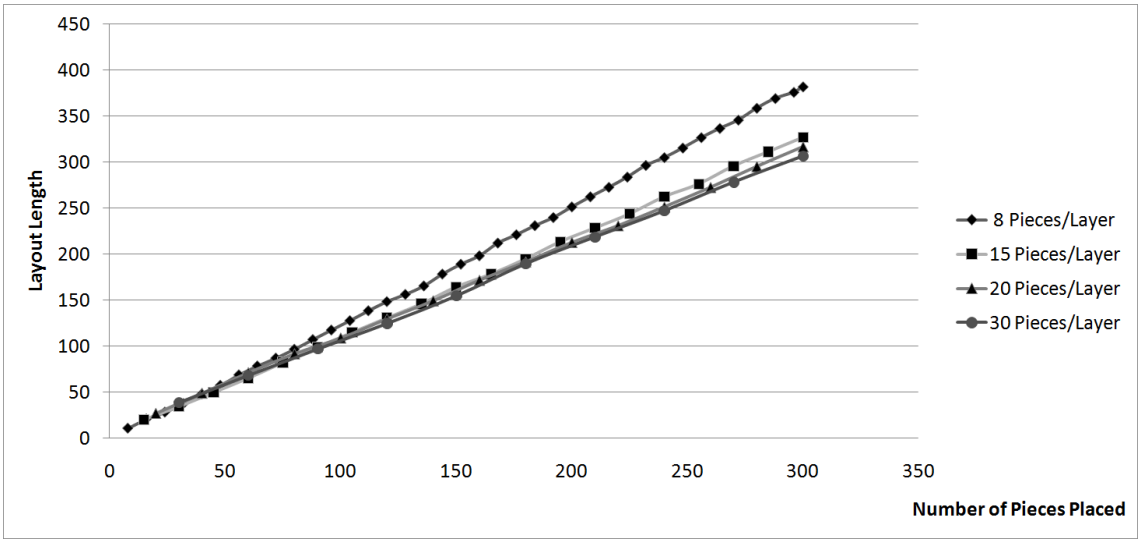


Figure 5.15: Results of compaction (average length) for instance *poly20a*, considering different number of pieces per layer.

The results in Table 5.8 show the impact that the number of pieces assigned to each set (which also determines the total number of layers) has on the computational cost, and in the quality of the solution. With the reduction in the number of pieces per layer, the computational cost is reduced, but the quality of the solutions is degraded. There is a significant impact when addressing this trade-off. In Table 5.8, by reducing the number of layers from 38 to 10, the length of the layout was reduced by about 25%, while the computational cost increased by more than 7 times. The impact on the quality of the layout is mainly due to the NLP model, which is better at adjusting pieces when all the pieces are available to be packed instead of packing in groups, one by one. For this reason, layouts that have less layers usually return better quality solutions, when compared to layouts with more layers. This can be seen in Fig. 5.15, where the same instance, when compacted with more pieces per layer (thus less layers) leads to higher quality solutions (with lower length). The added length at each layer compaction grows linearly, since the same number of pieces is being added to the layout, but due to the difficulties in adjusting pieces between layers, the final length is bigger for the configurations that use many layers. Fig. 5.16 shows the growth in computational cost, taking into account the number of pieces per layer. Instances with fewer pieces per layer have a very low computational cost, while the reverse leads to high computational cost.

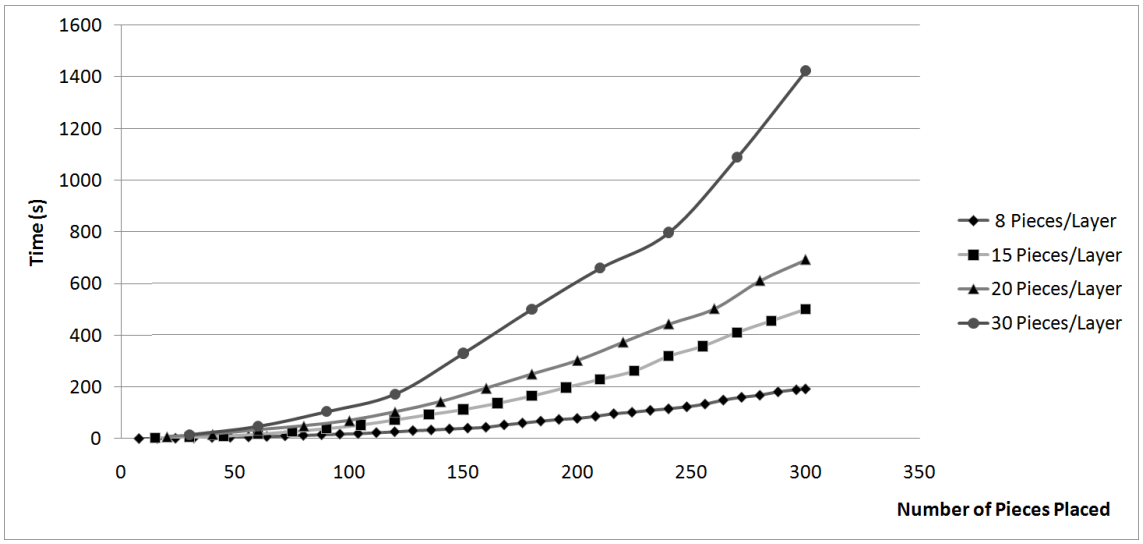


Figure 5.16: Results of compaction (average time) for instance *poly20a*, considering different number of pieces per layer.

The results presented in Table 5.8 could be improved if a Two-Layer approach as used. Unfortunately, the growth in the computational cost could make the problem difficult to solve in reasonable time. If this is addressed by reducing the number of pieces assigned to each layer, the benefit of using multiple layers would be reduced. Finding the best configuration for a given instance allows extracting the best quality possible while considering a reasonable computational cost.

The instance *swim4* was also compacted, being the best solution produced seen in Fig. 5.17 with a length of 31362.23 in 2454 seconds. All these computational experiments were done using *CCC*, with *HR* and *M1E* model variant.

The Multi-Layer approach enables tackling problems with large number of pieces, with a controlled computational time. Using one layer to compact the pieces proved to be the fastest configuration, but returning the worst results. Using two mobile layers showed an increase in the quality of the compacted layout, but with an increase of the computational time. The biggest challenge is find a way to increase the compaction quality in order to reach the level of other approaches. This approach is not only useful to tackle problems that are very computationally expensive to solve in a reasonable time, due to their large size, but also to address some technological limitations of the software used to solve it. Very large instances may produce a very large number of variables, constraints, which can exceed the limits for which the software was designed.

5.5 Extended Non-Linear Approaches Evaluation

In this section the best results for each instance, produced by all approaches in both previous and current chapter, will be compared to the best results in literature. Their configuration, which includes resolution, type of covering and model variant, is also presented in order to detect patterns

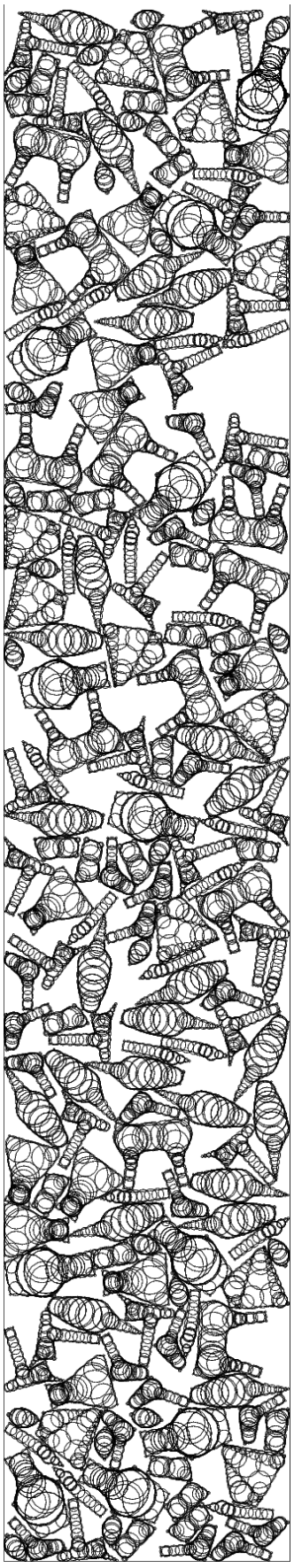


Figure 5.17: Layout produced with one mobile layer, for instance *swim4*.

Table 5.9: Best results produced by the extended NLP approaches.

	NLP-CC Configuration			*NLP-CC Results*	
Instance	Appr.	Normal Optimization	Post-Optimization	Min. O.F.	Time (s)
jakobs1	*NLP*	*HR-CCC-M2E*	—	13.85	75
poly1a	*2PH*	*HR-PCC-M1E + WID95%*	*HRP-CCC-M2E*	13.43	10
poly2a	*2PH*	*HR-PCC-M1E*	*VHR-CCC-M1E*	26.00	116
poly2b	*2PH*	*HR-CCC-M1E*	*HRP-CCC-M2S*	29.35	179
poly3a	*2PH*	*HR-PCC-M1E*	*HRP-CCC-M1E*	39.03	209
poly3b	*2PH*	*HR-PCC-M1E*	*VHR-CCC-M1E*	38.94	278
poly4a[a]	*ML*	*HR-CCC-M1E + 4L-1ML*	—	55.69	981
poly5a[a]	*ML*	*HR-CCC-M1E + 5L-1ML*	—	69.03	1953
poly10a	*ML*	*HR-CCC-M1E + 10L-1ML*	—	157.30	154
poly20a	*ML*	*HR-CCC-M1E + 10L-1ML*	—	305.70	1377
swim4	*ML*	*HR-CCC-M1E + 8L-1ML*	—	31 362.23	2454

[a] Tests using Core i7-2670QM@2.2Ghz, 8Gb Ram 1.33Ghz

in configuration parameters that lead to best solutions. The best results for the approaches presented in this chapter, are presented in Table 5.9, with their respective approach, and configuration parameters used during the normal optimization phase, and the Post-Optimization phase. The values refer to the minimum length achieved when compacting the layout, with the used computational time for that specific run. In order to facilitate the comparison with the literature, the set of all developed approaches will be denoted as *NLP-CC* (Non-Linear Programming with Circle Covering).

The instances presented in Table 5.9 consist not only of the previously introduced instances, but now also consider the results for instances *poly10a*, *poly20a* and *swim4*, which were created in order to test the performance of specific approaches when dealing with very large size instances. In this table, the configuration parameters are presented for both the normal optimization and Post-Optimization phases since they have a significant impact on the final quality of the solution. The instances that do not have Post-Optimization had already produced feasible solutions during the normal optimization phase, although some of the instances that show a Post-Optimization phase might also have been feasible, but used Post-Optimization to improve the result. As presented in the results, each approach was used to address specific types of instances, for which they have been developed. The instance *jakobs1* achieved the best result during the computational experiments with *Multi-Resolution* approach, although using only the normal compaction. The instances *poly1* to *poly3* were addressed by the *Two-Phase* approach (*2PH*) which focuses in solving small to medium size instances, aiming at achieving the best solution quality possible. The best results used a combination of *HR-PCC* with model variant *M1E* while requiring a Post-Optimization phase to correct infeasible solutions and improve the already feasible ones, as seen with *poly2b* (which was achieved by *CCC*). The configuration parameter that is identified as *WID95%* is related to the tweaking of the normal optimization phase, where the container width is reduced to 95 % and using 100 % during the Post-Optimization phase, to reduce difficulty in achieving feasible solutions.

The large size instances used the *Multi-Layer* approach (*ML*) using different configurations for the number of pieces used for each layer, and using only one layer with fixed pieces at each iteration. The additional configuration parameters that are used in the *Multi-Layer* approaches are related to the number of layers used. In Table 5.9, the instance *poly4a* uses parameter *4L-1ML* which determines that 4 sets of pieces are used, compacted into 4 layers, and also that only 1 set of pieces has free movement at each iteration (only one mobile layer). The other instances use the same type of configuration, 5 layers for *poly5a*, 10 layers for *poly10a* and *poly20a*, and 8 layers for *swim4*. The computational cost of instances *poly4a* and *poly5a* are significantly higher than the others due to being executed in a computer with different specifications. All these large instances were solved in less than an hour. In order to compare the best results for each instance, obtained by the approaches in both chapter 4 and chapter 5, to the best results in the literature, Table 5.10 presents an extension of Table 5.9, with the best results and the configuration that produced them.

The results presented in the left side of Table 5.10 show that the best results have been achieved by the extended approaches introduced in this chapter, except for the instances *jakobs1* and *swim*, which had their best solutions generated by the approaches introduced in chapter 4. By observing the results one can see that the configuration parameters have produced better solutions with a specific configuration, using model variant *M1E* with *HR* covering during the normal optimization phase. On average, the model variants that produce the best quality solutions are based on model *M2*. The best results for large and very large instances only used the configuration *HR-CCC* since higher resolutions increased significantly the computational cost, while different covering types required a Post-Optimization phase, which would be unable to address such large instances. All these results are compared to the best results in literature in the right side of Table 5.10.

The results shown in Table 5.10 are very close to the best results in literature, and even though they do not improve in terms of optimization result, there is a potential for better and potentially faster handling of large instances. This is due to the developed approaches using a Circle Covering that produces excess covering of the pieces, which prevents them from getting even closer, where the best results in literature consider that no separation distance between pieces exist. Without this separating distance, the results of the *NLP-CC* approaches could probably match and even surpass the quality of the current best in literature. The developed approaches are able to produce good quality solutions within an acceptable computational time, and also address instances with very large size. Some of these very large instances did not have results considering continuous rotations, such as the instance *swim*. The other very large instances (*swim4*, *poly10a*, and *poly20a*) are included to allow other authors to compare them with their approaches.

5.6 Concluding Remarks

This chapter presented several extensions to the solution approaches presented in the previous chapter, where each focused on addressing a particular limitation (such as computational cost, solution quality and size of the instance). The Multi-Resolution approach tackled the computational efficiency of the normal compaction, by separating the compaction process into two distinct steps:

Table 5.10: Configuration for the best results of *NLP-CC* based approaches, compared to the best in literature.

	NLP-CC Configuration			*NLP-CC Results*		(Stoyan et al., 2012)[a]	
Instance	Approach	Normal Optimization.	Post-Optimization	Min. O.F.	Time (s)	Min. O.F.	Time (s)
jakobs1	*NLP*	*HR-ICC-M2E*	*HRP-CCC-M2E*	13.08	173	11.82	495
poly1a	*2PH*	*HR-PCC-M1E + WID95%*	*HRP-CCC-M2E*	13.43	10	13.21	150
poly2a	*2PH*	*HR-PCC-M1E*	*VHR-CCC-M1E*	26.00	116	25.71	280
poly2b	*2PH*	*HR-CCC-M1E*	*HRP-CCC-M2S*	29.35	179	29.07	958
poly3a	*2PH*	*HR-PCC-M1E*	*HRP-CCC-M1E*	39.03	209	38.81	390
poly3b	*2PH*	*HR-PCC-M1E*	*VHR-CCC-M1E*	38.94	278	38.79	1156
poly4a[b]	*ML*	*HR-CCC-M1E + 4L-1ML*	—	55.69	981	52.08	410
poly5a[b]	*ML*	*HR-CCC-M1E + 5L-1ML*	—	69.03	1953	63.81	867
poly10a	*ML*	*HR-CCC-M1E + 10L-1ML*	—	157.30	154	—	—
poly20a	*ML*	*HR-CCC-M1E + 10L-1ML*	—	305.70	1377	—	—
swim	*NLP*	*HR-CCC-M2S*	—	7023.00	96 817	—	—
swim4	*ML*	*HR-CCC-M1E + 8L-1ML*	—	31 362.23	2454	—	—

[a] Results obtained using ϕ-functions and a mathematical model.

[b] Tests using Core i7-2670QM@2.2Ghz, 8Gb Ram 1.33Ghz

a first step that focused on fast compaction of the pieces' relative positions, while the second step refined the adjustments to the pieces' positions. The Two-Phase approach addressed the quality of the final solution by separating the compaction process into two phases: first phase compacted the larger pieces to obtain the overall structure of the layout, while the second phase placed the smaller pieces and compacted all of them while forcing them to achieve a feasible solution within an enclosed region. The number of infeasible solutions produced in the normal optimization and the Post-Optimization phase were tweaked by reducing the width of the containers in the normal phase, and using their full width in the Post-Optimization approach. The last approach, Multi-Layer, divided the pieces into different groups, and compacted them individually to deal with the growth in computational cost, which grows exponentially with the number of pieces. The results obtained in the computational experiments of these extended approaches allowed to analyze the impact that each had on the limitation that they addressed.

The Multi-Resolution approach was able to improve the quality of *LR* layouts significantly, with a modest increase in the computational cost by re-compacting its layout solution using *HR* covering. Comparatively to the single-step *HR* compaction, this approach managed to, most of the times, achieve similar quality and even surpassing it occasionally, with a much lower computational cost. Part of the occasional improvement in the solution quality with this approach over the single-step *HR* is explained by allowing a second attempt at finding a better local minimum than the current one.

The Two-Phase approach was shown to be able to achieve promising results, by improving the quality of the layouts over the other approaches, within a reasonable computational time. Selecting the appropriate configuration parameters is very important, since they have a strong influence on the quality of the results. The selection of the sets of big and small pieces, and definition of initial position and orientation of the big pieces influences the produced layout in the first phase. The second phase consists of assigning the remaining small pieces to the holes created in the first phase, selecting their best placement position and orientation, and then re-compacting everything, minimizing total length of the layout and solving existing overlaps. The hole assignment strategy has a great impact on the final solution, as the process to determine the best position and orientation of a piece inside a given hole.

The tweaking of the Post-Optimization method allows increasing the number of feasible solution produced by the presented approaches that tackle the Nesting problem with continuous rotations. The selection of the appropriate resolution is very important to be able to produce a feasible layout, but also the best solution quality possible. This approach was able improve the number of feasible solutions without a significant impact on the quality of the solution (for small reductions in width), since the pieces were already placed on their final relative positions, and only required minor adjustments. Using this approach with *CCC* was also able to produce a higher number of feasible layouts, due to the additional room required to remove overlaps created by the hole assignment and placement method.

The Multi-Layered approach enables solving problems of very large dimension, with hundreds of pieces, in a reasonable time. The growth in computational cost increases with the reduction in

the number of pieces used in each layer (which also increases the total number of layers required to be compacted), making the executions of the NLP model much easier. The computational cost has a higher increase when addressing the problem with many pieces per layer, which shows that the fixed pieces are responsible for a small portion of the total computational cost. The results from this approach also show that the best quality of compaction is achieved when compacting with fewer layers, which degrades as the number of layers increases, and the number of pieces per layer is reduced. This indicates that compacting the layout all at once should produce the best result, but with a prohibitive computational cost, while using a single piece per layer would achieve results comparable to placing one piece at the time.

The results produced by these approaches are not able to beat those present in the current literature, considering continuous rotations. However, the comparison is not entirely fair, since the published results have been compacted with geometrical representations that do not have ”enlarged" outlines due to covering, or other effect, since they use polygonal covering of the pieces, without any separating distance imposed. Since the results obtained by these approaches are very near the ones presented in the literature, it is possible that if they are re-compacted using a polygonal representation without excess waste, the results would be nearly equivalent, or even better, occasionally. This small difference validates these approaches, and with further development, they may be able to tackle problems with greater size with lower computational cost, and provide better quality results.

Chapter 6

Conclusion

The main objective of this thesis was to develop a solution approach that allowed to address real world instances of the Nesting problem with continuous rotations. The difficuty in solving real world instances is due to the very large number of pieces (hundreds) with very complex geometry, and due to the characteristics of the specific problem that is being addressed (such as limitations on the range of rotations, separation between pieces due to technological constraints, among others). In order to achieve this, the geometrical characteristics of the problem were addressed considering the specific requirements imposed, such as continuous rotations, and high quality of piece representation, through a Circle Covering representation. Since no existing methods were able to generate high quality circle coverings, taking into account the requirements of the Nesting problem, several approaches were developed, and improved over time.

The solution method used to determine the piece placement positions into the container, must also fully support continuous piece placement and orientations, while maintaining the pieces inside the container, and in a non-overlapping configuration, all while minimizing the length of the container. Since the mathematical description of the geometrical representation is based on non-linear equations, due to comparison between pairs of circles, and also include trigonometric operations, due to free-rotations, the natural choice was to use a Non-Linear Programming model based approach. Most of the development consisted in finding possible paths to improve the combination between Circle Covering Representation and Non-Linear Programming model approach in order to solve large size, real world problems, with high quality results, and reasonable computational cost.

The main objective was achieved with success, although there is still room for further improvements in order to produce better results. By developing a method that allows tackling real world Nesting problems with irregular pieces and continuous orientations, many industrial applications where this problem arises can be solved in a more efficient way.

The following sections present the main contributions of this thesis and possible paths of exploration for future work.

6.1 Main Contributions

The main objective (addressing real world instances of the Nesting problem with continuous rotations) was achieved by using an appropriate geometric representation that fully supported free-rotations (Circle Covering) and a solution approach (Non-Linear Programming Model) that could use it efficiently and effectively. The work focused on the expansion of each one of these components and also in determining the best parameter configuration that could lead to the best quality results.

The iterative algorithm based on the Medial Axis that was developed is able to provide Complete Circle Covering representations superior to those of comparable approaches. This approach is based on a constructive algorithm (*kCC-MA*) that uses the Medial Axis skeleton of a piece as its support, and computes the best placement positions for the circles. By controlling the approximation error, one can produce coverings with a desired level of resolution, and a reduced number of circles. This enables generating multiple types of resolutions, that can be adjusted to better fit the requirements considering the computational cost, and solution quality. Adjusting the approximation error parameters also enable this approach to produce coverings that adapt to different types of problems, such as Complete Circle Covering (*CCC*) for industrial applications that require a given distance between pieces, and Inner Circle Covering (*ICC*) for layouts with perfect fits, although producing a very large number of infeasible solutions. The other type of covering, the Partial Circle Covering (*PCC*), is the covering type that offers the best compromise between large number of feasible layouts and high quality compactions.

In order to address the Nesting problems with continuous rotations, the choice was based on the combination of Circle Covering representation with a Non-Linear Programming (NLP) model. Two formulations for the NLP model were created, based on different characteristics of the geometrical representation. One NLP model focused on the description of pieces by considering them as a set of circles where each has individual characteristics (Circle-Based Model – *M1*), while the other NLP model described the pieces as a single element (Piece-Based Model – *M2*). These two different formulations had difficulties tackling large problems due to the factorial growth in the number of non-overlapping constraints. In order to address these difficulties, several extensions to their formulations were explored, by aggregating similar constraints in a single summation expression. This allowed producing three types of model variants for each model. The model variants were formulated with the aim to produce different behaviors that can be used to tackle a wide variety of problem sized, by assigning the best model variant to the adequate problem to be addressed. The model variants with more constraints aggregated are able to have reduced computational cost, thus being able to tackle large size instances, although their average solution quality is slightly inferior to the model variants with less or none constraints aggregated. Having multiple formulations allows choosing the most adequate considering on the characteristics of the instances and the minimum solution quality required.

In order to tackle infeasible solutions (caused by incomplete coverings such as *ICC* and *PCC*, overlaps between circle coverings of the pieces caused by the approaches, numerical precision

errors due to small circles and aggregation of constraints, among others), a Post-Optimization approach was developed. It is based on the *CCC* representation with a sufficiently high resolution to ensure a very low approximation error and a NLP model variant selected depending on its computational cost. The Post-Optimization approach has been shown to successfully correct many infeasible solutions, but its efficiency can be improved if using an approach that may require the use of a layout with a reduced width during the normal compaction phase of the layout. This approach does not ensure that all infeasible layouts will be successfully corrected.

Due to limitations that originated from the use of a NLP approach (which converges to a local minimum) without having a method that was able to assist searching for better solutions, several approaches were developed that consist on the iterative use of NLP model to achieve a good solution. The three approaches focused on solving different aspects where the performance of the NLP approach was lacking. The first approach tackled the efficiency of the compaction of the layouts, by using a compaction process with two separate steps, which used low resolutions on a first step, and high resolution in the second. This approach is called *Multi-Resolution Approach*. Computational experiments show that this approach was able to achieve equivalent quality solutions to a compaction using a *HR* covering, with significantly reduced computational cost.

The second approach focused on improving the solution quality by separating the compaction process in two phases, thus being called *Two-Phase Approach*. The first phase selects big pieces, and compacts them up to a certain length, generating holes (space between the big pieces). The second phase assigns the small pieces to the holes, using specific rules, and compacts all pieces producing a layout. Due to this process, the layout may be infeasible (by not being able to solve the overlaps), therefore requiring a Post-Optimization approach to correct the infeasibilities. Computational experiments have shown that this process is able to produce layouts with superior quality than the previously presented approaches, with a reasonable computational cost.

The third and last approach focused on tackling large size problems. This approach, called *Multi-Layer Approach*, separates the pieces by groups, using a defined rule, and compacts each set, into the layout, independently of the other sets. When a set is compacted, its pieces are defined as fixed pieces and treated as forbidden placement regions. This strategy allows the NLP model to only consider adjusting the pieces that are being placed in the current set, while assigning only non-overlapping constraints between the pieces of the current set, and all the previously placed pieces. An extension has been made to this approach, that allows considering the pieces of the previously placed layer as not fixed, which allows a better adjustment between pieces from different layers, improving the solution quality at the expense of the computational cost. This approach has been shown to be able to successfully compact instances with up to 300 pieces, which allow tackling real world problems.

These approaches are able to address successfully Nesting problems with continuous orientations, taking into account their different characteristics. The results produced by all the presented approaches fall very short of the published results from the current literature, and although they are not improved in terms of quality, there is a potential for better and faster handling of large instances. However, the comparison with the results in the literature is not entirely fair, since the pub-

lished results have been compacted with geometrical representations that do not have "enlarged" outlines due to covering, or other effect. It is possible that our solutions could be equivalent, and occasionally better, than the ones presented in the literature if they are re-compacted using a polygonal representation without excess waste separating the pieces. The small difference in solution quality validates the application of the presented approaches to address the Nesting problem with continuous rotations. With some minor extensions and improvements, these approaches may be able to produce better results consistently when compared to the current literature results. Also referring to the current results in the literature, these approaches were able to successfully solve the instance *swim*, for which no results existed previously considering continuous rotations.

6.2 Future work

The future developments of these approaches consider possible paths of exploration that may lead to improvements in the solution quality, reductions in computational cost, or both. These approaches may also be extended to support other types of problems with irregular pieces, using closed containers.

The *kCC-MA* approach that generates the different types of circle covering, with controlled approximation error, can be improved by using different types of simplifications for the polygonal representation of the piece and the skeleton. The circle merging algorithm could use a different method to select which pairs of circles are merged, since the current method uses a greedy approach. One problem of the *kCC-MA* algorithm is its dependency on the correct construction of the Medial Axis, which is built following an iterative algorithm. This naturally leads to numerical precision errors which fail to produce a correct Medial Axis skeleton for pieces with very complex outlines. A great improvement over the current algorithm would be developing an approach that was not iterative, could be parallelized and with high tolerance to numerical precision errors. Another possible path is the extension of circles to ellipses, which may allow reducing the number of circles required to cover a piece, although the computational efficiency of comparing pairs of ellipses would need to be high due to the less efficient comparison between ellipses than the comparison between circles. The circle covering algorithm can be extended to 3D, using spheres instead of circles, but that requires a support structure such as the Medial Axis skeleton in 3D. This would allow tackling packing many problems with irregular 3D pieces.

The improvements to the solution approach may start by using different non-linear solvers. Our current solver (Algencan) starts from an initial solution and converges to a local minimum, but other solvers may employ different search strategies which can lead to better results. Considering the current formulations for the NLP model variants, these can be modified in order to simplify the equations that define the multiple constraints, at the expense of additional constraints, such as treating the trigonometric operations as independent constraints, whose value is used in the current constraints in the form of variables.

The post-optimization approach may still be further improved, in order to always generate a feasible solution. This may require the use of a completely different approach, based on heuristics

or linear programming. Increasing the number of feasible solutions improves the global computational efficiency, since no computational effort is wasted by producing no feasible solution.

A particular component that can be greatly improved is the generation of the initial solution. Our approach places the pieces into a grid, in random sequence and arbitrary orientations. Better approaches to the initial solution may be able to place the pieces into positions where they would adjust better with each other, and enable the solution approach to converge to better solutions.

Considering the Two-Phase approach, one aspect that can be improved is the selection of big and small pieces. The current selection method is based on their area, which does not say any relevant information about its geometry. Other aspects can be considered, such as perimeter, bounding box, medial axis, among others. This would enable a better distinction between pieces, so that they could be better assigned, depending on the characteristics of the problem. Another aspect that can be improved for the Two-Phase approach is the hole assignment and piece placement procedure. The hole assignment also requires knowing if a certain piece fits inside a hole, by analyzing not only their relative area but also their shape. The area is only useful to exclude bigger pieces, since smaller pieces may have inferior area but they may not fit due to their shape. If a given piece fits inside a hole, then the next problem to be solved is how to determine the best position and orientation to place it inside the hole. Solving this problem would allow tackling many different problems in a more efficient way, where detecting suitable holes and feasible placement positions is important. In order to improve the results of the Two-Phase approach, more extensive tests should be done to verify its behavior and select the best configuration parameters that lead to the highest quality results.

The Multi-Layer approach can be extended to include the Multi-Resolution approach and the Two-Phase approach in the individual compaction of each layer, which could lead to much better compaction results, and also greater computational cost. Selecting the best combination of pieces to be compacted at each layer is relevant to the quality of the final solution, but no extensive experiments were done. Developing an approach that could do this could enable improving the quality of the produced layout. A modification to the Multi-Layer approach that would analyze the layout as it was being constructed, to detect holes where pieces could fit, and re-enabling the compaction of previously compacted layers, could bring also improvements. An extension with such modifications to the Multi-Layer could be interpreted as a different approach to the construction of the layout.

An approach that also may bring additional improvements is the implementation of a local search method, that improves the most promising layouts produced by these approaches, by switching pieces, and making other operations over the layout.

While this work is able to solve the Nesting problem with continuous rotations, it still leaves many possible paths to explore, which can bring significant progress in solving the Nesting problem with continuous rotations.

References

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

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

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

C. Alves, P. Brás, J. Valério de Carvalho, and T. Pinto. New constructive algorithms for leather nesting in the automotive industry. *Computers & Operations Research*, 39(7):1487–1505, 2012.

N. Amenta, S. Choi, and R.K. Kolluri. The power crust. In *SMA '01 Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pages 249–266, 2001.

R. Andreani, E.G. Birgin, J.M. Martínez, and M.L. Schuverdt. On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18:1286–1309, 2007.

R. Andreani, E.G. Birgin, J.M. Martínez, and M.L. Schuverdt. Augmented lagrangian methods under the constant positive linear dependence constraint qualification. *Mathematical Programming*, 111:5–32, 2008.

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

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

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

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

J.A. Bennell and J.F. Oliveira. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:93–105, 2009.

J.A. Bennell and X. Song. A comprehensive and robust procedure for obtaining the no-fit-polygon using minkowski sums. *Computers & Operations Research*, 35(1):267–281, 2008.

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

E.G. Birgin and F.N.C. Sobral. Minimizing the object dimensions in circle and sphere packing problems. *Computers and Operations Research*, 35(7):2357–2375, 2008.

E.G. Birgin, L.H. Bustamante, H.F. Callisaya, and J.M. Martínez. Packing circles within elipses. *International Transactions in Operational Research*, 20(3):365–389, 2013.

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

H. Blum and R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167–180, 1978.

G. Bradshaw. *Bounding Volume Hierarchies for Level-of-detail Collision Handling*. PhD thesis, Trinity College Dublin, 2002.

G. Bradshaw and C. O'Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation SCA '02*, pages 33–40, 2002.

G. Bradshaw and C. O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.*, 23(1):1–26, 2004.

A. Broutta, D. Coeurjolly, and I. Sivignon. Hierarchical discrete medial axis for sphere-tree construction. *Combinatorial Image Analysis Lecture Notes in Computer Science*, 5852:56–67, 2009.

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

S. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automation*, 6(3):291–302, 1990.

B. Chazelle and D.P. Dobkin. *Optimal convex decomposition*, pages 63–133. G.T. Toussaint (Ed.), Computational Geometry, North-Holland, Amsterdam, 1985.

N. Chernov, Y. Stoyan, T. Romanova, and A. Pankratov. Phi-functions for 2d objects formed by line segments and circular arcs. *Advances in Operations Research*, 2012, 2012.

R. Cuninghame-Green. Geometry, shoemaking and the milk tray problem. *New Scientist*, 1677: 50–53, 1989.

P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14: 227–248, 1980.

H.T. Dean, Y. Tu, and J.F. Raffensperger. An improved method for calculating the no-fit-polygon. *Computers & Operations Research*, 33(6):1521–1539, 2006.

T.K. Dey and W. Zhao. Approximate medial axis as a voronoi subcomplex. In *SMA '02 Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 356–366, 2002.

T.K. Dey and W. Zhao. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algoritmica*, 38(1):179–200, 2003.

K.A. Dowsland and W.B. Dowsland. Packing problems. *European Journal of Operational Research*, 56:2–14, 1992.

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

R.G. Edwards. Determining the skeleton of a simple polygon in (almost) linear time. Technical report, Oak Ridge, Tennessee, 2010. URL http://maptools.home.comcast.net/~maptools/Skeleton.pdf.

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

M. Etzion and A. Rappoport. Computing the voronoi diagram of a 3-d polyhedron by separate computation of its symbolic and geometric parts. In *SMA '99 Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 167–178, 1999.

M. Fischetti and I. Luzzi. Mixed-integer programming models for nesting problems. *Journal of Heuristics*, 15(3):201–226, 2009.

M. Foskey, M.C. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *In Proceedings of the eighth ACM symposium on Solid modeling and applications (SM '03)*, pages 96–107, 2003.

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

A.M. Gomes and J.F. Oliveira. A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, 141(2):359–370, 2002.

A.M. Gomes and J.F. Oliveira. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3):811–829, 2006.

J.E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry, Second Edition*. Chapman & Hall/CRC, April 2004.

D.H. Greene. The decomposition of polygons into convex parts. *Computational Geometry, Advances in Computing Research*, pages 1:235–259, 1983.

R. Heckman and T. Lengauer. A simulated annealing approach to the nesting problem in the textile manufacturing industry. *Annals of Operations Research*, 57(1):103–133, 1995.

A. Heppes and H. Melissen. Covering a rectangle with equal circles. *Periodica Mathematica Hungarica*, 34(1-2):65–81, 1997.

S. Hertel and K. Mehlhorn. Fast triangulation of simple polygons. *Foundations of Computation Theory, Lecture Notes in Computer Science*, 158:207–218, 1983.

K.E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286, 1999.

P.M. Hubbard. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Dept. of Computer Science, Brown University, 1995.

T. Imamichi and H. Nagamochi. A multi-sphere scheme for 2d and 3d packing problems. *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, 4638:207–211, 2007.

T. Imamichi and H. Nagamochi. Designing algorithms with multi-sphere scheme. In *Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS '08)*, pages 125–130, 2008.

T. Imamichi, M. Yagiura, and H. Nagamochi. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6(4):345–361, 2009.

S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88:165–181, 1996.

D. R. Jones. A fully general, exact algorithm for nesting irregular shapes. *Journal of Global Optimization*, pages 1–38, 2013. ISSN 0925-5001.

M. Konopasek. Mathematical treatments of some apparel marking and cutting problems. Technical Report Report 99-26-90857-10, U.S. Department of Comerce, 1981.

L. Lam, S.-W. Lee, and C.Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

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

A. Mahadevan. *Optimization in computer aided pattern packing*. PhD thesis, North Carolina State University, 1984.

T.C. Matisziw and A.T. Murray. Area coverage maximization in service facility siting. *Journal of Geographical Systems*, 11(2):175–189, 2009.

J. B. M. Melissen and C. Schuur. Covering a rectangle with six and seven circles. *Discrete Applied Mathematics*, 99(1-3):149–156, 2000.

V. Milenkovic. Robust construction of the voronoi diagram of a polyhedron. In *Proceedings of the Fifth Canadian Conference on Computational Geometry*, pages 473–478, 1993.

B. Mirtich. Timewarp rigid body simulation. In *SIGGRAPH'00 Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 193–200, 2000.

A. Moore. The circle tree – a hierarchical structure for efficient storage, access and multi-scale representation of spatial data. *Proceedings of the 14th Annual Colloquium of the spatial Information Research Centre*, 2002.

G. Németh and K. Palágyi. 2d parallel thinning algorithms based on isthmus-preservation. *7th International Symposium on Image and Signal Processing and Analysis (ISPA 2011)*, 2011.

J.F. Oliveira and J.S. Ferreira. Algorithms for nesting problems. *Applied Simulated Annealing, Lecture Notes in Economics and Mathematical Systems*, 396:255–273, 1993.

J.F. Oliveira, A.M. Gomes, and J.S. Ferreira. Topos – a new constructive algorithm for nesting problems. *OR-Spektrum*, pages 263–284, 2000.

C. O'Sullivan, R. Radach, and S. Collins. A model of collision perception for real-time animation. In *In Proc. 1999 Conference on Computer Animation and Simulation - Eurographics Workshop (EGCAS)*, pages 67–76. Springer, 1999.

I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995.

F.P. Preparata and M.I. Shamos. *Computational geometry: an introduction*. Springer-Verlag New York, Inc. New York, NY, USA ©1985, 1985. ISBN 0-387-96131-3.

I. Ragnemalm. The euclidean distance transform in arbitrary dimensions. *Pattern Recognition Letters*, 14(11):883–888, 1933.

P. Rocha, A.M. Gomes, R. Rodrigues, and F.M.B. Toledo. Circle covering using medial axis. *Proceedings of the 11th IFAC Workshop on Intelligent Manufacturing Systems*, 11:402–407, 2013a.

P. Rocha, A.M. Gomes, R. Rodrigues, F.M.B. Toledo, and M. Andretta. Circle covering representation for nesting problems with continuous rotations. *Submitted to the Proceedings of the 19th IFAC World Congress*, November 2013b.

S.A. Sagenreich and L.M. Braga. Optimal nesting of general plane figures: A monte carlo heuristical approach. *Computers & Graphics*, 10(3):229–284, 1986.

A.K. Sato, T.C. Martins, and M.S.G. Tsuzuki. Collision free region determination by modified polygonal boolean operations. *Computer-Aided Design*, 45(7):1029–1041, 2013.

K. Siddiqi, S. Bouix, A. Tannenbaum, and S.W. Zucker. Hamilton-jacobi skeletons. *International Journal of Computer Vision*, 48(3):215–231, 2002.

Y. Stoyan, J. Terno, G. Scheithauer, N. Gil, and T. Romanova. Phi-functions for primary 2d-objects. *Studia Informatica Universalis*, 2:1–32, 2001.

Y. Stoyan, G. Scheithauer, N. Gil, and T. Romanova. Phi-functions for complex 2d-objects. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(1):69–84, 2004.

Y. Stoyan, M.V. Zlotnik, and A.M. Chugay. Solving an optimization packing problem of circles and non-convex polygons with rotations into a multiply connected region. *Journal of the Operational Research Society*, 63(3):379–391, 2012.

Y.G. Stoyan and L.D. Ponomarenko. Minkowski sum and hodograph of the dense placement vector function. Technical report, Reports of the Ukrainian SSR Academy of Science., Ser. A. v10, 1977.

P.E. Sweeney and E.R. Paternoster. Cutting and packing problems: A categorized, application-orientated research bibliography. *The Journal of the Operational Research Society*, 43(7):691–706, 1992.

S. Takahara, Y. Kusumoto, and S. Miyamoto. Solution for textile nesting problems using adaptive meta-heuristics and grouping. *Soft Computing*, 7:154–159, 2003.

F.M.B. Toledo, M.A. Carravilla, C. Ribeiro, J.F. Oliveira, and A.M. Gomes. The dotted-board model: A new mip model for nesting irregular shapes. *International Journal of Production Economics*, 145(2):478–487, 2013.

G.M. Turkiyyah, D.W. Storti, M. Ganter, H. Chen, and V. Munikumar. An accelerated triangulation method for computing the skeletons of free-form solid models. *Computer-Aided Design*, 29(1):5–19, 1997.

M. Vincze and B. Kovári. Comparative survey of thinning algorithms. *10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics*, pages 173–184, 2009.

J.M. Vleugels and M.H. Overmars. Approximating generalized voronoi diagrams in any dimension. Technical Report UU-CS-1995-14, Department of Information and Computing Sciences, Utrecht University, 1995.

B. Wang. Coverage problems in sensor networks: A survey. *ACM Computing Surveys (CSUR)*, 43(4), 2011.

H. Wang, W. Huang, Q. Zhang, and D. Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(3): 440–453, 2002.

G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

P.D. Watson and A.M. Tobias. An efficient algorithm for the regular w1 packing of polygons in the infinite plane. *Journal of the Operational Research Society*, 50(10):1054–1062, 1999.

G. Whitwell. *Novel heuristic and metaheuristic approaches to cutting and packing*. PhD thesis, University of Nottingham, 2004.

C. Yao and J. Rokne. A straightforward algorithm for computing the medial axis of a simple polygon. *nternational Journal of Computer Mathematics*, 39(1-2):51–60, 1991.

W. Zhang and Q. Zhang. Finite-circle method for component approximation and packing design optimization. *Engineering Optimization*, 41:971–987, 2009.

Y.Y. Zhang and P.S.P. Wang. Analytical comparison of thinning algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(5):1227–1246, 1993.