AN ONLINE ALGORITHM FOR THE GLASS CUTTING PROBLEM WITH DEFECTS OF MULTIPLE GRADES AND PRODUCTS WITH QUALITY CLASSES

by

Bahadır Durak

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Systems Engineering

Yeditepe University
2018

AN ONLINE ALGORITHM FOR THE GLASS CUTTING PROBLEM WITH DEFECTS OF MULTIPLE GRADES AND PRODUCTS WITH QUALITY CLASSES

APPROVED BY:

Assoc. Prof. Dr. Dilek Tüzün Aksu (Thesis Supervisor)	[illegible]
Prof. Dr. Linet Özdamar	[illegible]
Prof. Dr. Temel Öncan	[illegible]
Assoc. Prof. Dr. Güvenç Şahin	[illegible]
Assist. Prof. Dr. Semih Yalçındağ	[illegible]

DATE OF APPROVAL: ..../..../2018

ACKNOWLEDGEMENTS

At the beginning of my thesis, I would like to express my sincere gratitude to my advisor Dilek Tüzün Aksu who made this thesis possible and an unforgettable experience for me. I thank her for the continuous support of my Ph.D. study and research, her systematic guidance, patience, motivation, enthusiasm, and immense knowledge.

I must express my gratitude to Pervin Taşçı Durak, my wife, for her continued support and encouragement. Her devotion, unconditional love, sense of humour, patience and optimism was more valuable than I could ever imagine.

ABSTRACT

AN ONLINE ALGORITHM FOR THE GLASS CUTTING PROBLEM WITH DEFECTS OF MULTIPLE GRADES AND PRODUCTS WITH QUALITY CLASSES

In this thesis, we focus on the problem of placing cutting patterns on a sheet of flat glass that contains various types of defects. In flat glass production, a continuous glass sheet is cut into glass products of different sizes and different quality classes. Each quality class indicates the maximum number of defects of each type that can be tolerated on a glass product. Products that do not meet the quality requirements defined by their quality classes are considered scrap and sent back to the furnace to be recycled. In a continuous glass production line, patterns to be cut from the glass sheet have to be determined in real time, which limits the time available for decision making. The main goal of the glass cutting problem is to determine the cutting patterns in a limited time so as to minimize the total area of scrap glass. In selecting the products to be cut, daily production targets of each product type are also considered to ensure timely delivery of orders. To solve this real time glass cutting problem, we propose an online algorithm that solves a series of static cutting problems over a rolling horizon using various approaches and implements the first few cuts from each static solution to avoid a myopic decision.

In this study, we develop genetic algorithm (GA), dynamic programming (DP) and Mixed Integer Programming (MIP) based methods for solving the static cutting problem on a glass sheet of fixed size that contains defects. These methods are integrated into the online algorithm and tested using realistic instances with different defect densities. In the initial versions of the algorithm, production targets are used as constraints. Later, production targets are integrated into the objective function in order to the improve solution quality by balancing the production of different products during the production run. Moreover, an adaptive version of the algorithm, which is capable of adjusting itself based on the current status of the production run, is also provided in this study. This thesis is one of the first studies in the literature that solves a real time cutting problem with defects of multiple grades and products with quality classes.

ÖZET

FARKLI HATA TÜRLERİ VE ÜRÜN KALİTE SINIFLARI İÇEREN BİR CAM KESİM PROBLEMİ İÇİN GERÇEK ZAMANLI BİR ÇÖZÜM ALGORİTMASI

Bu tez çalışmasında, çeşitli hata noktaları içeren düz bir cam şeritten kesilecek ürünlerin cam üzerine yerleştirilmesi problemine odaklanmaktayız. Düz cam üretiminde, üretim hattı üzerinde sürekli olarak akmakta olan cam şeritten, farklı boyutlarda ve farklı kalite sınıflarında cam ürünleri kesilir. Kalite sınıfları, bir ürün üzerinde her hata türünden en çok kaç adet hatanın tolere edilebileceğini gösterir. Ait olduğu kalite sınıfının kalite gereksinimlerini yerine getirmeyen ürünler artık cam olarak kabul edilir ve geri dönüştürülmek üzere fırına yollanır. Sürekli bir cam üretim hattında, kesilecek ürünlere karar verme işlemi gerçek zamanlı olarak yapıldığından karar verme süresi kısıtlıdır. Bu problemin temel amacı, sınırlı bir süre içinde cam şeridinden kesilecek ürünlerin yerleşimini toplam artık cam alanını en aza indirecek şekilde belirlemektir. Siparişlerin zamanında teslim edilebilmesi için kesilecek ürünlere karar verirken her ürün türünün günlük üretim hedefleri de göz önünde bulundurulur. Bu gerçek zamanlı problemin çözümü için, kayan ufuk üzerinde bir dizi statik kesim problemini kesin ya da yaklaşık yöntemlerle çözerek, her statik çözümün sadece ilk birkaç kesimini uygulayan bir çevrim içi algoritma önerilmektedir.

Bu çalışmada sabit uzunlukta ve hatalar içeren bir cam şeridi üzerindeki statik kesim problemini çözmek için Genetik Programlama (GP), Dinamik Programlama (DP) ve Karışık Tamsayılı Programlama (KTP) yöntemleri geliştirildi. Bu yöntemler gerçek zamanlı çözüm algoritmasına entegre edilerek, farklı hata yoğunlukları içeren gerçekçi üretim örnekleri üzerinde test edildi. Algoritmanın ilk versiyonlarında üretim hedefleri birer kısıt olarak dikkate alındı. Sonraki versiyonlarda ise üretimin daha dengeli yapılmasını sağlayarak sonuçların kalitesini artırmak amacıyla üretim hedefleri amaç fonksiyonuna eklendi. Bununla birlikte, gerçek zamanlı algoritmanın değişen üretim koşullarına uyum sağlayabilen adaptif bir uyarlaması da çalışmaya dahil edildi. Bu tez farklı hata türleri ve ürün kalite sınıfları içeren gerçek zamanlı bir kesim probleminin çözümünü içeren ilk çalışmalardan biridir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

L	Length of glass sheet
W	Width of glass sheet
CBF	Cut behind fault
CP	Critical point
DP	Dynamic programming
EE	Explicit enumeration
GA	Genetic algorithm
MIP	Mixed integer programming
MIP+C	Mixed integer programming including production targets as constraints
MIP+CO	Mixed integer programming including production targets as constraints and an additional term in the objective function
MIP+COC	Mixed integer programming including production targets as constraints and an additional term in the objective function with a configuration limit
OGCP	Online glass cutting problem
OGCPwCBF	Online glass cutting problem with CBF
SCP	Static cutting problem
SLOPP	Single large object placement problem
TSG	Total scrap glass

1. INTRODUCTION

In this thesis, we focus on a cutting problem that arises in the flat glass production industry. Flat glass production is a continuous process in which a ribbon of molten glass is produced, cooled, carried on rollers and cut into rectangular products of varying sizes and quality specifications. This production line, where the glass sheet from the furnace is converted into glass products is called a float line (Figure 1.1).

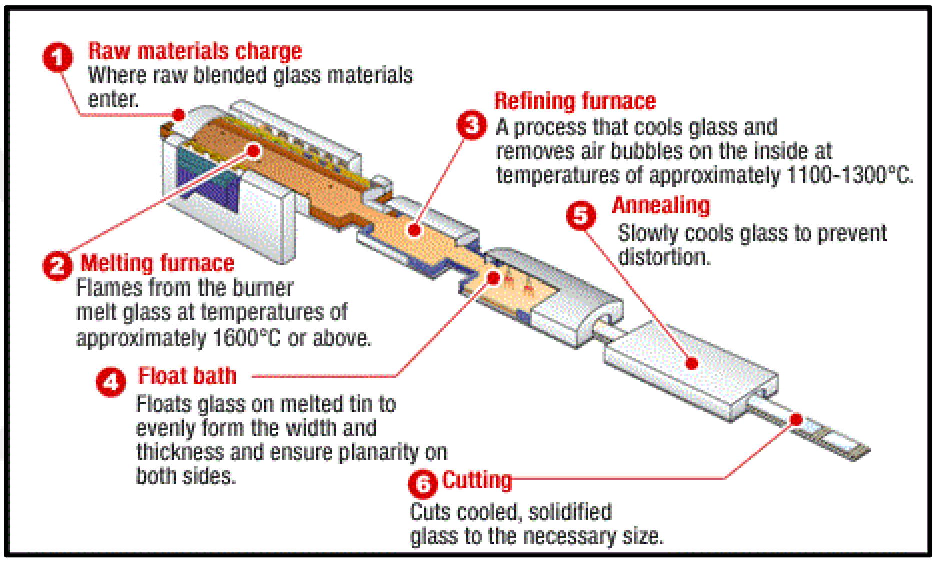

1 Raw materials charge
Where raw blended glass materials enter.
3 Refining furnace
A process that cools glass and removes air bubbles on the inside at temperatures of approximately 1100-1300°C.
5 Annealing
Slowly cools glass to prevent distortion.
2 Melting furnace
Flames from the burner melt glass at temperatures of approximately 1600°C or above.
4 Float bath
Floats glass on melted tin to evenly form the width and thickness and ensure planarity on both sides.
6 Cutting
Cuts cooled, solidified glass to the necessary size.

Figure 1.1. Flat glass production [1]

An important decision that has to be made in real time in a float line is to determine which products should be cut and how these products should be placed on the glass sheet. This problem can be viewed as a two dimensional cutting problem in which a number of two dimensional products are to be cut from a larger two dimensional stock. This decision is complicated by the existence of defects on the glass sheet and the fact that the location of these defects becomes available only a few seconds before the cutting decision is made.

The defects on the glass sheet are categorized into quality grades based on their severity. While the glass sheet moves on the rollers, the position and grade of defects that are present on the sheet are detected by a camera that has been placed on the float line. Once the defects are detected, the product patterns to be cut from the glass sheet are to be decided within a timeframe of few seconds. The cutting decision is complicated by the fact that each product has quality requirements that limit the number of defects from each quality grade allowed on the product, which prohibits cutting that product from certain defective areas. In some cases, it may be preferable to discard pieces of glass that contain defects as scrap to avoid cutting products from defective areas. A common practice is to remove strips of defective glass by cutting immediately behind a defect, which is called a Cut Behind Fault (CBF) in float line terminology. Cutting patterns that are decided are immediately communicated to cutting bridges, which score the pattern on the glass sheet in the form of vertical and horizontal lines using a cutting wheel. The lines that are parallel and perpendicular to the glass ribbon width are called x-cuts and y-cuts, respectively. Both x- and y-cuts are end to end, i.e. they traverse the entire width and length of the glass, respectively. Such cuts are called guillotine cuts in cutting literature.

Figure 1.2 depicts a glass sheet on which defects of two quality grades are marked as small circles and squares and products 1-3 are marked with vertical and horizontal lines. Shaded areas represent the scrap glass that results from cutting. Note that the same grade defect (circle) causes product 1 to be discarded as scrap whereas product 3 is acceptable despite the same defect. This is because the two products belong to different quality grades: product 3's quality class allows a circle defect while product 1's does not.

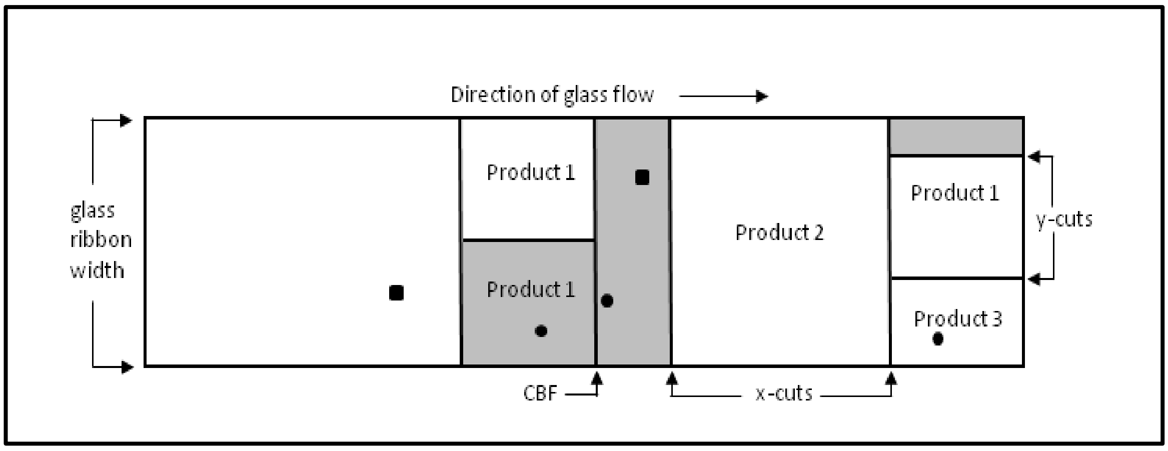


Figure 1.2. Illustration of a cutting pattern placed on the glass sheet on a float line

Depending on the speed of the glass sheet and the distance between the camera and the cutting bridges, the next pattern to be cut should be decided within few seconds. The short time frame available for decision making necessitates an efficient algorithm that can produce a solution within seconds. On the other hand, simplistic approaches may result in low quality solutions that result in large amounts of scrap. Thus, it is necessary to develop a solution approach that can produce good quality solutions within limited time.

Similar cutting stock problems occur in metal industry [2] and LCD production [3]. Rectangular products are cut offline from a flat sheet that generally does not contain defects. Unlike these industries, defects are common in many other industries such as lumber [4], furniture [5], paper [6], textile [7], and cake manufacturing [8]. Since defective products are not valuable in most of the cases, solution methods are designed for defect removal. In some production environments such as lumber manufacturing [9], defects are classified and defective products with minor defects are accepted.

To the best of our knowledge, there are a limited number of studies in the literature that investigate an online cutting problem with defects of varying severity. Moreover, few studies work with products that belong to multiple quality classes such as the one considered in this thesis. The static version of the problem, which is solved over a fixed length, differs from other problems in the literature in the way that the defects are handled. Unlike the problem in this thesis, most of the studies use defect removal instead of defective product evaluation. Considering all of these characteristics, this thesis proposes a general solution approach for various cutting problems. While this problem is inspired by the glass production industry, the static sub problem solved at each step of the online algorithm resembles other cutting problems with defects that arise in the metal, lumber, furniture, paper, LCD, textile and cake manufacturing. Therefore we expect that the solution approaches developed here can be adapted to these problems as well.

In this study, we propose an algorithm that solves the online glass cutting problem iteratively by decomposing it into a series of static cutting problems (SCP) solved over overlapping pieces of glass of a fixed length, L. In each iteration, a SCP is solved to determine the patterns to be placed on the fixed length of glass, but only a small portion of these patterns are implemented. In the next iteration, a new cutting problem is solved for the next L meters immediately after the implemented patterns. The usage of a rolling horizon for cutting decisions allows the algorithm to avoid making myopic decisions.

Two versions of the glass cutting problem are examined according to the nature of scrap cuts implemented to remove defective areas. In the first and more general version, which we call the Online Glass Cutting Problem (OGCP), scrap cuts of arbitrary length can be made between products to remove defective areas. In the second version, called the Online Glass Cutting Problem with Cut Behind Fault (OGCPwCBF), scrap cuts can occur only right after a defect. As mentioned earlier, such scrap cuts are called Cut Behind Fault or CBF. Restricting scrap cuts to CBF is a common practice in the flat glass industry. When the scrap cuts are limited to CBF, the number of locations along the length of the glass ribbon where a scrap cut can be applied is finite. While this rule simplifies the problem significantly, its impact on solution quality is not clear. By comparing the results of OGCP and OGCPwCBF, we investigate how the computational requirements and solution quality are affected by the CBF rule. A genetic algorithm (GA) and a dynamic programming (DP) approach are used to solve OGCPwCBF. On the other hand, we use different versions of a mixed integer programming (MIP) model in order to solve OGCP. All of the solution methods are later modified to solve realistic problems with production targets. In production environments, production targets is not balanced because of variable customer demand. Moreover, defect density and distribution may not be the same along the glass ribbon during the production run. Therefore, an adaptive version of MIP, which can adapt itself to changing production conditions, is also proposed in this thesis. This last version uses a multi-objective approach where scrap glass minimization and production balancing are assigned different weights in the objective function. The model is designed to estimate the future scrap glass amounts using different weights for the two objective function terms and update their coefficients in order to obtain better solution quality in the long run.

The main objective of using two different problem versions, OGCPwCBF and OGCP, is to examine the impact of the CBF rule on solution quality and time. By comparing the solutions of two versions, we try to find out whether CBF rule leads to an optimal or suboptimal solution. Similarly, we have also developed different solution methods for both OGCPwCBF and OGCP. Solution qualities of all methods are compared in order to reach the best solution methodology for each version. We have also designed new experiments for optimizing the parameter sets in all of these solution methods. Finally, we explore the effect of production targets on solution quality. For this purpose, we generate new versions of all methods which use production targets. We also tested the performance of adaptive version of MIP which takes production targets into consideration regulates itself according to changing conditions in the production line.

Experiments show us DP has better solution quality than GA; however, DP requires longer CPU times. The fact that DP cannot be guaranteed to terminate within a certain time makes it difficult to use with longer SCP lengths. On the other hand, MIP has better solution quality than both GA and DP which shows that CBF rule provides suboptimal solutions rather than optimal. Finally, we have shown that the best solution quality among all proposed methods can be obtained by the adaptive version of MIP.

The general structure of the thesis is summarized below:

Glass cutting problem has various parameters, restrictions and assumptions. All of these details are provided in Chapter 2. A simple cutting process is visualized for the reader to provide better understanding. One can find the structure of the problem and the main objectives in the same chapter.

A summary of previous studies in literature is discussed in Chapter 3. Studies on SCPs are classified under two separate sections, one-dimensional and two-dimensional. We have included another section for online cutting problems which is the main focus of this thesis. One can find the contribution of this thesis to literature in the last subsection of the same chapter.

Chapter 4 explains the solution methods in detail. The chapter starts with a description of the online glass cutting algorithm, which is solved by converting the online cutting algorithm into a series of SCPs. Solution methods for SCPs that use the CBF rule, namely the genetic and dynamic programming methods, are explained under the next section. In the next section, the reader can find the details of the MIP model which is a solution method for the SCP without CBF rule. Finally, versions of online algorithm that employ production targets are also described in the same chapter. Moreover, one can find the adaptive version of MIP in detail under the final subsection of the chapter.

Chapter 5 consist of three main subsections, experimental results of the methods without production targets, with production targets and the adaptive versions of these methods. Solution methods with production targets have two different versions. In the first version, production targets are used as constraints. Therefore, solution methods do not produce any product that reaches its production target. In the second version, the difference between the production targets and the total number of items produced so far is added to objective function for each product type with a certain weight. This new term balances the production distribution and generally delays the time that a product reaches its target. Under each subsection, reader can find how the experiments are designed. Solution qualities and computational times corresponding to each version of the solution methods are also presented.

There are many studies in the literature on cutting stock problems. The methodology developed in this thesis can be used to solve some of these problems, provided that they have structural similarities. Chapter 6 explains what these similarities are and how our methodologies can be used to solve other similar problems. We also discuss other types of cutting problems which cannot be solved by using our methodology and state the reasons.

Chapter 7 summarizes our main findings, explain the contribution of this thesis and provide suggestions for future work.

2. PROBLEM DEFINITION

In the online glass cutting problem, products $p = 1, \dots, P$ should be placed on a glass sheet of a fixed width W. The sheet has point defects d that are characterized by their x- and y-coordinates, x_d and y_d and quality grades $g_d \in 1,2,..,G$ (x_d and y_d denote the defect positions along the length and width of the sheet, respectively). Each product p is of rectangular shape, characterized by its width w_p and length l_p and value z_p. Quality of product p is specified by the number of defects allowed of each quality grade g, denoted by Q_{pg}. Moreover, the value of product p (z_p) is independent of the number of defects on the product as long as it does not exceed the allowed limit $Q_{pg} \forall g \in G$.

Certain restrictions apply to the cuts to be performed on the glass sheet:

- All cuts should be guillotine cuts performed along the entire width or length of the glass sheet.
- Cuts should be performed in at most two stages, i.e. an x-cut that cuts the entire glass width end-to-end, followed (possibly) by a y-cut that cuts the resulting rectangle into smaller rectangles.
- In some cases, the rectangular pieces resulting from the two-stage cutting process can be further trimmed down to the required size outside the production line.
- If a product p is placed on an area which does not meet its specifications (i.e. for at least one grade g, the number of defects of grade g in that area is greater than Q_{pg}) the product is considered as scrap.
- The number of y-bridges that hold the cutting wheels for scoring y-cuts is limited (usually to 3-5 y-bridges per float line) and the positions of the cutting wheels on these bridges are fixed during the production run. Therefore, there are a finite number of configurations (i.e. cutting patterns) that can be cut along the y-axis. These possible configurations are defined in the algorithm before the cutting process begins.
- There is no restriction on the number of products of each type that should be placed on the glass sheet. The current practice is to adjust the product values (z_p) in real time during cutting process to balance the demand and production.

However, this assumption is relaxed in the latter versions of our solution methodology and production targets are taken into consideration during production run.

- There should be a minimum distance between two consecutive x-cuts. This minimum distance depends on technical restrictions and the speed of float line.

The primary objective of the glass cutting problem is the minimization of the total area of scrap glass, followed by the maximization of the total value of cut products. In this thesis, we assume that product value per unit area is the same for all products. Therefore considering only scrap minimization is sufficient since the value of remaining glass sheet will have a constant value independent of the product quantities. However, one can use the same solution methodology with different product unit values.

An important aspect of the glass cutting problem is the daily production targets that should be met for each product type. As the cutting decisions are made on an online basis, daily production targets of each product type should be considered to ensure timely delivery of orders. At the beginning of the day, products to be cut may be determined mostly based on the minimization of scrap and maximization of product value objectives. However, as the day advances, it may become necessary to implement cuts of high scrap and/or low value to produce products that are well below their targets. Therefore, the solution proposed for the online glass cutting problem should be adaptive, i.e. as the production day progresses, it should be able to adapt to the changing priorities of the line so that all targets of all product types for that day are achieved. As mentioned above, the current practice is to manually adjust the product values in real time during the production process. The online algorithm to be developed in this thesis should eliminate the need for such manual intervention.

As mentioned above, the number of possible configurations along the y-axis is limited and these configurations are known in advance. Thus, in order to reduce the problem to a one dimensional problem along the length of the glass sheet, we can first enumerate all possible cutting wheel configurations that can be implemented using the available bridges. Figure 2.1 illustrates four possible configurations of products 1-3 that can be scored with one y-bridge on which four cutting wheels are positioned at 0.1m, 1.1m, 2.1m and 3.1m. Note that depending on the configuration to be scored, cutting wheels can be lifted up to make them inactive which makes it possible to score a wider range of configurations using a single y-bridge. For instance, in Configuration 2, only the cutting wheels at y=0.1m, 2.1m and 3.1m are active. Also note that there are two distinct configurations (Configurations 2 and 3) that consist of one of products 1 and 2 each. On a defective sheet, the value of these two configurations may vary depending on the distribution of defects in the area where the configuration is placed. Having enumerated all possible configurations, we can treat each configuration as a distinct item, for example, Configuration 1 is an item of length 1.5m in a one dimensional problem. Width of all configurations is equal to the glass sheet width, W. Defining the cutting problem over configurations of fixed width instead of products of varying length and width allows us to reduce the cutting problem from a two dimensional problem to a one dimensional one.

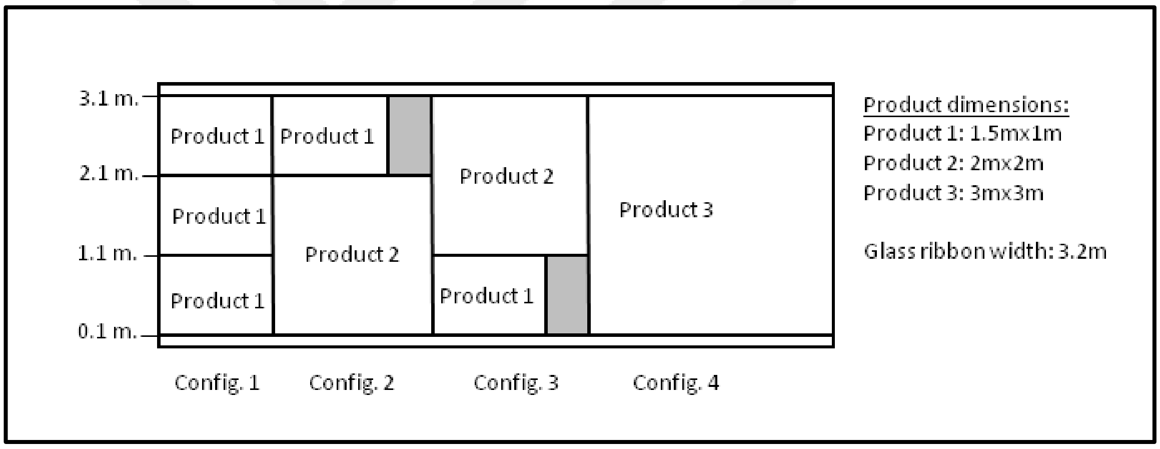


Figure 2.1. Four configurations of products 1-3 that can be scored with one y-bridge on which four cutting wheels are positioned at y=0.1m, 1.1m, 2.1m and 3.1m.

3. LITERATURE REVIEW

There is a large body of literature on cutting problems, which deals with placing a number of small objects of varying sizes on one or more larger objects. One can refer to survey articles published on this subject for a review of the available literature [10], [11] and [12]. In recent years there have been many studies focusing on industrial cutting applications, see [13], [14] and [15] from the automotive, window frame and LCD industries, respectively. One of the recent papers by Wäscher et al. [16] provides a topology for the categorization of cutting problems and classifies the existing literature according to this topology. According to their topology, the static problem solved in each iteration of the proposed online algorithm is a variant of the two dimensional Single Large Object Placement Problem (SLOPP), where a set of weakly heterogeneous small objects have to be placed on a single large object such that the total value of the small objects placed are maximized. The term weakly heterogeneous refers to the case where the small objects can be grouped into relatively few classes (in relation to the total number of objects), for which the objects are identical with respect to shape and size. In our version of this problem, the value of the small objects depends on their placement on the large object: if the small object is placed in an area where an unacceptable number of defects exist, the small object has no value because it has to be discarded as scrap. In the review below, studies on both one and two dimensional cutting problems with defects are discussed in Sections 3.1 and 3.2, respectively. Most of these studies are on static problems, but there is also a limited amount of work on online problems, which are reviewed separately in Section 3.3. A list of studies in the literature that are relevant to our thesis work is shown in Table 3.1 with their basic problem features. All of these studies are explained in the remainder of this chapter.

Table 3.1. Comparison of problem structures of references where * indicates the problem studied in this thesis

Reference	**Year**	**Online**	**n-dimensional Problem**	**Multiple Defects**	**Defect Types**	**Product Quality Grades**	**Rectangular Stock**	**Rectangular Product**	**Guillotine-type Cutting**	**Production Targets**
*****		X	2	X	X	X	X	X	X	X
[4]	**1984**		1			X				
[5]	**2005**	X	2	X	X	X	X	X	X	X
[6]	**1998**		1							
[7]	**2000**		1	X		X				
[17]	**1965**		2	X			X	X	X	
[18]	**1968**		2	X			X	X	X	
[19]	**1966**		2				X	X		
[20]	**2013**		2				X	X		
[21]	**2014**		2				X	X		
[23]	**2009**		2	X			X	X	X	
[24]	**1999**		2	X			X	X	X	
[25]	**2015**		2	X						
[26]	**2014**		2	X			X	X	X	
[27]	**2009**		2				X	X	X	
[28]	**1981**		1	X						
[29]	**1988**		1	X			X	X	X	
[30]	**1990**		1			X				X
[31]	**2016**	X	2	X	X	X	X	X	X	

3.1. TWO-DIMENSIONAL CUTTING PROBLEMS WITH DEFECTS

The literature on two dimensional cutting problems with quality variations and defects is rather limited, most of them are on static versions of the problem. One of the first studies in the cutting literature where defects have been discussed is the seminal work of Gilmore and Gomory [17]. In this paper, the authors study the one and two dimensional cutting stock problem, in which all small objects should be placed on a minimum number of large objects (stocks) of fixed dimensions. In addition to the standard cutting stock problem, they consider the case where the value of the small object depends on its position on the large object. Position dependent values allow the authors to account for defects and quality variations on the large object. They suggest a recursive equation to solve both problems.

Hahn [18] studies a two dimensional SLOPP that may arise various industries. The author works with rectangular stocks which contain multiple rectangular defects. There are small objects of various dimensions to be placed on the stock. They also have rectangular shapes and their dimensions are known in advance. These small objects should not be placed on the areas that contain defects. The cutting process is completed in three stages. All cuts are restricted to be guillotine cuts along x or y axis. In each cutting stage, guillotine cut on either x or y axis is used. In first stage, the stock is split into sections. Similarly, sections are split into strips in the second cutting stage. Finally in the third cutting stage, strips are cut into pieces which form the small objects of the problem. The main objective is to find a solution with minimum waste of material. The author applies the dynamic programming method suggested by Gilmore and Gomory [19] to solve the problem.

[20], [21] suggest corrections and improvements in computations over the proposed DP algorithm of Gilmore and Gomory [19] for the cutting stock problem without defects. In [20], the authors suggest three corrections on the solution proposed in [19]. The first correction is for an error that was previously discovered by Herz [22] in 1972. The other two errors are discovered by the authors. In [20], they suggest corrections for all three errors regarding incorrect elimination of feasible solutions. The second paper of the authors [21] suggests five different computational improvements for the same solution methodology introduced by Gilmore and Gomory [19].

More recently, Gelder and Wagelmans [23] focus on a problem that arises in roller blind production. Small rectangular objects are placed on a rectangular stock to produce window covering products. The stocks are fabrics which may contain small defects. No defects are accepted on products, therefore avoiding defects is crucial in order to produce valuable products. Two cutting stages are used in the production process. In both stages, only guillotine cuts are allowed. The first stage splits the stock by using a vertical guillotine cut to create a smaller rectangular part called shelf. This shelf is used as a stock in the second cutting stage. The second stage uses horizontal guillotine cuts on the shelf to produce window covers of various dimensions. The objective of this problem is minimization of waste. For this purpose, authors suggest a two-stage heuristic where both stages are supported by integer programming models. In the first stage, an integer programming model is solved in order to find the products to be cut and to determine the length of the shelf. In second stage, another model is run to decide the placement of the selected products on the shelf. Defect locations are a major input for this decision.

Twisselman [24] studies another cutting problem that arises in the steel industry. Stocks, products and defects are all in rectangular shapes. Only guillotine cuts are allowed such that in each cut two smaller rectangular parts are produced. The objective is to find all usable rectangular pieces that can be produced using guillotine cuts from a rectangular large object with multiple rectangular defects. Twisselman focuses on the problem with a distinctive perspective and considers it as a maximum empty rectangle problem which is solved by a two stage solution methodology.

Wenshu et al. [25] study a problem that emerges in wood processing. The problem is a cutting stock problem where stocks are decayed wood boards of different dimensions. Unlike previous studies explained in this section, the objective of this problem is not the placement of products on the stock. Instead, removal of any defective area with minimum material loss is desired. In their study, they start with feasible cutting patterns, turn them to binary genes and use a genetic algorithm to improve solution quality.

In their paper, Afsharian et al. [26] deal with a two-dimensional cutting problem where both stocks and small items are in rectangular shape. The only cutting option is guillotine-type cuts, however the number of stages is unlimited. There are multiple defects of irregular shapes on the stocks and no small item must overlap with a defective region. Maximizing the total value of small items is the objective of the problem. Authors develop a heuristic for this two-dimensional cutting stock problem. They use a dynamic programming approach and divide the single stock into small parts in each iteration to form sub problems. Combined solutions of these sub problems forms a solution to the original problem.

There are other authors who study problems where the large object contains a single defect. For instance, Neidlein et al. [27] study the two dimensional SLOPP with a single rectangular defect. Cuts are restricted to be guillotine-type and the objective is maximization of the total value of the small items. Small items must not overlap with the rectangular defect. They develop a branch and bound algorithm based on AND/OR graphs. In order to keep computational times reasonable, they improve their solution methodology with performance heuristics.

3.2. ONE-DIMENSIONAL CUTTING PROBLEMS WITH DEFECTS

In the literature, there are also several studies that concern one dimensional cutting problems with defects. [28] studies a cutting problem that arises in insulating tape production. The author considers a variant of the one dimensional cutting stock problem where small rolls of fixed size are to be cut from a long roll with defects. In this variant, the size of the small rolls is a random variable due to the defects that exists on the long roll. After the positions of the cutting knives are fixed, large rolls are cut to produce identical small rolls regardless of the size of large roll. If it is needed to produce other sizes of small rolls, positions of the knives should be changed before production. The objective of the problem is to find where to place the first knife on the large roll in order to minimize long term wastes. The author presents both exact and approximate solutions for different versions of the problem.

[4] studies a cutting problem in lumber manufacturing. The production process has two main steps. The first step is bucking the tree where a tree is cut into shorter logs. Second step is sawing these logs further into lumber. The defect information is collected by an electronic scanner. Tree segments are classified into four quality grades based on their defects. The value of a product depends on both its size and the quality grade of tree segment from where it is produced. In this problem, the size of the stock is not fixed and the stock’s shape is not rectangular. Moreover, rotating the logs by any angle before the cutting process is also possible in order to gain advantage by the changing positions of quality segments. The main objective of the problem is to maximize the total value of the cut lumbers. They use a staged dynamic programming algorithm to solve this realistic problem taken from the timber industry.

[29] considers a variant of the one dimensional SLOPP, where the large roll contains several punctual defects. The production is completed in two main steps. The first step is slitting the large roll into several small rolls and the second step is to cut rectangular pieces

from the small rolls. The objective is to achieve the maximum possible total value of products. The value of a rectangular product increases by its size. On the other hand, its value decreases by the number of defects on it. It is obvious that the number of defects on a product is proportional to its size, therefore the proposed solution methodology is designed to balance these two factors in order to optimize the total value. Author suggests a dynamic programming based solution methodology to achieve this objective.

Sweeney and Haessler [30] study a one dimensional cutting stock problem, where the large object (roll) contains various sections of hierarchical quality grades and the small objects (customer orders) have minimum acceptable quality grades. The authors tackle this problem by a two stage sequential heuristic in which a shadow price based procedure is used to select slitting patterns and the residual problem for the available first quality rolls is solved using linear programming. Since customer orders should be satisfied, they include orders as constraints in their solution methodology. The objective is to minimize the total trim loss.

[6] considers a problem from the paper industry, where a roll of paper that contains a single defective area is to be cut vertically to produce a set of sheets. All cuts are guillotine-type cuts and no quality grades exist except that all defects on the paper roll should be discarded. The objective is to find a cutting pattern that minimizes the total length of the defective sheets. Authors suggest a branch and bound method to solve the problem. Moreover, they support their solution methodology with two heuristics to reduce the branch and bound search and reach the optimal solution.

[7] focuses on a one dimensional cutting and wrapping problem from the textile industry. In this problem, a long piece of fabric with multiple defects is first cut into smaller pieces, each of which are assigned a quality grade based on the defects it contains. Pieces with the same quality grade are then wrapped to create rolls of given quality specifications. Quality specifications define the selling price of the fabric roll. On the other hand, there is a cost for handling and transporting the rolls. The main objective is to maximize the total profit of the company by considering both revenue and costs. The author suggests a MIP model for the solution of the problem. However, due to insufficient performance of the model in terms of computational time, the author proposes mutative simulated annealing as an alternative solution approach.

3.3. ONLINE CUTTING PROBLEMS WITH DEFECTS

As mentioned above, literature on static cutting problems is quite extensive; on the other hand work on online cutting problems is very limited. To the best of our knowledge, the only two studies in the literature that concern cutting of defective items in an online setting are by Ghodsi and Sassani [5] and Tuzun Aksu and Durak [31].

In [5], the authors introduce a new variant of the cutting stock problem with variable stock size that arises in solid wood furniture production. Each stock (wood strip) has both defective areas which need to be removed and quality variations along the length of the strip. The strips are produced online and cut into various items in real time; in this respect this problem is similar to the one proposed here. However, the two studies differ in the way that the defects and quality are treated. In [5], defects have to be removed completely by guillotine cuts applied along the width of the strip and all non-defective areas are separated into quality intervals such that within an interval the entire area has the same quality grade along the entire width of the strip, which reduces the problem to one dimension. In our problem, we consider point defects which only affect the items placed on the defective area (i.e. the item has to overlap with the defect on both dimensions), that does not allow us to reduce the problem to one dimension entirely. Furthermore, in [5] same quality grades are used for the strip and the items, in parallel to the quality definition of Sweeney and Haessler [30]. However, in our case, there is no one-to-one correspondence between the grades of defects and the quality grades required for each product. The quality grade of a product is defined as the number of defects of each grade that can exist on the product. Moreover, even if the problem in [5] is an online problem, the stocks are discrete. Therefore any remaining part on the stock with a length less than the minimum product length is discarded as waste. However, in our problem, any remaining part in the solution of SCP is evaluated in the next SCP again. Similar to our solution approach, [5] tries to minimize total waste while balancing the production targets. They use a sub-algorithm in order to prioritize the products by considering their lengths and quantities. On the other hand, our solution methodology provides a single model for both waste minimization and production target balancing.

In [31], we proposed the initial version of algorithm for the online glass cutting problem studied in this thesis. While [31] also considers the online glass cutting problem, it has three major differences from this thesis. First, it employs only the DP method to solve the static cutting problem, which limits the cuts to be made to CBFs. Second, it does not take into account the production targets specified for each product. And finally, it does not have any adaptive features that enable the algorithm to adjust itself according to the conditions of the production line. As we will show in the rest of this thesis, the algorithm proposed here is superior to the one in [31] in that it solves a more realistic version of the problem and offers better solution quality.

3.4. CONTRIBUTION TO LITERATURE

Although there are various studies on cutting problems that are similar to the one which proposed in this thesis, there are significant differences in terms of defect definitions, quality grades of products and cutting techniques. Moreover, all studies except [5] and [31] focus on static problems where the locations of defects and defect types are already known in advance. However, the proposed solution approach is developed for an online cutting problem where the information of defects become available in real time.

- A unique problem in terms of different types of defects and various product quality classes:

The online problem considered here includes defects of different types and products that belong to multiple quality classes. To the best of our knowledge, this online version of the problem has not been studied before in the literature. There are various product quality classes and a number of defect types in the problem here. Each quality class is defined with upper limits on the number of defects allowed of each defect type. Therefore, a product is valuable only if all types of defects on the product are within the defined limits of its quality class. As a results, the value of a configuration depends on its location on the glass sheet. Thus, our solution methodology needs to evaluate all of the products according to their positions on the glass sheet and their quality classes.

- Look-ahead approach for the online problem:

The proposed solution methodology also provides new perspective for dealing with both online and static cutting problems in general. The look-ahead feature, which avoids making myopic decisions, provides a framework for solving other similar online cutting problems.

Moreover, this framework is not limited to cutting problems and it can be used for solving various online problems such as online job scheduling. In addition, the solution methods that are developed for solving the static sub problem can also be adapted to other cutting problems with and without defects that arise in various other industries, as will be discussed in Chapter 6.

- Impact of the CBF rule:

Currently, the CBF rule is widely used in the glass cutting industry to limit the location of scrap cuts. However, the impact of this rule on solution quality and computational requirements has not been investigated. In this thesis, we propose solution methods both with and without the CBF rule and compare their performance to provide a quantitative answer to this research question.

- An adaptive model which adjusts its parameters to production conditions:

In a real time production environment, cutting decisions should also be made considering the production targets. Therefore, one important feature of the online glass cutting algorithm is its capability to adapt to the changing priorities of different product types so that the production targets can be met. Another contribution of this thesis is to provide an adaptive method that can be used in industrial glass cutting applications. In fact, this adaptive method can also be utilized to prioritize production in other industries where customer orders need to be considered in a real time fashion. Since the adaptive approach proposed here does not have any context specific features, it can readily be used in real time production settings from other industries.

In summary, this study provides a practical approach that can be readily implemented to achieve immediate improvement in productivity. Considering that automation of float lines has increased significantly in the last decades, online optimization approaches have become essential to improve the productivity of float lines. Therefore the main contribution of this thesis is to provide a solution for a cutting problem of practical significance. Moreover, the sub problem, which is solved in each iteration of the proposed algorithm, is a static cutting problem with multiple defect grades. The problem arises in many industrial settings such as the production of paper, fabric, metal, lumber, furniture and LCD. Therefore the solution approaches proposed here can also be implemented for similar cutting problems with minor modifications. The applications of our methodology in similar problems from different industries are discussed in Chapter 6.

4. METHODOLOGY

Our solution methodology uses a look-ahead approach to solve a series of partially overlapping sub problems to reach a solution for the online glass cutting problem. Since defect information for the entire glass ribbon is not available at the beginning of production, an online approach becomes necessary. Only a limited area can be scanned by cameras, therefore defect information of a limited length of glass ribbon is available for decision making. In the remainder of this chapter, we will first describe the online glass cutting algorithm in general and then discuss solution approaches for the static sub problems to be solved at each iteration of this algorithm.

4.1. ONLINE GLASS CUTTING ALGIRITHM

The online glass cutting algorithm iterates over a series of SCPs. At each iteration, a static one dimensional cutting problem of fixed length L is solved. Once the static solution is obtained, the first m configurations are communicated to the cutting bridges for execution. The value of m is chosen as a small integer, usually 1 or 2. This way, only a small part of the static solution is implemented, whereas the rest of the static solution is computed solely to account for the impact of the implemented solution on the upcoming solutions. Once the first m configurations of length l_m is communicated to the cutting bridges, the algorithm advances its starting point $start$by l_m to solve a new static problem between coordinates $start$ and $start +\ L$. The flowchart of the look-ahead algorithm is given in Figure 4.1.

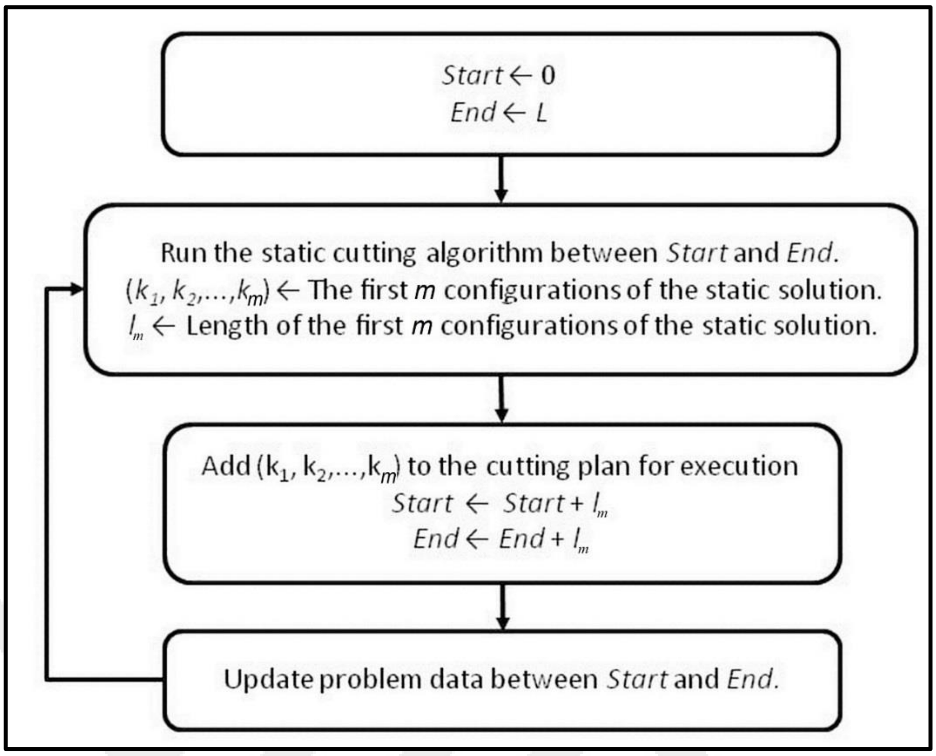

$Start \leftarrow 0$
$End \leftarrow L$
Run the static cutting algorithm between *Start* and *End*.
$(k_1, k_2, \ldots, k_m) \leftarrow$ The first m configurations of the static solution.
$l_m \leftarrow$ Length of the first m configurations of the static solution.
Add $(k_1, k_2, \ldots, k_m)$ to the cutting plan for execution
$Start \leftarrow Start + l_m$
$End \leftarrow End + l_m$
Update problem data between *Start* and *End*.

Figure 4.1. Flowchart of the online glass cutting algorithm

4.2. SOLUTION OF STATIC GLASS CUTTING PROBLEM UNDER THE CBF RULE

When the scrap cuts are limited to CBF, scrap cuts can occur only immediately following a defect, which limits the location of scrap cuts. As explained in the next three subsections, this allows us represent scrap cuts using a dummy configuration. Under the CBF rule, we proposed to use two approaches to solve the static glass cutting problem: a genetic algorithm (GA) and a dynamic programming (DP) algorithm. Both algorithms try to find the configuration sequence with the best objective function value. Please note that this configuration sequence may include one or more dummy configurations between product configurations to allow for the usage of CBF.

4.2.1. Explicit Enumeration

Line automation systems used in the glass industry also use an algorithm similar to the one displayed in Figure 4.1, except for a major difference: instead of solving a static problem over a fixed length L at each iteration, the current practice is to use explicit enumeration (EE) to select the best alternative among all possible permutations of c consecutive configurations and implement the first c' configurations from the selected permutation. In order to consider solutions that contain scrap cuts, applying a CBF is also evaluated as a dummy configuration during this enumeration. We have also implemented a version of the online algorithm that uses this EE approach to use as a benchmark in our computational study.

4.2.2. Genetic Algorithm

In our genetic algorithm, the SCP is represented by a permutation encoding, where a solution is simply defined as a permutation of configurations that can fit within a length of L. However, since the length of each configuration is variable, the number of configurations that can fit on a glass sheet of length L may vary. To remedy this, we create a sufficiently long chromosome and use only the portion of the chromosome that corresponds to a feasible solution, which is denoted as the valid length of the chromosome. Another issue with this approach is how to allow strips of scrap glass between configurations in the form of CBF. In the permutation encoding, we represent a CBF by a dummy configuration, say configuration 0 (E.g. Figure 4.2 displays a chromosome encoded as (3,1,0,2,1,2,3) that has a valid length of 5 configurations including one CBF). A series of consecutive configuration 0's denotes cutting behind a series of consecutive defects. Due to technical restrictions, there should be a minimum distance between two consecutive x-cuts. Therefore, configuration 0 has a minimum feasible length. When a defect is closer than this length, GA performs the CBF (configuration 0) after this minimum feasible length and sacrifices a larger glass strip. As a crossover operator, we use a single point crossover which we have modified so that the crossover only occurs at a position that corresponds to a cutting solution. To ensure this, when we crossover two parents i and j with valid lengths vl_i and vl_j, we choose the crossover point R to be in the range $1 \leq R \leq min(vl_i, vl_j)$. This way, we prevent the crossover operator from producing an offspring that represents the same solution as its parent. For instance, Figure 4.3 displays the crossover of the chromosome in Figure 4.2 with another chromosome of valid length of 4 that results in two new offsprings of valid lengths of 3 and 5, respectively.

The classical mutation operator is also modified so that the gene position F to be mutated is within the valid gene range of the selected chromosome i, i.e.$1 \leq F \leq vl_i$. As for the fitness value of a chromosome, we use the total value of products minus the total area of scrap glass generated by the corresponding static solution.

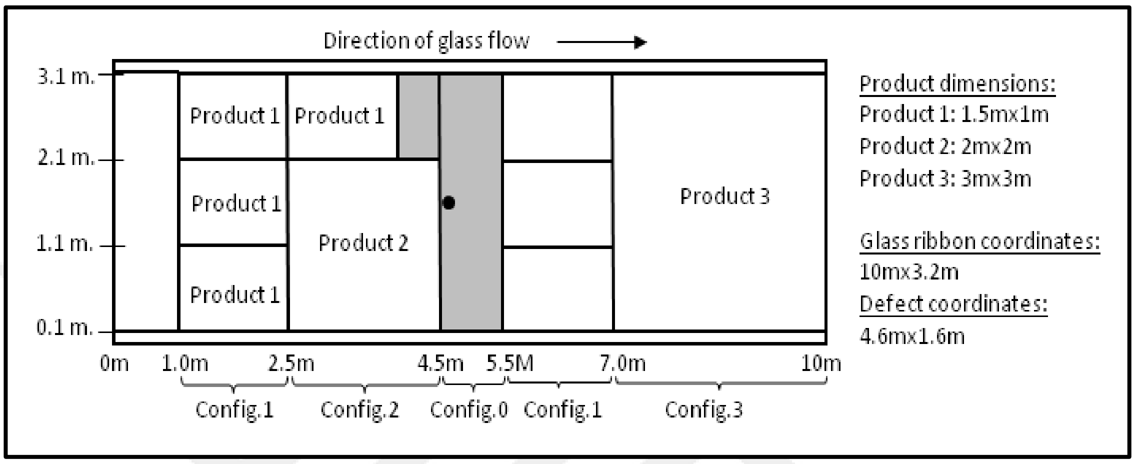


Figure 4.2. Cutting solution represented by chromosome (3,1,0,2,1,2,3) with a valid length of 5 chromosomes on a glass sheet of length 10m.

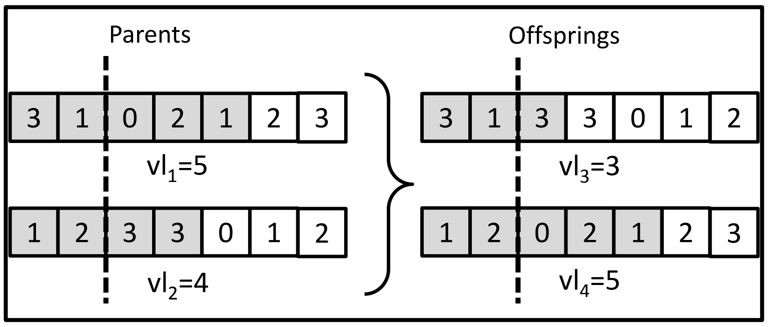


Figure 4.3. Crossover of two chromosomes with valid lengths of 5 and 4 (valid lengths shaded in grey) at crossover point R = 2.

The advantage of GA is that it is guaranteed to terminate with a feasible solution within the allotted time. On the other hand, it does not guarantee the optimal solution for the static problem and the time limit may affect the solution quality negatively.

4.2.3. Dynamic Programming Algorithm

The dynamic programming (DP) procedure described below is inspired by the approach proposed by Gilmore and Gomory [17] for the cutting stock problem variant where the value of each item is dependent on its position on the stock. However, significant modifications are needed to account for the case of multiple discrete defects and the option of using CBF.

Equation (4.1) shows the original recursive function proposed by Gilmore and Gomory. Here, $F_s(x)$ denotes the best value of a stock of length x when only the first $s$items are used. Π_s and l_s are the value and length of item$s$ respectively. If item s is used in the stock, then $F_s(x) = \Pi_s + F_s(x - l_s)$, otherwise $F_s(x) = F_{s-1}(x)$. Since $F_1(x) = \Pi_1[x/l_1]$ and $F_s(0) = 0$ are initially defined, $F_s(x)$ produces the best value for stock of length x.

$$F_s(x) = \max_s\{\Pi_s + F_s(x - l_s), F_{s-1}(x)\} \quad , \quad for\ s > 1 \tag{4.1}$$

Similar to the recursive function in Equation (4.1), the DP procedure that we propose calculates the best sequence of configurations and CBF's to be placed on the glass sheet of length L. The state of the DP procedure is defined by the x-coordinate of the point (between 0 and L) for which a cutting decision is under consideration. The optimal objective function value is calculated as $V(L)$ using the recursive procedure outlined below. Before stating the procedure, the following definitions should be noted:

L : length of the glass sheet to be cut (glass sheet is positioned between x-coordinates 0 - L),

N : number of all possible product configurations that can be obtained by positioning cutting wheels on y-bridges,

D : number of defects on the glass sheet,

x_d : x-coordinate of defect d on the glass sheet, $d = 1,2, \dots, D$,

l_i : length of configuration i along the x-axis, $i = 1,2, \dots, N$,

$\pi_i(x)$: value of configuration i when it is placed between x-coordinates x and $x - l_i$, $i = 1,2, \dots, N$,

μ : value of scrap glass per unit length (The width of the scrap glass is constant at W and the value of scrap glass is independent of its position on the glass sheet).

$\delta(x)$: Length of the scrap cut performed behind the first defect after point x in the direction glass flow. Note that if the distance between x and the defect is less than $MinCut$, the length of the scrap cut is increased to M, i.e.:

$$\delta(x) = \begin{Bmatrix} Max\left(x - \max_{d:\, x_d \leq x} x_d \,, Mincut\right) \\ x \end{Bmatrix} \begin{matrix} \exists d \,:\, x_d \leq x \\ \nexists d \,:\, x_d \leq x \end{matrix} \qquad (4.2)$$

$V(x)$: the maximum value that can be obtained for the portion of the glass sheet that lies between coordinates 0 and x.

One should note that $\pi_i(x)$, i.e. the objective function contribution of configuration i placed at coordinate x, should be calculated based on the position of the products in configuration i and the coordinates of defects between x-coordinates $x - l_i$ and x. The value of μ, i.e. the value of scrap glass per unit length and width W, can be assumed 0, or any value deemed appropriate by the decision maker. If the decision maker deems scrap to be undesirable, its value can be assigned a high negative value to discourage cuts resulting in scrap glass. In general, the values of π_i and μ should be set based on the importance of the two objectives stated previously (scrap area minimization and product value maximization) for the decision maker.

Based on the notation described above, the DP procedure can be defined as follows.

Step 1. (Initialization)
1.a) List all configurations i that can be obtained using available y-bridges,$(i = 1,2,\dots,N)$.
1.b) Calculate $l_i,\ \forall i = 1,2,\dots,N$.
1.c) Initialize $V(x),\ \forall x < \min_i\{l_i\}$:

$$V(x) = \mu \,.\, x \quad \forall x < \min_i\{l_i\} \qquad (4.3)$$

Step 2. (Recursion)
Calculate $V(L)$ using the recursive function below:

$$V(x) = max \begin{Bmatrix} \max_{i:x \geq l_i}\{\pi_i(x) + V(x - l_i)\} \\ \mu\delta(x) + V(x - \delta(x)) \end{Bmatrix} \qquad \forall x \geq \min_i\{l_i\} \qquad (4.4)$$

In the above recursive function, the first argument calculates the maximum value that can be obtained by placing any configuration i between coordinates x and $x - l_i$ of the glass sheet (see Figure 4.4). The second argument gives the value that can be obtained by placing a cut behind the next defect after coordinate x (i.e. a cut behind fault, or CBF performed after x) (see Figure 4.5). One should note that there should be a minimum distance of $MinCut$ between two consecutive x-cuts, which is a technical restriction due to the set up of the float line. When a CBF results in a scrap cut of length less than $MinCut$, $\delta(x)$ is set as $MinCut$ to prevent a technically ineasible cut. By taking the maximum over both arguments, the recursive equation finds the maximum value that can be obtained by either placing a configuration or performing a CBF at point x.

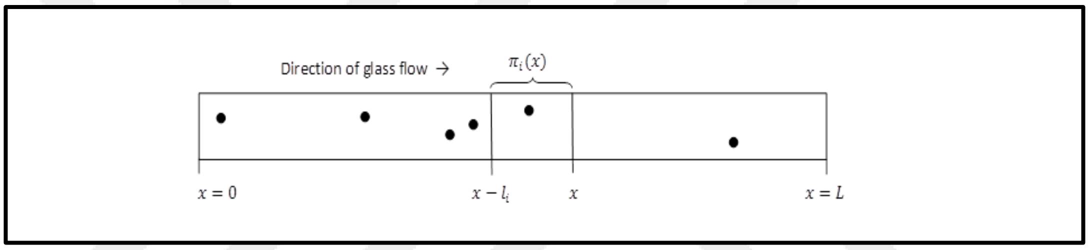


Figure 4.4. Illustration of the first argument in the recursive function in Equation (4.4).

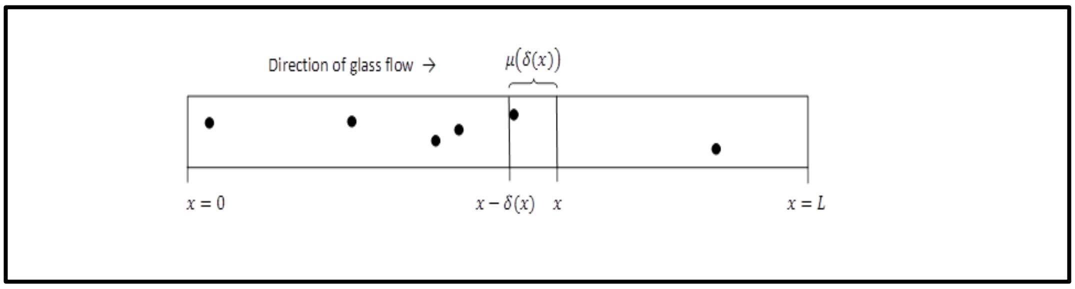


Figure 4.5. Illustration of the second argument in the recursive function in Equation (4.4).

The advantage of DP procedure is that it finds the optimal solution for the SCP under the CBF rule. However, for large static problem lengths and high defect densities, it may not give a solution within allotted time.

4.3. SUBOPTIMALITY OF THE CBF RULE

CBF is a common practice used in glass cutting industry. However, as we demonstrate in this section the rule of CBF does not guarantee an optimal solution. This finding lead us to develop new algorithms that do not rely on the CBF rule.

In the next two subsections, we provide two examples where the CBF rule leads to a suboptimal solution. This first example in Subsection 4.3.1 is for the static problem solved at each step of the OGCP. The second one in Subsection 4.3.2 demonstrates the sub optimality of CBF for the classical one dimensional cutting stock problem with defects. In both subsections, the solutions under the CBF rule are generated using the DP method described in Subsection 4.2.3 and the solutions that do not rely on the CBF rule are created using the MIP model described later in Section 4.4.

4.3.1. Suboptimality of the CBF Rule for the Static Problem in the Online Version

In the first example, we provide a counterexample that proves the sub optimality of CBF for the SCP in the online glass cutting algorithm where the glass area behind the configurations is not considered as scrap glass since it will be used in the next static problem. Solutions of the problem with and without the CBF are shown in Figure 4.6 (a) and (b) respectively. In these figures $\star$, $\times$ and $+$ represent defect types 0, 1 and 2 respectively and the numbers represent product types. The defect tolerances of all products according to their quality classes are listed in Table 4.1. There are two critical defect points (of grade 2) at $x =$106mm and $x =$2,731mm. Under the CBF rule, the algorithm can remove the first defect by cutting just behind it. However the minimum cutting distance restriction forces the algorithm to cut a minimum length of 500mm, thus the first cutting decision is made at $x =$500mm. After cutting two configurations, the algorithm encounters with the second critical defect. The CBF rule and the minimum cutting distance restriction force the algorithm to cut another glass strip of length 500mm. On the other hand, it is possible to sacrifice some more glass at the beginning by cutting the first strip at $x =$601mm. Since the 601mm long strip satisfies the minimum cutting distance restriction, this cut is a valid decision. Moreover, using this decision we can also overcome the second critical defect with less scrap, since the type 2 defect now lies within the second configuration. Only 1/3 of the second configuration is discarded as scrap and there is no need for the next scrap cut. Sacrificing some glass to save more in the subsequent area of the glass sheet is the reason why the second solution has better solution quality. In this problem, algorithm with the CBF rule creates 4,354,900mm^2 scrap glass whereas without the CBF rule the second solution yields only 4,215,800mm^2 scrap glass.

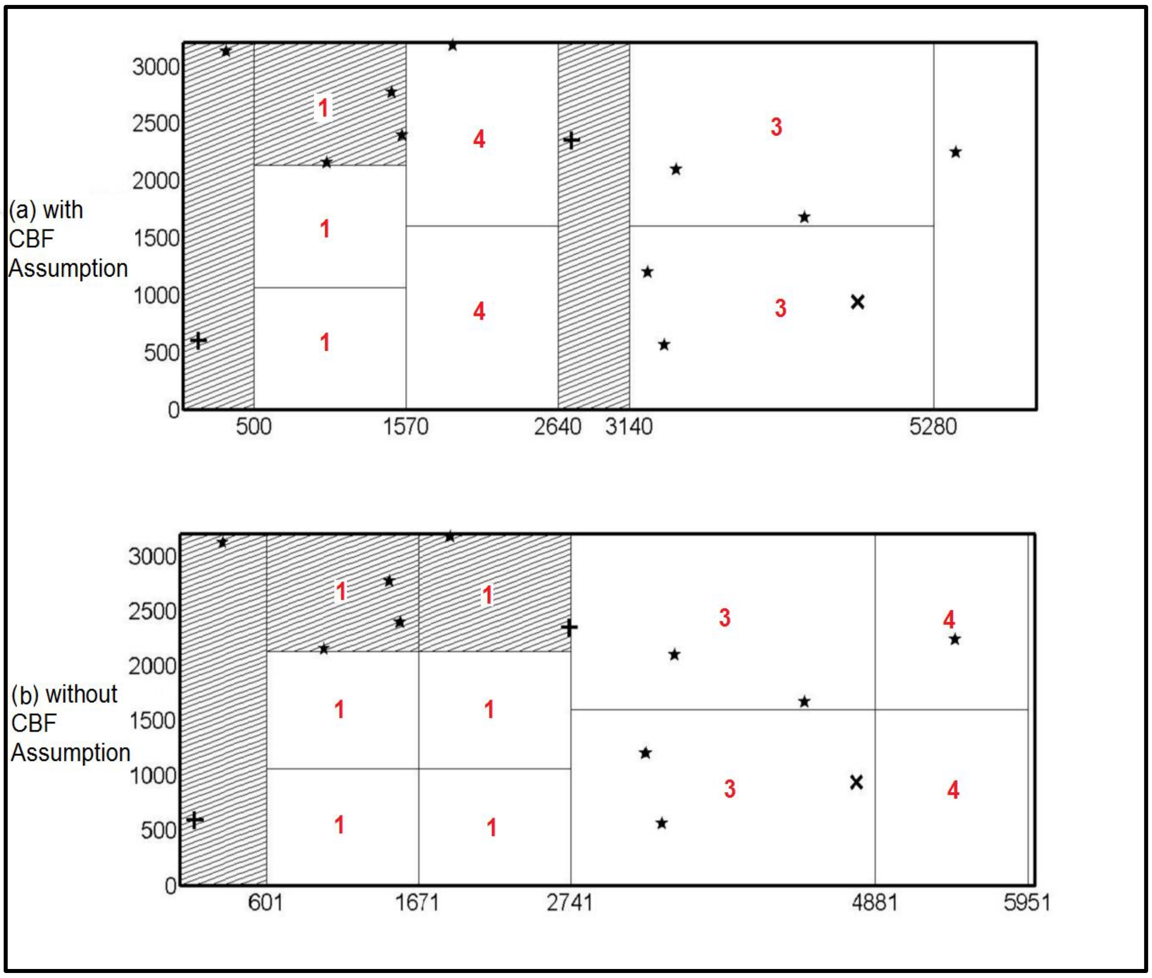


Figure 4.6. Solutions of the static cutting problem instance of 6,000mm with (a) and without (b) the CBF rule

Table 4.1. Defect tolerances of each product type in the examples shown in Figure 4.6 and Figure 4.7

		Products				
		1	**2**	**3**	**4**	**5**
Defect Types	*****	1	1	2	1	2
	x	1	1	1	1	1
	+	0	0	0	0	0
	.	0	0	0	0	0
	#	0	0	0	0	0

4.3.2. Suboptimality of the CBF Rule for the Classical Cutting Stock Problem

In this section, we provide an example for the classical cutting stock problem with defects and demonstrate that the CBF rule may lead to suboptimal decisions for this classical version as well. In this example, we solve a SCP on a glass of fixed length L, where any glass remaining at the end of the sheet is also considered as scrap. Two different solutions with and without the CBF rule are shown in Figure 4.7 (a) and (b) respectively. Defect and quality class descriptions are the same as in Subsection 4.3.1. There are three critical defect points two of grade 2 at $x =$3,796mm, $x =$3,902mm and one of grade 3 at $x =$5,333mm. Both solutions start with the same configuration. Then, in Figure 4.7 (a) the DP solution does not make a scrap cut because of the CBF rule which forces the method to cut only after defects. DP could make a scrap cut at $x =$2,105mm, however that does not result in placing the two type 2 defects in the same product and cutting another valuable configuration in the remaining part of the glass sheet. This causes DP to produce only three valuable products (one less than the alternative solution without CBF) and a wider glass strip at the end of the sheet. On the other hand, as shown in Figure 4.7 (b) we sacrifice a glass strip of length 702mm even though there is no high grade defect in the area. However this decision makes it possible to collect two type 2 defects in a single product and to cut a valuable configuration before the critical defect of type 3. Similar to the first example, sacrificing some glass to save more in the subsequent area of the glass sheet is the reason for the better solution quality produced without the CBF rule. In this problem, DP creates an area of 11,531,925mm^2 scrap glass while the second solution without the CBF rule creates 10,673,250mm^2 scrap glass.

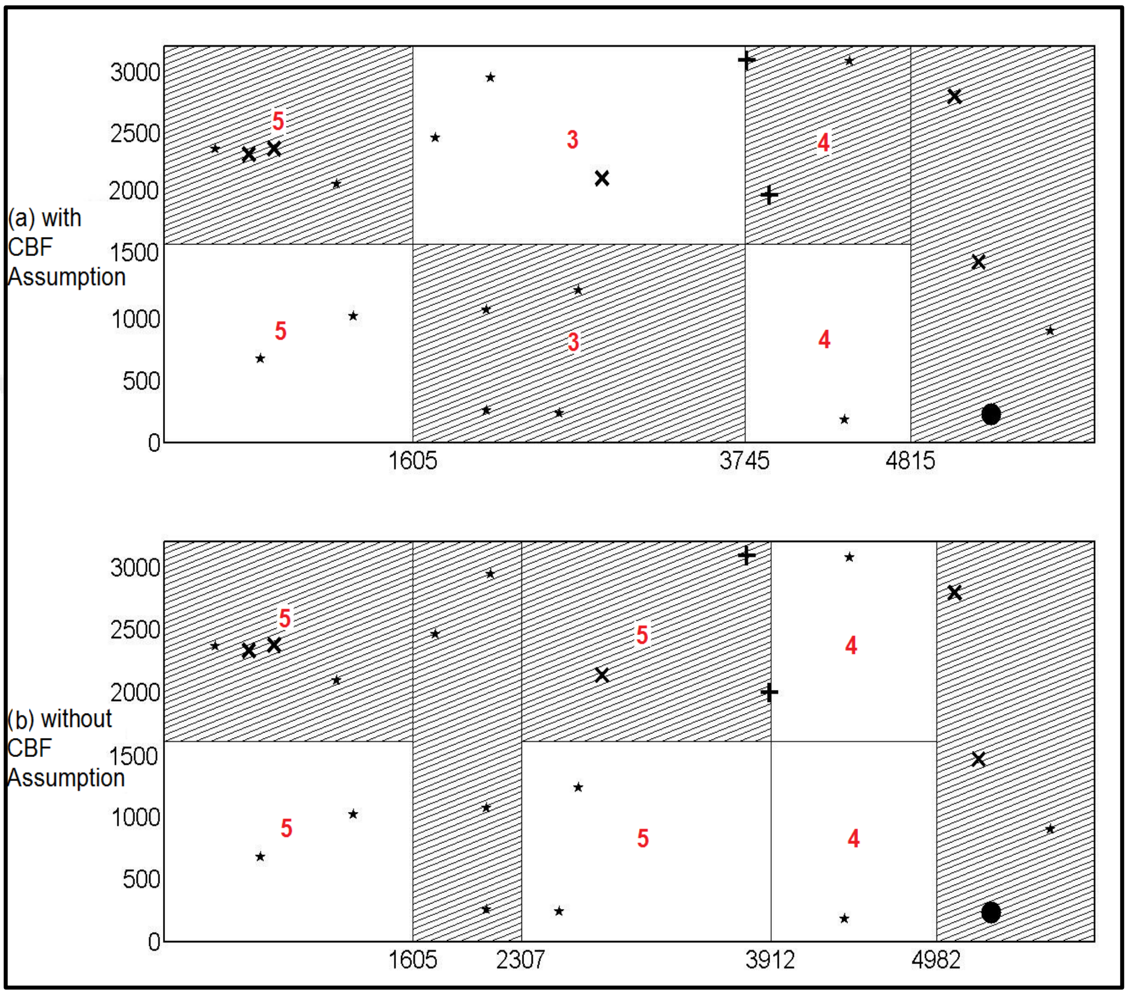


Figure 4.7. Solutions of the static cutting problem instance of 6,000mm with (a) and without (b) the CBF rule

4.4. SOLUTION OF STATIC GLASS CUTTING PROBLEM WITHOUT THE CBF RULE

In this version of the static cutting problem, we relax the CBF restriction and allow scrap cuts of arbitrary length between product cuts. Since the length of the scrap cut is now variable, we can no longer use the three approaches in Section 4.2 that represent the scrap cuts as dummy configurations of known length. Therefore the problem is no longer a sequencing problem, which makes it more difficult to represent and solve. In the following parts of this chapter, methods to solve this more general problem are explained in detail.

4.4.1. Mathematical Programming Model

The static glass cutting problem without the CBF rule can be formulated using a mixed integer programming (MIP) model as explained in this subsection. Before we go into the model details, one should note that we run a preprocessing algorithm prior to solving the MIP model, where the ordered set of configuration set (I^0) and the values of each configurations in this set (v_i) are created before the MIP model. In the MIP model below, the value of a configuration depends on its location on the glass sheet. The preprocessing algorithm finds all intervals where the value of a configuration remains constant and creates a copy of it for each interval. Therefore in the set I^0, there are many configuration copies all of which have a validity interval and a constant value. The set I^0 is a set of configurations ordered by their starting coordinate of validity intervals. Details of the preprocessing algorithm are explained in subsection 4.4.2.

$$Max \sum_{i\in I^0} v_i y_i + \mu \left(X_{end} - Start - \sum_{i\in I^0} l_i y_i\right) - \varepsilon \sum_{i\in I^0} x_i \tag{4.5}$$

s.t.

$$x_i \leq x_{max_i} y_i \quad , \quad \forall i \in I^0 \tag{4.6}$$

$$x_i \geq x_{min_i} y_i \quad , \quad \forall i \in I^0 \tag{4.7}$$

$$x_i + l_i y_i \leq x_j + L\left(1 - y_j\right) \quad , \quad \forall i,j \in I^0, i < j \tag{4.8}$$

$$x_j - x_i - l_i y_i \geq MinCut\, z_i - L\left(2 - y_i - y_j\right) \quad , \quad \forall i,j \in I^0, i < j \tag{4.9}$$

$$x_j - x_i - l_i\, y_i \leq L\, z_i + L\left(2 - y_i - y_j\right) \quad , \quad \forall i,j \in I^0, i < j \tag{4.10}$$

$$x_i - Start \geq Mincut\, w_i - L\,(1 - y_i) \quad , \quad \forall i \in I^0 \tag{4.11}$$

$$x_i - Start \leq L\, w_i + L\,(1 - y_i) \quad , \quad \forall i \in I^0 \tag{4.12}$$

$$x_i + l_i \leq End \quad , \quad \forall i \in I^0 \tag{4.13}$$

$$x_i \geq Start - L\,(1 - y_i) \quad , \quad \forall i \in I^0 \tag{4.14}$$

$$x_i + l_i - L(1 - y_i) \leq X_{end} \quad , \quad \forall i \in I^0 \tag{4.15}$$

$$End - Minl + 1 \le X_{end} \ , \ \forall i \in I^0 \tag{4.16}$$

$$x_i \ge 0 \ , \ \forall i \in I^0 \tag{4.17}$$

$$X_{end} \ge 0 \tag{4.18}$$

$$y_i, z_i, w_i \in \{0,1\} \ , \ i \in I^0 \tag{4.19}$$

where,

Sets

I^0: ordered set of configurations,$i \in I^0 : i = 1,2,\dots,N$

P : set of products, indexed by p

Parameters

v_i : value of configuration i, $i \in I^0$

μ : value of scrap glass per unit length (The width of the scrap glass is constant at W and the value of scrap glass is independent of its position on the glass sheet).

D_p: production target of product p

$pt_i(p)$: number of valuable products of type p configuration i

$xmin_i$: minimum possible starting coordinate of configuration i

$xmax_i$: maximum possible starting coordinate of configuration i

$Minl$: minimum length of available configurations

$Start$: starting coordinate of static problem

End : end coordinate of static problem

ε: a sufficiently small positive number

Variables

x_i : starting x-coordinate of configuration $i, i \in I^0$

X_{end} : end coordinate of last configuration

$y_i : \begin{cases} 1 & \text{if configuration } i \text{ is chosen for cutting} \\ 0 & \text{otherwise} \end{cases}$

$z_i : \begin{cases} 1 & \text{if there is a scrap cut after configuration } i \ (i \in I^o) \\ 0 & \text{otherwise} \end{cases}$

$w_i : \begin{cases} 1 & \text{if there is a scrap cut between Start and configuration } i \ (i \in I^o) \\ 0 & \text{otherwise} \end{cases}$

The objective function is given in Equation (4.5). First term in this equation denotes the total value of configurations. These v_i values are calculated by a preprocessing algorithm that is run prior to solving the MIP model. One should note that v_i values in MIP model are not the same with $\pi_i(x)$ values in DP. In the MIP model, v_i values are parameters and they are known in advance whereas $\pi_i(x)$ is a function of configuration i at coordinate x which is calculated by DP within the recursive function at required x values. The second term shows the total scrap area between configurations. A negative μ value should be chosen to penalize the scrap glass production. The last term in the objective function is added so that the model favours solutions that are closer to the starting coordinate $(Start)$ among alternative optimal solutions. Since the last part of the static length will be re-evaluated, such solutions are considered to be superior. Equations (4.6) and (4.7) ensure that all configurations are placed between their allowed minimum and maximum x coordinates $xmin_i$ and $xmax_i$. These coordinates are also calculated by the preprocessing algorithm. Equation (4.8) prevents placement of two configurations on the same area. This equation works only if configurations are ordered according to their x coordinates and there are no two configuration ranges that overlap. The preprocessing algorithm ensures that these two conditions are satisfied. According to (4.9) and (4.10), the configurations should be placed immediately after one another, leaving no space in between consecutive x-cuts or leaving a scrap cut of length of at least $MinCut$. Similarly, Equation (4.11) and (4.12) force the space between the starting point of the glass sheet and first configuration to be 0 or at least $MinCut$. Equations (4.13) and (4.14) ensure that all configurations are placed after the starting point of glass sheet and before the end point of the glass sheet for the current static problem. Equation (4.15) sets X_{end} as the end point of the last configuration and Equation (4.16) ensures that the length of remaining glass sheet after the last cut is less than $Minl$.

4.4.2. Preprocessing

Figure 4.8 provides the flowchart of the preprocessing algorithm. The objective of the preprocessing algorithm is to define an ordered set of configurations I^0, such that each $i \in I^0$ has a range of x coordinates that it can be placed and within this range the value of configuration i remains constant at v_i. The preprocessing algorithm creates the minimum cardinality set I^o such that no two configurations have overlapping ranges. Construction of I^o guarantees that the MIP model in Equations (4.5) – (4.19) works.

The preprocessing algorithm shown in Figure 4.8 has three main steps. At the first step, algorithm finds all intervals within which the value of a configuration remains constant. In order to find these intervals, it places a configuration at point $x = 0$ and starts to slide it along the glass sheet by increasing its starting coordinate x. Figure 4.9 explains a defect's impact on the value of a configuration depending its position on the glass sheet. Configuration have different values depending on its location on the glass sheet. These value changes result from the defects on the glass sheet. Let's say that the value of a configuration i is v at some point x. When we shift the configuration by a small distance ϵ along the glass sheet, the new value of configuration i is still v at point $x + \epsilon$, since it contains the same defects on its products (Figure 4.9 (a)). On the other hand, if we shift the configuration by a distance γ which is just enough to place a new defect within the configuration (or a current defect point outside the configuration), then the configuration will contain a different set of defect points (Figure 4.9 (b)). Thus, the value of configuration i may be different at $x + \gamma$. We call these threshold points "*leap points*" where the value of a configuration changes.

Between any two consecutive leap points, the value of the configuration is the same. The preprocessing algorithm finds all of these leap points for each configuration and defines the intervals between the leap points where the value does not change. It creates a dummy configuration for every interval by duplicating the original one; so we have many dummy configurations i with a fixed value v_i and with a defined valid $[min_i, max_i]$ interval.

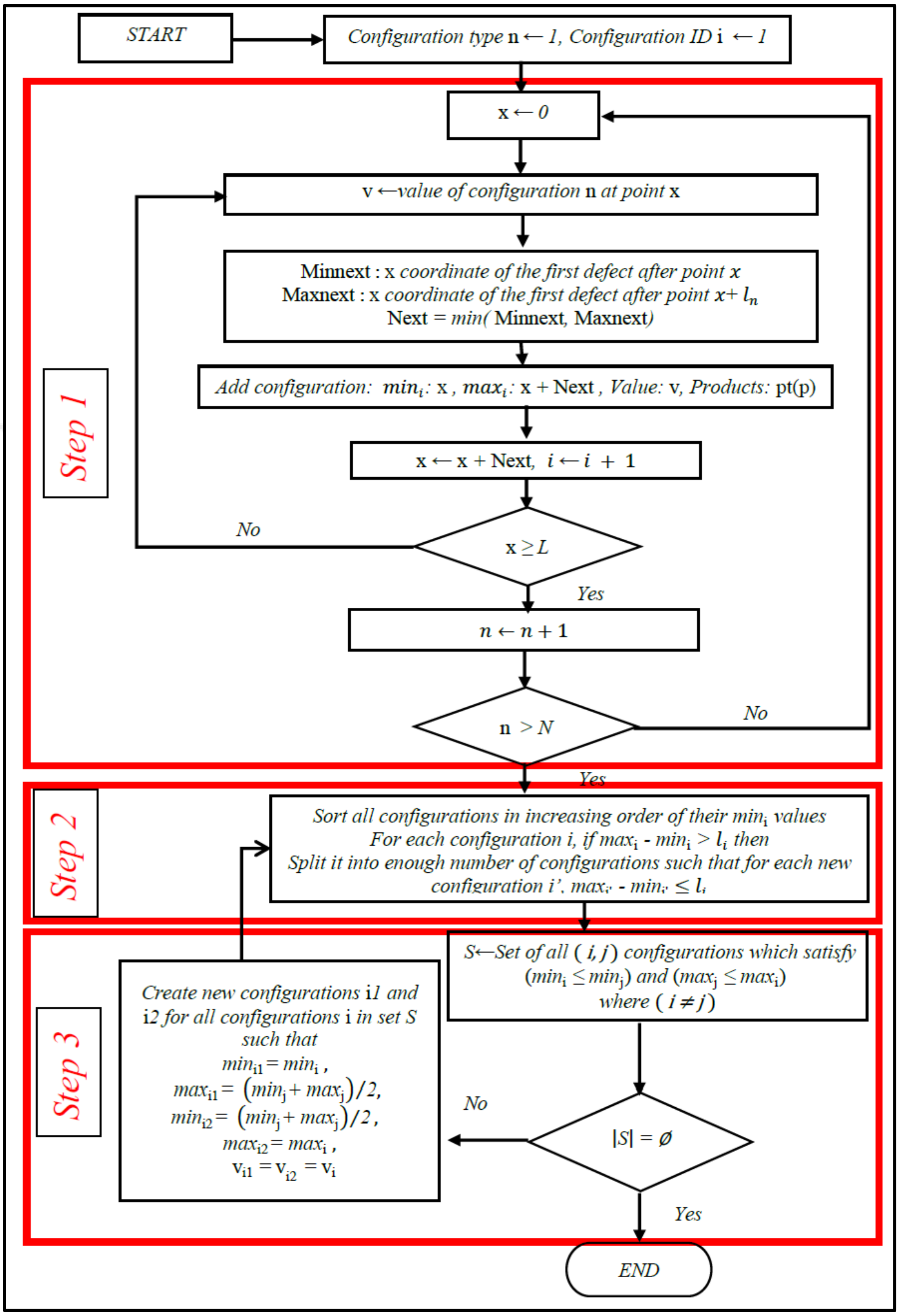

START
Configuration type n ← 1, Configuration ID i ← 1
x ← 0
v ← value of configuration n at point x
Minnext : x coordinate of the first defect after point x
Maxnext : x coordinate of the first defect after point x+ l_n
Next = min(Minnext, Maxnext)
Add configuration: min_i: x , max_i: x + Next , Value: v, Products: pt(p)
x ← x + Next, i ← i + 1
x ≥ L
No
Yes
n ← n + 1
n > N
No
Yes
Step 1
Step 2
Sort all configurations in increasing order of their min_i values
For each configuration i, if max_i - min_i > l_i then
Split it into enough number of configurations such that for each new configuration i', $max_{i'}$ - $min_{i'}$ ≤ l_i
Step 3
S←Set of all (i, j) configurations which satisfy (min_i ≤ min_j) and (max_j ≤ max_i) where (i ≠ j)
|S| = Ø
No
Yes
Create new configurations i1 and i2 for all configurations i in set S such that
$min_{i1} = min_i$,
$max_{i1} = (min_j + max_j)/2$,
$min_{i2} = (min_j + max_j)/2$,
$max_{i2} = max_i$,
$v_{i1} = v_{i2} = v_i$
END

Figure 4.8. Preprocessing algorithm

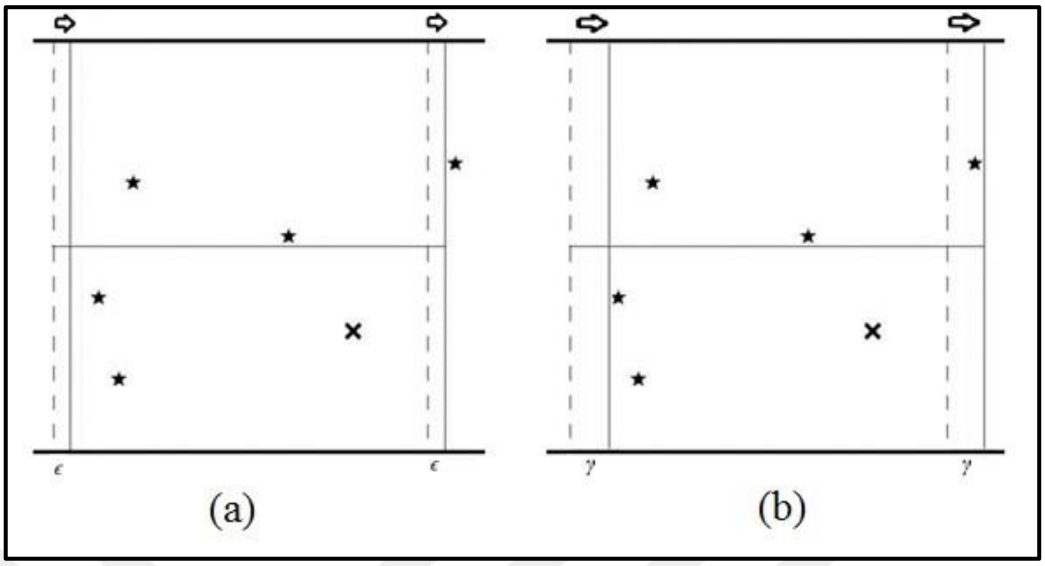


Figure 4.9. Leap point at $x + \gamma$ where value of the configuration changes due to the entrance of third defect into the upper product

The second step of the preprocessing algorithm in Figure 4.8 prevents unnecessary elimination of feasible solutions. Equations (4.6) and (4.7) of the MIP model ensure that all configurations are placed between their allowed minimum and maximum x coordinates $xmin_i$ and $xmax_i$. However, the model can place the configuration only once in its validity interval even if there is enough space to place more than one configuration. This causes the model to ignore some of the feasible placements. Therefore, if the length of the interval is larger than the configuration length l_i, then the preprocessing algorithm simply divides the interval into sub intervals and creates additional dummy configurations. The length of these sub intervals is limited to the length of the configuration, since it is not possible to place more than one configuration in this interval.

The third step of the preprocessing algorithm in Figure 4.8 eliminates the violation risk of Equation (4.8) in case of overlapping configurations.

$$x_i + l_i y_i \leq x_j + L\left(1 - y_j\right) \quad , \ \forall i, j \in I^0, i < j \tag{4.8}$$

Equation (4.8) is inactive when at least one of the configurations i and j is not used (i.e. $y_i = 0$ or $y_j = 0$). On the other hand, when both configurations i and j are selected ($y_i = y_j = 1$) then the model creates a feasible solution only if configurations i and j do not overlap. In this case Equation (4.8) becomes $x_i + l_i \leq x_j$.

In an ordered set I^0, there are three cases for valid intervals of configurations i and j where $i < j,\ i,\ j \in I^0$.

(a) $max_i < min_j$

In this case, it is obvious that in all feasible solutions that use i and j, Equation (4.8) prevents overlapping configurations for the following validity intervals of i and j as depicted below (Each rectangle represents the valid intervals of a configuration.):

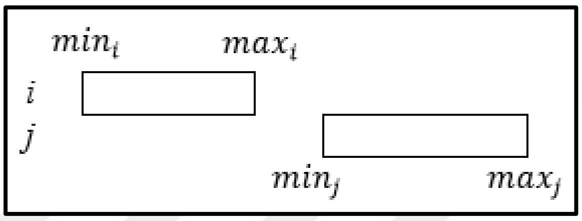


Figure 4.10. Validity intervals of configurations $\boldsymbol{i}$ and $\boldsymbol{j}$ in case (a)

(b) $min_j < max_i < max_j$

Since all validity intervals are limited by the length of their configurations in Step 2, if i is used, j cannot start before following shaded interval. Equation (4.8) forces j to begin from the second part of its validity interval, which prevents the two configurations from overlapping.

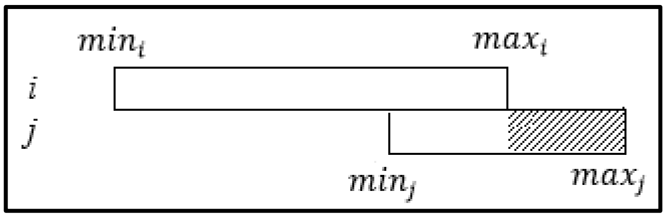


Figure 4.11. Validity intervals of configurations $\boldsymbol{i}$ and $\boldsymbol{j}$ in case (b)

(c) $min_i < min_j < max_j < max_i$

In this last case, each configuration can start from any point within its indicated shaded areas. This solution is feasible, however $x_i + l_i \geq x_j$, which is a violation of Equation

(4.8). Thus, in the mathematical model, this feasible case will be accepted as infeasible because of Equation (4.8).

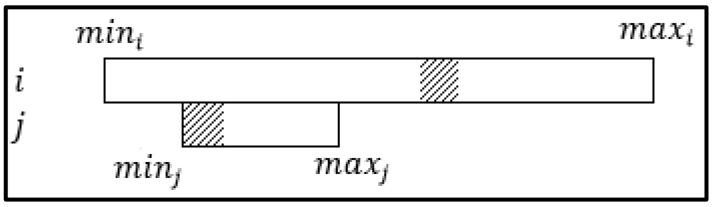


Figure 4.12. Validity intervals of configurations i and j in case (c)

As a result, Equation (4.8) ensures that there is no overlapping configurations but eliminates some of the feasible solutions in case of overlapping validity intervals. Since Equation (4.8) is crucial for the model, we transformed case (c) to the following one by dividing the validity interval of configuration i further into two new configurations $i'$and i''.

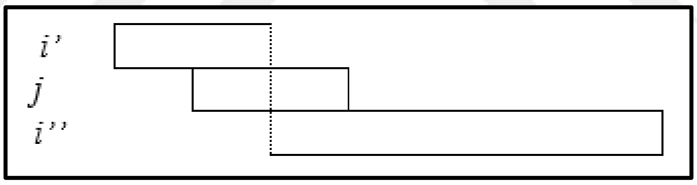


Figure 4.13. Transformation of case (c) in step 3 of preprocessing algorithm

With this transformation, we eliminate case (c) and create two new cases (b) (between $i'$and j, j and i'') and a new case (a) (between i' and i'') which can be handled by Equation (4.8). This step of creating dummy configurations is given in third step of Figure 4.8.

After these two steps of duplicating configurations, the mathematical model is ready to run with dummy configurations. The resulting ordered set of configurations I^0 can be used in the MIP model in Equations (4.5) – (4.19) to create feasible solutions.

4.4.3. Alternative Solution with a Limited Number of Configurations

The total number of configurations after the duplication described in the previous section is much larger than the original problem. Thus, the performance of the MIP model highly depends on the number of dummy configurations. In a static problem with 12m length and medium defect density, there can be more than 200 configurations. On the other hand, there are less than 10 configurations in the solution.

In many cases, allowed time is not enough for the MIP model to find an optimal solution and the best feasible solution found within the allowed time is executed. So, reducing the problem size (i.e. the number of configurations) may increase the solution quality or it enables the algorithm to find the optimal solution in a shorter time.

As a result, we decided to reduce the size of the configuration set I^0. In order to test this hypothesis we created a version of the algorithm where we use only first n configurations in the model. The result of this version is reported in Chapter 5.

4.4.4. Fall-Back Heuristic

As mentioned before, there is very limited time to solve the static problem. Frequently the model can reach optimal solutions; but sometimes we have to execute the best feasible solution since time is not enough to find the optimal one. Unfortunately, there is one another case. Rarely, the algorithm cannot find any solution neither optimal, nor feasible. There is no way to stop the production line, so we have to find a configuration as good as we can. For this purpose, we have developed another very simple heuristic. In this method, we check two different possibilities:

- All configurations that can be cut starting from the starting coordinate ($Start$)
- All configurations that can be cut starting from $MinCut$ (minimum cutting distance)

We can immediately cut a configuration from the beginning or if there are critical defects, we can first cut a portion of 500mm, then start from the beginning of the remaining strip. In the first case, the heuristic finds the values of all possible configurations. In the second case the heuristic calculates the total value corresponding to the scrap cut, plus the configuration placed after that cut. At the end, the best possible option is executed. In case of any tie, the configuration with the smallest x-coordinate is chosen to maximize the remaining strip for the following iterations.

4.5. INCORPORATION OF PRODUCTION TARGETS

In the current industrial applications, production targets are manually controlled by the personnel that oversee the glass production line and the product priorities are updated in real time throughout the day, taking into account the product targets and the actual production quantities. The glass cutting algorithm used in the production line determines the products to be cut using these priorities, thus achieving daily targets in each product type. In the original algorithm we have developed, the production targets are ignored and the algorithm chooses which products are cut solely based on their impact on the objective function (i.e. value maximization or scrap minimization). In this version of the online glass cutting algorithm, we made modifications to take into consideration the production targets during the execution of the online algorithm so that the algorithm can replace the manual control mechanism currently being used. In this version, there is no need to manually prioritize the cutting line, and problems arising from not intervening in time can be avoided. There are two ways to incorporate production targets in the algorithm:

- Impose constraints on the number of units produced of each product type
- Include a term in the objective function to balance the production according to these targets.

4.5.1. Production Targets as Constraints

In this version, we made modifications to all versions of the methods so that production targets are observed as cutting solutions are generated. In the GA, a gene corresponding to a product which has already reached its production target is evaluated as scrap glass. So, the GA does not choose that solution. Similarly in DA, when the production target of a product is reached, value function π_i accepts that product in configuration i as scrap.

In the MIP model, production targets are included using the following constraint set:

$$\sum_i pt_i(p)y_i \leq D_p \quad , \quad \forall p \in P \tag{4.20}$$

In this constraint, D_p is the upper bound on the production of product p (the difference between the production target of product p and the number of products of type p produced so far) and $pt_i(p)$ is the number of products of type p in configuration i. The summation on the left hand side calculates the total number of products of type p that will be cut in the current SCP. Thus, constraint (4.20) ensures that the model produces solutions within the given production targets. At the end of each iteration, production upper bounds of all products are updated based on the quantities of each product type produced in that SCP. We name this version of MIP model MIP+C which denotes the MIP model with production target constraints.

4.5.2. Production Targets in the Objective Function

Incorporating production targets only as constraints does not allow the algorithm to balance the production quantities of each product type throughout the production process. Early completion of some products during the production process reduces the product variety that the algorithm can select towards the end of production; thus the algorithm is forced to choose configurations with higher scrap glass content. In order to be able to achieve a more balanced production distribution, the difference between the production targets and the realized production ratios is added to the objective function with a certain weight. This difference is calculated as the sum of the absolute value of the difference between the production target of each product and the actual production amount. In the GA, the fitness function is updated in the same manner. At the beginning of each iteration of the online glass cutting algorithm with DP, the difference between the production targets and the realized production ratios are updated. Then, the numerical effect of producing one more product p on this difference is calculated as Δ_p for each product $p \in P$. Finally the updated product value π_i' function is calculated as in Equation (4.21) and used in DP instead of the original value π_i.

$$\pi_i' = \pi_i + \sum_{p \epsilon P} pt_i(p)\Delta_p \tag{4.21}$$

In the MIP model, production targets are now included with additional constraints and a modified objective function. The additional constraints are the equations (4.20) mentioned in Subsection 4.5.1. In the new objective function (4.22), f_p is the ratio of the current production of each product p to the total quantity of all the products produced so far and f_p^0 is the ratio of the production target of product p to the sum of the production targets of all products. This is why the minimization of $\left|f_p - f_p^0\right|$ value for all products is intended to produce in accordance with the production targets and this balanced production is also expected to result in reaching targets of all products around the same time.

$$Max\ [\textstyle\sum_{i \in I^0} v_i y_i] - W\ [X_{end} - Start - \textstyle\sum_{i \in I^0} l_i\, y_i] - \alpha\ \textstyle\sum_{p \in P} \left|f_p - f_p^0\right| \qquad (4.22)$$

The objective function has a nonlinear structure due to the added absolute value term. To linearize this function, the constraint $f_p - f_p^0 = f_p^+ - f_p^-$ is added by defining two variables for each product type p, f_p^+ and f_p^- (f_p^+, $f_p^- \geq 0$). The last term in the objective function is changed then to ($-\ \alpha\ \sum_{p \in P}\left(f_p^+ + f_p^-\right)$). The linearized version of the objective function (4.22) is given in Equations (4.23) – (4.25) below.

$$Max\ [\textstyle\sum_{i \in I^0} v_i y_i] - W[X_{end} - Start - \textstyle\sum_{i \in I^0} l_i y_i] - \alpha\ \textstyle\sum_{p \in P}\left(f_p^+ + f_p^-\right) \qquad (4.23)$$

$$f_p^{+} \geq f_p\text{-}f_p^{0}\ and\ f_p \geq f_p^0 - f_p \qquad (4.24)$$

$$f_p^{+}, f_p \geq 0 \qquad (4.25)$$

We name this version of the MIP model with Equations (4.6) – (4.19) and (4.23) – (4.25) MIP+CO. CO stands for updating both the constraints and the objective function according to production targets. The last term in the objective function reduces the weight of first two terms which focus on maximizing production value and minimizing scrap glass. This may seem as a contradiction when we consider only short term results. However, our motivation for this term in objective function is to achieve the best total objective function value (i.e. sum of the first two terms) for the whole production process. Figure 4.14 shows an example problem where scrap minimization was the main objective. In this figure, we see the scrap glass area produced by the MIP versions with and without the production balance term in the objective function, MIP+CO and MIP+C respectively. One can see that MIP+CO produces more scrap glass in order to balance the production distribution. On the other hand, the time when a product reaches its production target can be delayed by this method. The algorithm with MIP+C completes production of one product at iteration 412. After that point, MIP+C starts to produce more scrap glass since it has less product options to choose from. However MIP+CO reaches its first production target at iteration 606. Similar to MIP+C, it starts to produce more scrap glass after that point; but since it has a better performance between iterations 412 and 606, its total scrap glass area is less at the end.

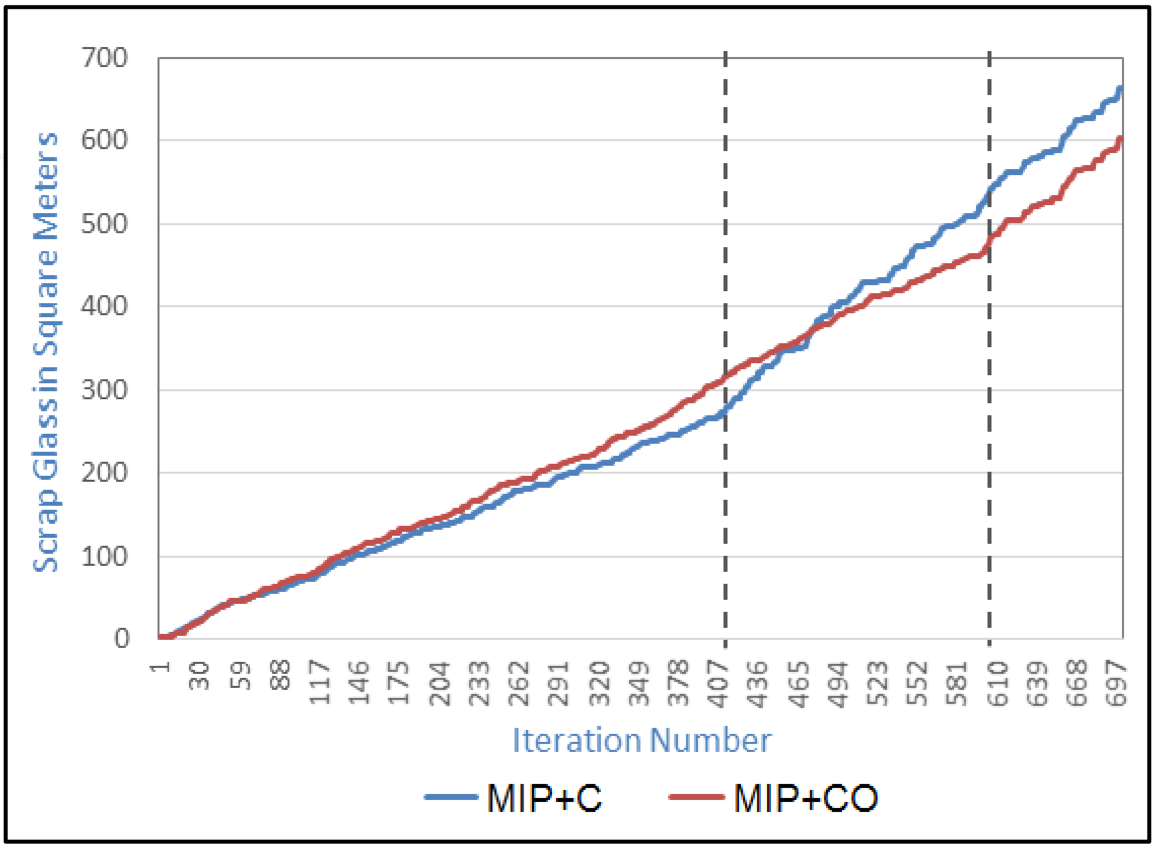


Figure 4.14. Scrap glass produced by the algorithm versions MIP+CO and MIP+C (with and without production balance)

4.5.3. Adaptive Algorithm

Clearly, the performance of the algorithm in Subsection 4.5.2 depends largely on the weight placed on production balance (α). As we experimented with different α values, we observed that the best value for α changes with the defect density and values of the production targets. It is a fact that both defect density and production targets will vary depending on the industry and the production conditions. For this reason, it is important that the algorithm can adapt itself according to the changing conditions of the production line by updating α value in real time. However, in the production environment, determining a suitable value for α and adjusting this value manually would require manual intervention by an operator who understands the how the algorithm works. The adaptive algorithm eliminates this human intervention and ensures that the value of α is adjusted objectively based on performance.

Unfortunately, it is very difficult to determine the best value of α that will yield the optimum long-term performance since introducing a high value of α causes the short-term performance to suffer. Therefore, we need a way to anticipate the long term impact of a particular α value in real time. To achieve this, we performed test runs with different values of α in parallel with the actual production run. By comparing the results of the actual and test runs, we decide which α value is better in the long term.

As we inspect Figure 4.14, we see that there are three different values we need to know for both α values (current and test) so that we can estimate total scrap at the end of the production process. These are the critical point where a product reaches its production target and scrap glass produced per unit length before and after that critical point. For instance, in Figure 4.14, the algorithm with MIP+CO completes the production of one product at iteration 606. The end coordinate of the last product of this type on the glass sheet is our critical point for the current α. Similarly, we name this product type as the critical product. One can see that the slope (total scrap per length) is not changing significantly during the iterations before the critical point. Although the critical point leads to a change of this slope, we can see that the slope after this point also does not change much within its range. Therefore, we can assume two different slopes before and after critical point. If we estimate these three values, we can predict the total scrap glass at the end of production run with the following formula:

$$S_\alpha\, CP + S_\alpha{}'(TL - CP) \tag{4.26}$$

where TL is the total length of glass sheet, CP is the critical point, S_α and S'_α are the slopes before and after the critical point respectively.

In Figure 4.15, we described the algorithm to estimate CP, S_α, S'_α and update the value of α during the production run. The algorithm decides the α value by comparing the estimated results of the current α value with the estimated results of the test α (α_test) value in each $4k$ iterations, where k > 1. Among these $4k$ iterations, $2k$ are performed to execute the production run with the current α value, whereas two sets of k iterations are run using a new α value (α_test) to estimate the values of S_{α_test} and S'_{α_test}. In a real production environment, $2k$ test runs and $2k$ production runs must be done concurrently on identical computers to prevent delays. In our experiments, we simulated the runs on a single computer to make sure that two runs were on identical computers and performed the runs in sequence. Our adaptive model is given in Figure 4.15. The algorithm is initiated using the same value for α and α_test. After $4k$ static solutions, algorithm begins to update α value.

In the first $2k$ iterations of each cycle (Figure 4.15, Step 1), the adaptive algorithm runs regularly with the current α and calculates the amount of scrap glass produced. At the end of the $2k$ iterations, we obtain the total area of scrap glass produced with the current α value. Algorithm estimates S_{α}, by dividing total scrap glass by total length used. Algorithm also keeps the production quantities of each product and calculates number of products of each type produced per unit length. Thus, it is easy to estimate the coordinates where each product type reaches its production target, and the minimum among these coordinates gives us the CP for the current α. Each current α value has already been tested as α_test before it was assigned as the current α. For this reason, the algorithm has already calculated and logged the results of the current α value when one of the product types reaches its production target; so it estimates S'_{α} from these logs. After the algorithm estimates all necessary values for the current α (S_{α}, S'_{α} and CP), it creates a new α_test value for the test runs. If the previous update of α is an update in the direction of increase, α_test becomes $\alpha + \varepsilon$, otherwise $\alpha - \varepsilon$ where ε denotes the step size. This information is recorded in a variable named $direction$. In order to test α_test, the algorithm is rerun on the same area of the glass sheet using with α_test.

In the third set of k iterations (Figure 4.15, Step 2), the algorithm calculates the total length of glass used and the total area of scrap glass produced to estimate S_{α_test}. Similarly algorithm keeps the production amounts of each production type and estimates CP_test. However, it cannot estimate S'_{α_test} since α_test may be a new value. Then, there is no information of α_test gathered after the critical point CP_test is reached. So, as shown in

Step 3 of Figure 4.15, the algorithm runs another set ok k iterations using α_test and setting the production target of the critical product to zero $(T_c = 0)$. Then it can estimate S'_{α_test} and also keeps the results to be used in the next cycles.

At the end of the $4k$ iterations, the algorithm has estimated the values of S_α, S'_α, CP, S_{α_test}, S'_{α_test} and CP_test. It then calculates the estimates of the total scrap glass to be produced using α and α_test, TSG_α and TSG_{α_test} respectively by using Equation (4.26) as shown in Step 4 of Figure 4.15. If $TSG_{\alpha\_test} < TSG_\alpha$ then, α_test will replace α. Otherwise α does not change but $direction$ is updated in order to test a better α_test value around the current α in the next cycle.

Parameter k can be assigned according to the conditions of production environment. We know that the best α value depends on the defect density. Although one can classify a problem instance by looking at its average defect density, the density is not the same everywhere on the glass sheet. In our experiments, we noticed that this nonhomogeneous distribution of defects causes wrong α decisions frequently when a small k value is used. Therefore, we recommend not to use very small values to avoid instability of α values. On the other hand, large k values reduce the update rate. As depicted in Figure 4.15, α is updated only once in every $4k$ iterations. Therefore the choice of k value is an important decision that affects the performance of the adaptive algorithm. We discuss how the value of k is selected for our adaptive model in Subsection 5.3.1.

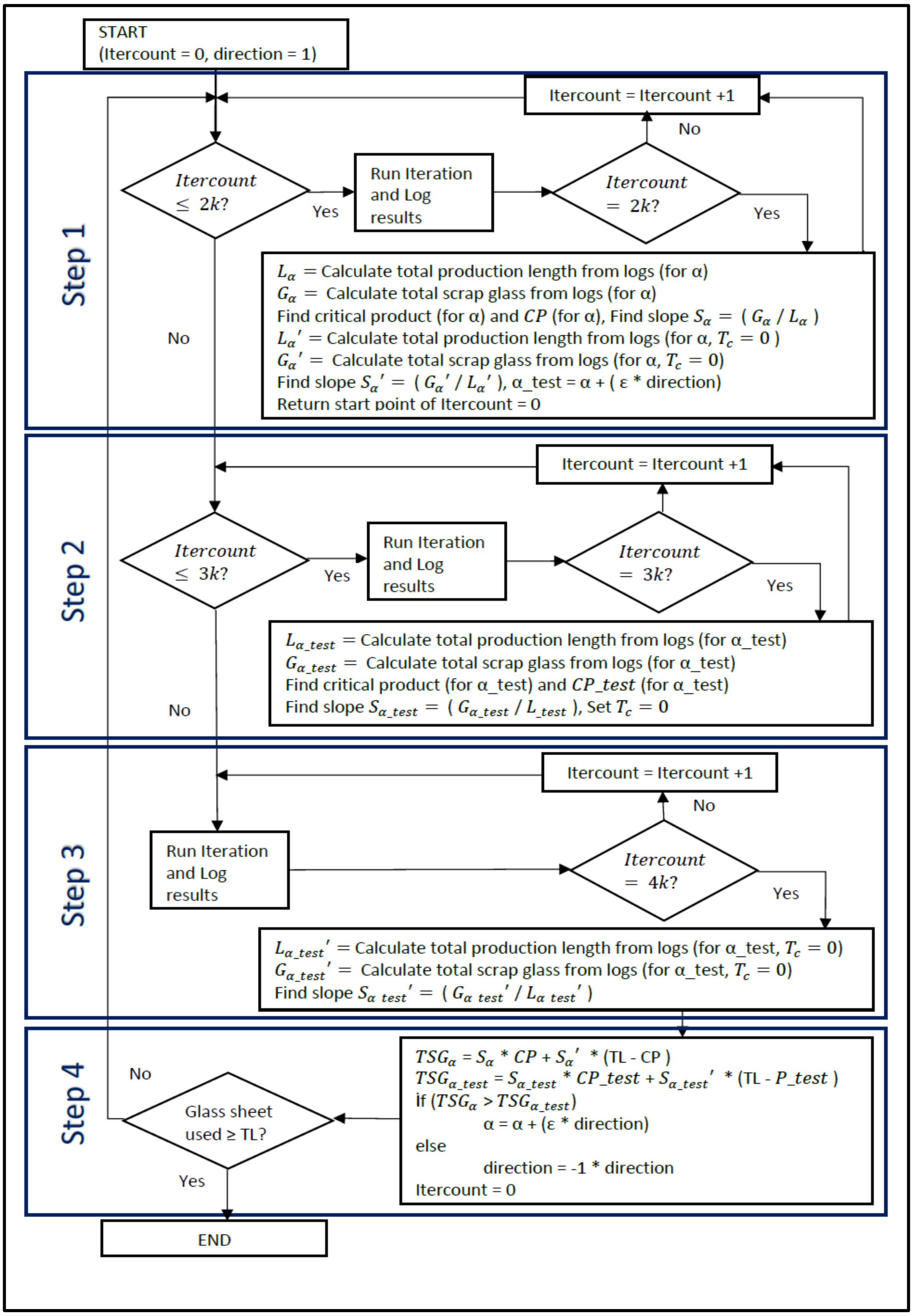

START
(Itercount = 0, direction = 1)
Step 1
Itercount = Itercount +1
No
$Itercount \leq 2k$?
Yes
Run Iteration and Log results
$Itercount = 2k$?
Yes
No
L_α = Calculate total production length from logs (for α)
G_α = Calculate total scrap glass from logs (for α)
Find critical product (for α) and CP (for α), Find slope $S_\alpha = (G_\alpha / L_\alpha)$
L_α' = Calculate total production length from logs (for α, $T_c = 0$)
G_α' = Calculate total scrap glass from logs (for α, $T_c = 0$)
Find slope $S_\alpha' = (G_\alpha' / L_\alpha')$, α_test = α + (ε * direction)
Return start point of Itercount = 0
Step 2
Itercount = Itercount +1
$Itercount \leq 3k$?
Yes
Run Iteration and Log results
$Itercount = 3k$?
Yes
No
L_{α_test} = Calculate total production length from logs (for α_test)
G_{α_test} = Calculate total scrap glass from logs (for α_test)
Find critical product (for α_test) and CP_test (for α_test)
Find slope $S_{\alpha\_test} = (G_{\alpha\_test} / L_{\_test})$, Set $T_c = 0$
Step 3
Itercount = Itercount +1
No
Run Iteration and Log results
$Itercount = 4k$?
Yes
L_{α_test}' = Calculate total production length from logs (for α_test, $T_c = 0$)
G_{α_test}' = Calculate total scrap glass from logs (for α_test, $T_c = 0$)
Find slope $S_{\alpha\ test}' = (G_{\alpha\ test}' / L_{\alpha\ test}')$
Step 4
No
Glass sheet used ≥ TL?
Yes
$TSG_\alpha = S_\alpha * CP + S_\alpha' * (TL - CP)$
$TSG_{\alpha\_test} = S_{\alpha\_test} * CP\_test + S_{\alpha\_test}' * (TL - P\_test)$
if ($TSG_\alpha > TSG_{\alpha\_test}$)
α = α + (ε * direction)
else
direction = -1 * direction
Itercount = 0
END

Figure 4.15. Adaptive algorithm

5. COMPUTATIONAL RESULTS

Computational results are collected under three main sections. Section 5.1 covers the results of GA, DP and MIP model without any production targets which is described in Section 4.4. Section 5.2 presents the behaviour of the methods under production targets. Different versions of solution methods with production targets mentioned in Subsections 4.5.1 and 4.5.2 are addressed in this section. Section 5.3 is dedicated to the adaptive model which has the most favourable results among all solution methods. Finally the reader can find an overview of all computational results in Section 5.4.

5.1. COMPUTATIONAL RESULTS OF THE SOLUTION METHODS WITHOUT PRODUCTION TARGETS

In this version of the problem, production targets are unlimited. In the original online glass cutting algorithm, three methods are used to solve the underlying SCP without production targets. GA and DP are two solution methods which use the CBF rule. On the other hand, MIP model was used to obtain the results for the same problem instances without the CBF rule.

5.1.1. Experimental Design

Experiments are performed by using 3 different SCP lengths (6m, 9m, 12m), 3 different defect densities (low – 0.55 defect/m^2, medium – 0.83 defect/m^2, high – 1.1 defect/m^2). These densities are representative of contemporary float lines that are currently in operation. 5 problem instances are created randomly for each case. Each problem instance is a continuous problem on a glass sheet of 1,200m length and 3.21m width.

For DP, 3 seconds are enough to solve the SCP of length L = 6m and 9m. Thus, we do not use any time limit for DP. In fact, since DP does not produce a result until it is completed, we do not have an option of setting a time limit, we can only measure the elapsed time. We also allowed GA 3 seconds to solve each SCP. On the other hand, MIP model solves a more complicated version of the problem, so time limit is more important. We run it with unlimited time, with 15 seconds and 3 seconds to compare the performance of the algorithm as a function of the CPU time.

In order to set the parameters for the GA, we run some preliminary experiments. Based on these experiments, we set the population size to 20 and choose to use an elitist selection with and elite count of 2.

Experiments on EE are designed to see how much our methods improve the solution quality against the explicit enumeration approach currently adopted in the glass industry. EE is run only for $L = 9$m with $c = 3$ and $c' = 1$, since the current focus is to enumerate 3 configurations and implement the first one no matter what the length L is.

In all methods, the minimum cutting distance $MinCut$ is set as 500mm. The values of all products per m^2 are assumed to be the same in all cases. Under this assumption, the objective function can be reduced to minimizing the total scrap area. Therefore we focused on this objective in the evaluation of computational results. This set of experiments aims to compare solution qualities of the proposed methods. As explained in Subsection 4.4.3, we have observed that the solution quality of MIP model can be improved by limiting the number of configurations in the model. Effects of configuration limit on the MIP performance are also tested in this section. Various configuration limits between 10 and 140 are applied to the same problem in order to observe the impact on the results.

5.1.2. Solution Quality

A comparison of the results produced by two different solution methods that employ the CBF rule (GA and DP) is shown in Table 5.1. Moreover, we include the test results of EE on the experiment set of 9m in order to compare the solution qualities of our methods with the current practice in the glass industry. This table shows the amount of scrap glass in square meters. The rows of the table show the sample problems grouped by their defect density, and the columns show the SCP lengths L used in each method. We observe that solution quality of both methods, GA and DP, is better than the solution quality of EE, which is a common approach in glass cutting industry. We obtain an improvement of 25 per cent by DP, and 18 per cent by GA on the average over the solution quality of EE. The results of the DP method using $L = 12$m is included so that improvement in solution quality can be observed even though the DP method does not always terminate in 3 seconds. In GA, we can make sure that SCPs are solved within 3 seconds because we can technically set the time limit. The Table 5.1 shows that the DP produces better results as the SCP length increases. The reason for this is that the DP method has defect information of longer distances. On the contrary, we see that the results of GA deteriorate as the SCP length increases. This is due to the attempt to solve a more complex problem within the same time limit. When we compare the two methods, we observe that the GA produces better results only in the case of low defect density problems using short SCP lengths. In all other cases, we see that DP produces better results. Within the defined time limit of 3 seconds, the best results are produced by the DP method using SCP length of 9m. Due to this overall, superiority of DP, we used this method to compare the performance of the methods that use the CBF rule and those that do not use the CBF rule. Table 5.2 summarizes these results.

Table 5.1. Comparison of EE, DP and GA methods in terms of solution quality (the amount of scrap glass in m^2)

Defect Density	**Problem Instance**	**EE**	**DP**			**GA**		
		9m	**6m**	**9m**	**12m**	**6m**	**9m**	**12m**
Low	**1**	549.69	384.10	366.16	343.17	369.50	377.23	411,29
	2	588.95	415.87	404.71	385.24	397.26	404.71	445,35
	3	589.81	419.02	411.29	398.70	421.02	424.74	424,74
	4	534.71	388.11	370.65	350.90	374.65	394.69	394,40
	5	587.40	423.31	415.58	392.68	407.28	407.28	453,36
Average (Low)		**570.11**	**406.08**	**393.68**	**374.14**	**393.94**	**401.73**	**425.83**
Medium	**6**	1044.00	775.08	765.92	742.74	811.15	834.04	891,86
	7	1004.10	741.02	720.70	705.53	789.96	801.41	801,41
	8	1016.40	791.40	773.65	748.75	810.00	838.34	868,96
	9	1010.80	754.19	728.14	705.53	791.11	790.54	865,24
	10	976.83	753.33	744.17	714.97	789.68	817.16	862,38
Average (Medium)		**1010.43**	**763.00**	**746.52**	**723.50**	**798.38**	**816.30**	**857.97**
High	**11**	1503.80	1195.50	1180.40	1156.00	1320.30	1312.30	1357,50
	12	1481.20	1175.80	1165.80	1138.30	1277.10	1275.40	1334,90
	13	1477.50	1164.30	1144.60	1123.70	1299.70	1303.20	1357,50
	14	1485.20	1137.20	1133.10	1110.30	1247.60	1283.10	1318,60
	15	1451.00	1191.00	1161.80	1138.90	1294.90	1303.50	1358,70
Average (High)		**1479.74**	**1172.76**	**1157.14**	**1133.44**	**1287.92**	**1295.50**	**1345.44**
Average (Overall)		**1020.09**	**780.62**	**765.78**	**743.69**	**826.75**	**837.84**	**876.41**

Table 5.2. Comparison of DP and MIP in terms of solution quality (the amount of scrap glass in m^2)

Defect Density	**Problem Instance**	**DP**		**MIP (unlimited time)**		**MIP (3 sec)**		**MIP (15 sec)**	
		6m	**9m**	**6m**	**9m**	**6m**	**9m**	**6m**	**9m**
Low	**1**	384.10	366.16	360.34	339.73	363.78	481.41	357.77	364.35
	2	415.87	404.71	388.39	382.67	405.28	500.59	390.11	398.98
	3	419.02	411.29	392.68	391.83	415.30	506.60	391.54	404.71
	4	388.11	370.65	363.20	348.32	371.79	492.86	360.34	363.49
	5	423.31	415.58	408.14	397.84	421.02	508.32	409.86	404.13
Average (Low)		**406.08**	**393.68**	**382.55**	**372.08**	**395.43**	**497.96**	**381.92**	**387.13**
Medium	**6**	775.08	765.92	764.49	728.43	767.93	835.76	761.91	745.31
	7	741.02	720.70	711.54	688.07	729.29	833.76	712.11	721.56
	8	791.40	773.65	755.33	724.71	767.93	870.39	754.76	739.59
	9	754.19	728.14	723.56	690.93	737.30	844.63	721.27	701.52
	10	753.33	744.17	731.00	717.84	739.59	856.66	727.00	718.69
Average (Medium)		**763.00**	**746.52**	**737.18**	**710.00**	**748.41**	**848.24**	**735.41**	**725.33**
High	**11**	1195.50	1180.40	1164.60	1141.70	1176.40	1226.50	1163.80	1152.00
	12	1175.80	1165.80	1157.50	1135.20	1177.20	1217.60	1166.90	1127.10
	13	1164.30	1144.60	1135.70	1098.50	1149.50	1193.80	1137.70	1110.00
	14	1137.20	1133.10	1124.80	1089.40	1135.70	1169.50	1122.80	1103.40
	15	1191.00	1161.80	1163.20	1136.60	1171.80	1212.10	1164.30	1147.20
Average (High)		**1172.76**	**1157.14**	**1149.16**	**1120.28**	**1162.12**	**1203.90**	**1151.10**	**1127.94**
Average (Overall)		**780.62**	**765.78**	**756.30**	**734.12**	**768.65**	**850.03**	**756.14**	**746.80**

In Table 5.2, each row shows a sample problem grouped by defect density, and each column shows the SCP lengths used in two methods. The results of the problems display the amount of scrap glass produced (in m^2) by the two methods using the SCP lengths (L) indicated. The MIP model was run with 3 different time limits. The CPU time allowed in the first set of runs is considered infinite. We explored this setting to obtain the best results that the MIP model can produce. Thus, the effects of the time limits on the model can be seen more clearly. The second set of runs were completed under a 3 second time limit based on the restriction in the glass industry. A 15 second limit was used in the third set of runs. The reason for these last set of runs is the difference in CPU speeds between the computers we use for our study and the computers used in the industry. It is possible to produce a solution much faster with the computer systems used in the industry than the solutions produced in a personal computer. Therefore it is important to observe whether the solution quality can be improved when a more powerful computer is used. This improvement is quantified in the results under a 15 second time limit.

In the MIP results using a SCP length of $L = 6$m, there is no significant difference between the solution quality with unlimited time and under a 15 second time limit. However, limiting the CPU time to 3 seconds caused a slight deterioration in solution quality. In all instances with $L = 6$m, the results produced with MIP model are better than those produced with DP.

For solutions using $L = 9$m SCP length, the effect of time limit is much higher due to the larger size of the SCP. Since 15 seconds is still sufficient for solving many of those larger SCPs, the results produced are close to unlimited time solutions. However, reducing the time limit to 3 seconds has a significant impact on the results. Although MIP yields better results, under no time limit and with a limit of 15 seconds, MIP results with a 3 second limit are much worse than DP results for this SCP length. The reason is that the MIP model cannot achieve an optimal result for many of the SCPs, and it even has to resort using the fall back heuristic for some of the SCPs since no feasible solution could be obtained.

Table 5.3. Percentages of optimal, feasible solutions and solutions created by the fall back heuristic used by the MIP model in runs with SCP length L=9m under 3 second time limit

Defect density	**Problem instance**	**Optimal (%)**	**Feasible (%)**	**Heuristic (%)**
Low	**1**	2.3	96.7	1.1
	2	1.1	98.2	0.7
	3	1.7	97.3	0.9
	4	8.0	92.0	0.0
	5	3.1	88.4	8.5
Average (low)		**3.2**	**95.8**	**1.0**
Medium	**6**	9.1	89.2	1.7
	7	12.0	87.2	0.8
	8	11.3	87.7	1.0
	9	10.9	89.0	0.1
	10	15.1	83.7	1.2
Average (medium)		**11.7**	**87.4**	**1.0**
High	**11**	47.4	52.0	0.6
	12	40.3	59.7	0.0
	13	53.6	45.7	0.7
	14	43.8	55.7	0.4
	15	48.6	51.1	0.3
Average (high)		**46.7**	**52.8**	**0.4**
Overall average		**20.5**	**78.7**	**0.8**

Table 5.3 shows the usage frequency of three possible outcomes of the MIP model with SCP length $L = 9$m under a 3 second time limit. Even if the model rarely uses fall back heuristic, one can see that feasible solutions dominates the others. Therefore, solution quality of the MIP model is affected negatively by low optimal solution percentage.

As mentioned in Subsection 4.4.3, we also develop an alternative version of the method where the number of configurations in the MIP model is restricted to reduce the problem size. The impact of this limitation are depicted in Figure 5.1, 5.2 and 5.3 for $L = 6$, 9 and 12m, respectively. The graph in the upper part of each figure, the purple line shows how the scrap glass changes as the configuration limit increases. The blue line indicates the result of the MIP model without any configuration limit. These two lines depict the difference in solution quality between two methods for various configuration limits. Since less scrap glass is desired, a low value means better solution quality in these graphs. For small SCP lengths such as 6m, the solution quality is almost the same as the unlimited version when the configuration limit is higher than 60. When the SCP length is 9m, there is a configuration limit range (50-100) where the alternative solution method gives better results. For larger SCP lengths such as 12m, the alternative solution method is better for a larger configuration limit range (20-140). It is also obvious that the solution quality is significantly improved (up to 20 per cent) when SCP length is high ($L = 12$m). The third line of the graphs in the upper part shows average run time per SCP for the version with configuration limit. For SCPs with $L = 9$m, the alternative solution method gives the same result with the unlimited one in 1.31 seconds which is 57 per cent faster than the version without configuration limit. For $L = 12$m, 0.61 second is sufficient to reach the same solution quality, which is an 80 per cent improvement. Overall, we see that the alternative solution is able to provide better solutions within the same time or the same solution in less time.

In the lower parts of Figure 5.1 – 5.3, the line with label “Optimal” indicates the percentage of SCPs for which the MIP model reaches the optimal solution. The label “Feasible but not Optimal” line shows the percentage of feasible but not optimal solutions. Finally the line labelled “Fall-back Heuristic” shows the percentage of solutions obtained using the fall back heuristic. Since there is no other possibility, the sum of the percentages of these three solution options is 100 per cent for all configuration limits. With low configuration limits, the model can find the optimal solutions for the reduced problem easily; but the solution quality is low. It is because the low configuration causes most of the available data to be ignored. With high configuration limits, even if there is sufficient data, solution quality is still low. On the other hand, a high configuration limit makes the problem difficult and the model cannot find the optimal solution in most of the cases. Consequently, using feasible solutions and the fall-back heuristic instead of optimal solutions reduce the solution quality. That is why only a certain range of configuration limit is capable of significant improvement in solution quality.

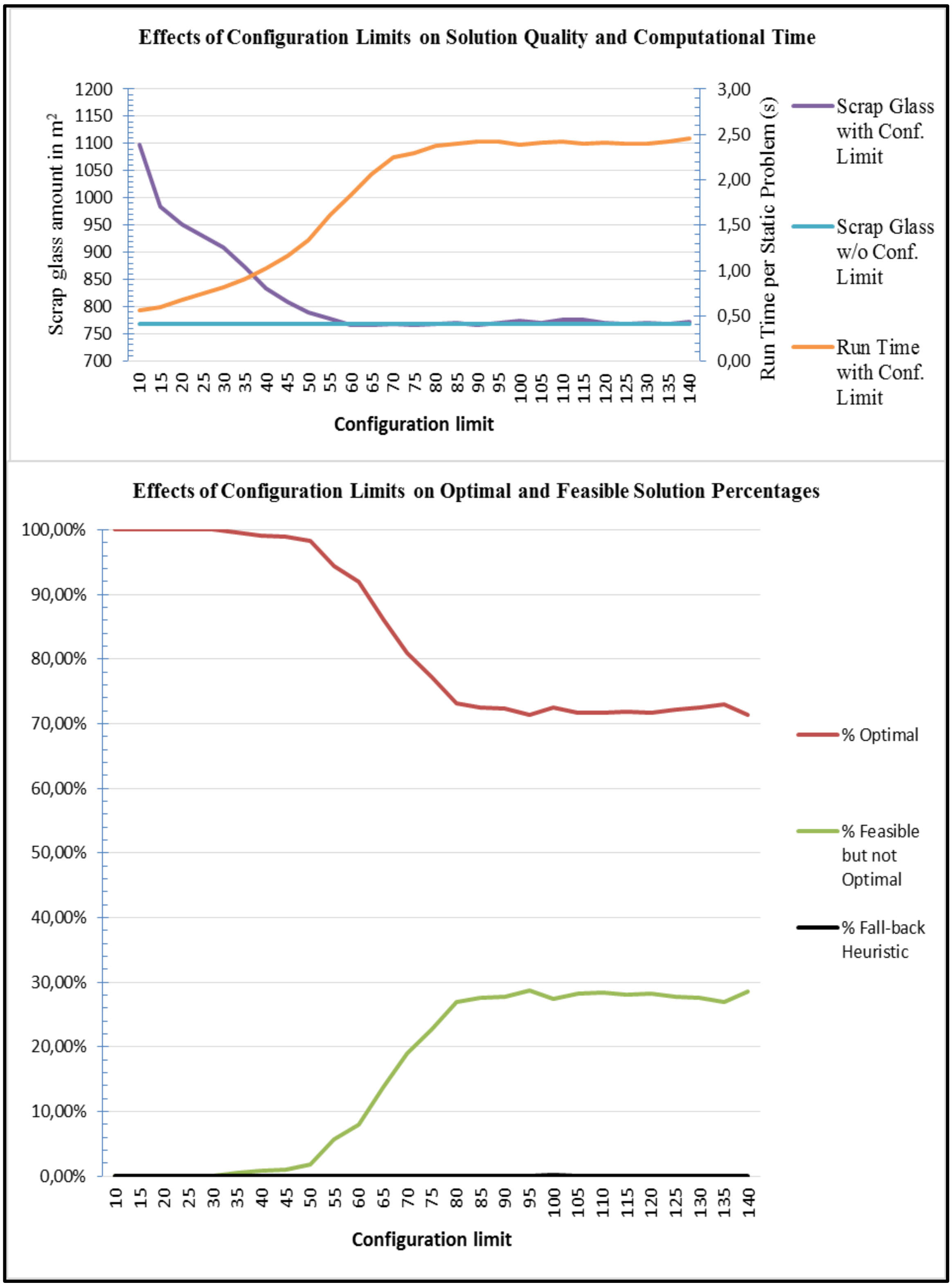


Figure 5.1. Effect of configuration limit on solution quality for an $L = 6$m SCP length and under 3 seconds run time limit

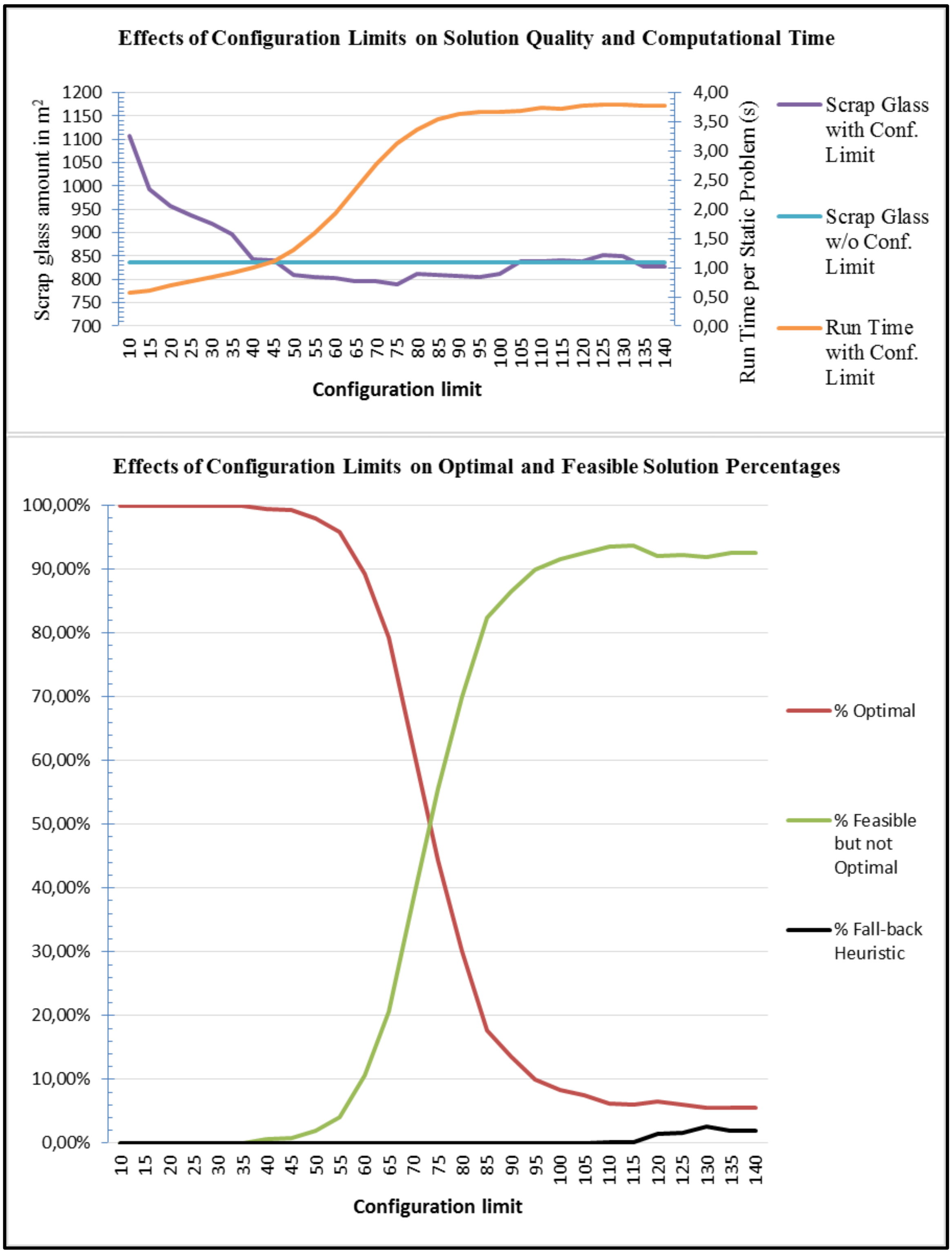


Figure 5.2. Effect of configuration limit on solution quality for an $L = 9$m SCP length and under 3 seconds run time limit

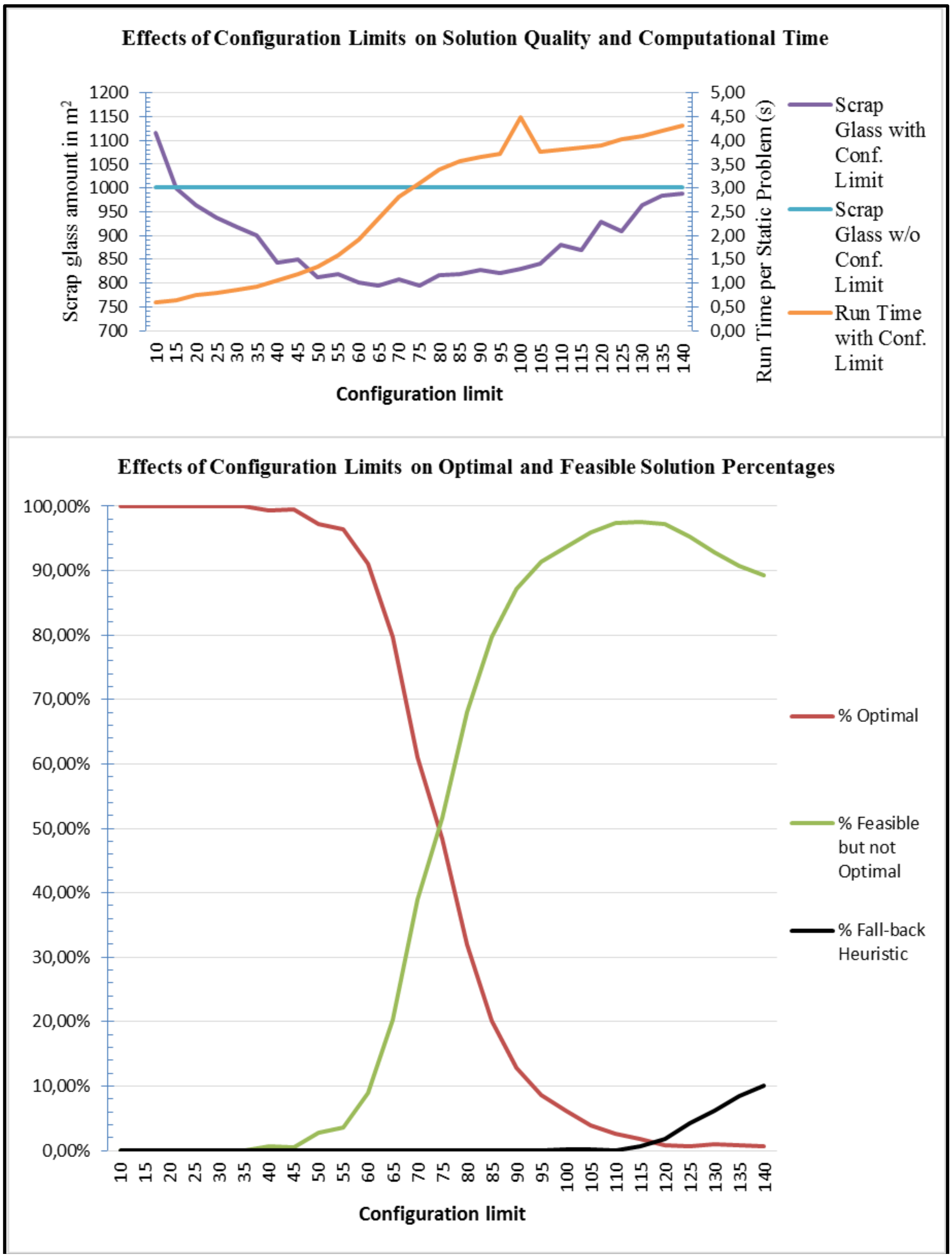


Figure 5.3. Effect of configuration limit on solution quality for an $L = 12$m SCP length and under 3 seconds run time limit

Based on our findings summarized in Figures 5.1 – 5.3, we see that there is no need to limit the configurations for the SCPs with $L = 6$m the version with no configuration limit offers comparable or better solution quality. Moreover, Table 5.2 indicates that the MIP model has better solution quality in this SCP length than DP, even when the number of configurations is unlimited. On the other hand, imposing a configuration limit for the larger SCPs created for with $L = 9$m or with $L = 12$m results in better solution quality. In fact, lowest scrap values are created for both SCP lengths when configuration limit is set to 75. Therefore, we decided to solve the MIP model under a configuration limit of 75 for our experiments with the 9m SCP length. The results are summarized in the right most column of Table 5.4. In this table, we see that the configuration limit led to an improvement in all instances. Although the MIP model still produces worse results than DP, the difference is significantly reduced compared to the MIP with unlimited configurations. By limiting the number of configurations, we were able to record an average improvement of 7 per cent in the MIP model and the average difference between DP and MIP was reduced from 11 per cent to 4 per cent. In the latter part of this study, we will see that the problem size, which will be further reduced by including production constraints in addition to the configuration limit, becomes a greater advantage in terms of MIP.

Table 5.4. Solution quality of DP, MIP and MIP limited to 75 configurations with $L = 9$m in m^2 of scrap glass produced

Defect Density	**Problem Instance**	**DP**	**MIP (3 sec)**	**MIP (3 sec) with configuration limit of 75**
Low	**1**	366.16	481.41	412.15
	2	404.71	500.59	441.63
	3	411.29	506.60	471.40
	4	370.65	492.86	419.30
	5	415.58	508.32	457.09
Average (Low)		**393.68**	**497.96**	**440.31**
Medium	**6**	765.92	835.76	788.25
	7	720.70	833.76	754.19
	8	773.65	870.39	763.35
	9	728.14	844.63	745.31
	10	744.17	856.66	771.93
Average (Medium)		**746.52**	**848.24**	**764.61**
High	**11**	1180.40	1226.50	1184.10
	12	1165.80	1217.60	1170.90
	13	1144.60	1193.80	1150.30
	14	1133.10	1169.50	1136.00
	15	1161.80	1212.10	1177.20
Average (High)		**1157.14**	**1203.90**	**1163.70**
Average (Overall)		**765.78**	**850.03**	**789.54**

5.1.3. Computational Time

In this section, we will report the computational performance of the online algorithm versions with both DP and MIP. Since the solution time per SCP of the GA is fixed at 3 seconds, we omit the GA results for simplicity. Similarly computational time of EE is less than 0.1 seconds per SCP. For this reason, we also omit the computational time of EE.

Table 5.5 shows total time spent by the methods in seconds for solving a 1,200m problem. Note that this total time includes the time to solve hundreds of SCPs within the 1,200m range. In general, MIP spends more time to solve a problem with low defect density. It is because a smaller number of defects leads to many dummy configurations, which results in a larger problem. Unlimited solution of MIP shows us the required time to solve all SCPs to optimality. On the other hand, version of MIP with a CPU time limit, especially of 3 seconds, shows the total time we can spend for production decisions. In real life settings the difference between solution times indicates the need for a performance improvement in the MIP model. This was the main reason for implementing a configuration limit to reduce the problem size in order to improve the MIP results.

Table 5.5. Total solution times of DP and MIP for solving a 1,200m problem

Defect Density	**Problem Instance**	**DP**		**MIP (unlimited time)**		**MIP (3 sec)**		**MIP (15 sec)**	
		6m	**9m**	**6m**	**9m**	**6m**	**9m**	**6m**	**9m**
Low	**1**	192.69	599.78	6464.78	98374.44	1793.57	2380.00	2809.48	9829.76
	2	192.69	756.31	4806.19	82975.25	1730.50	2354.54	2525.21	9848.22
	3	197.69	604.06	4272.54	110590.57	1751.80	2387.72	2730.64	10095.39
	4	196.24	576.16	4313.84	94998.10	1774.67	2385.44	2854.00	9919.13
	5	199.38	591.39	4350.30	93550.21	1822.66	2372.40	2690.53	10127.86
Average (Low)		**195.73**	**625.54**	**4841.53**	**96097.71**	**1774.64**	**2376.02**	**2721.97**	**9964.07**
Medium	**6**	256.63	909.82	5013.88	37924.86	1417.28	2236.34	2080.36	7745.28
	7	253.92	854.70	3978.33	39771.66	1413.72	2215.28	1984.28	7739.02
	8	250.06	947.14	4888.29	30438.52	1402.05	2205.65	1825.86	7939.34
	9	254.00	935.07	3065.23	30194.33	1387.51	2228.96	1924.73	7834.22
	10	264.54	1109.68	3252.89	45699.59	1455.73	2245.34	2135.45	7912.06
Average (Medium)		**255.83**	**951.28**	**4039.72**	**36805.79**	**1415.26**	**2226.31**	**1990.14**	**7833.98**
High	**11**	374.90	1279.17	3205.99	21579.68	1062.64	1994.00	1389.46	5281.23
	12	339.81	1333.23	4206.28	19824.56	1054.82	1996.69	1279.85	5510.22
	13	358.91	1302.65	1888.77	13281.31	980.21	1862.08	1284.80	4687.82
	14	358.59	1328.92	3054.38	19670.17	1025.74	1972.59	1354.52	5242.83
	15	396.50	1281.71	4892.01	15752.32	1001.95	1873.09	1306.38	5117.84
Average (High)		**365.74**	**1305.14**	**3449.49**	**18021.61**	**1025.07**	**1939.69**	**1323.00**	**5167.99**
Average (Overall)		**272.44**	**960.65**	**4110.25**	**50308.37**	**1404.99**	**2180.67**	**2011.70**	**7655.35**

5.2. COMPUTATIONAL RESULTS OF THE METHODS WITH PRODUCTION TARGETS

All of the experiments up to this point are based on the assumption that the production targets are unlimited. However, production preferences in a real production environment also vary according to production targets. In this section, we report our computational results for sample problems with production targets.

In the first set of experiments, we include production targets as constraints into our solution methodologies as explained in Subsection 4.5.1 and report the performance of the methods under production restrictions. Then, in the second set of experiments, we include production targets into the objective function as explained in Subsection 4.5.2. Lastly, all experiments are repeated for the MIP model with configuration limit in order to improve solution quality.

These experiments aim to evaluate the performance of the proposed methods with production targets and asses their suitability for the production environment.

5.2.1. Experimental Design

All previous parameters explained in Subsection 5.1.1 are also used in the creation of this experimental design. In addition, the production targets were set as 320 for low defect density instances for each product, 280 for medium defect density instances and 240 for high defect density instances. These values were calculated based on the average yield for each defect density level. When we were testing the new versions of the solution methods which include the production targets, we used 9m SCP length and 3 seconds time constraint. As discussed previously, this setting provides a good balance between the solution quality and CPU time requirement. For the performance improvement heuristic explained in Subsection 4.4.3, the first 75 configurations in I^0 are used in the MIP model.

In order to include production targets into the objective function, we have to set the value of coefficient α discussed in Subsection 4.5.2. Before starting the experiments, various α values were tested by using a subset of the problems in our experimental design. The aim of this preparation is to find a suitable α value for the model.

5.2.2. Solution Quality

In this section, we summarize the results of different versions of the online algorithm with production targets in terms of solution quality. To compare the performance of the methods with GA, DP and MIP, constraints on production targets have been added to all methods and the objective functions have been updated as described in Subsection 4.5.2.

Finding a good value for the coefficient α is crucial for the performance of the model with production targets. When the α value is too small, the production targets will not have a significant effect on the objective function. When it is too large, the model will focus on homogenously distributing production targets regardless of the amount of scrap glass. For this reason, choosing the α value is vital for the quality of model output. We performed a binary search to determine a good α value. We ran the MIP model for one problem instance from each defect density by changing α values between 100 and 1,000,000. Based on the result of our tests, the best value for α was determined as 88,000 for low and medium defect density instances and 100,000 for high defect density instance.

Table 5.6 summarizes the solution quality of methods for all problem instances in terms of generated area of scrap glass in square meters. In the table, GA+CO column shows the results of GA and DP+CO column shows the results of DP method. Both methods are updated to include production target both as constraints and in the objective function. MIP+C stands for the version of the MIP model where production targets are considered only as constraints as described in Subsection 4.5.1. Similarly, MIP+CO (with production target constraints and updated objective function) column shows the results of the MIP model with coefficient α. In the last column, MIP+COC (stands for MIP model with production target constraints, updated objective function and configuration limit) shows the results of the MIP model with a configuration limit of 75. All values are scrap glass amounts in m^2 produced by methods; therefore low values mean better solution quality. At first glance, we can compare the MIP+C and MIP+CO columns to see that a more homogeneous production distribution increases solution quality by about 3 per cent. Similarly, we can say that the MIP+CO version yields 6 percent better results than GA+CO and 4 per cent better results than DP+CO on the average.

We previously have tested the impact of configuration limits on the problem instances without production targets and found that configuration limits in MIP model can lead better solutions. For this reason, we have also tested this practice for the MIP model with production targets. In this version, we can see an additional 1.5 per cent improvement over MIP+CO results. In summary, when we compare the solution qualities of all methods with production targets, we observe that the MIP model yields about 7.5 per cent better results than GA+CO and about 5.5 per cent better results than DP+CO.

Table 5.6. Solution quality of methods (in m^2 of scrap glass) with production targets ($L = 9\text{m}$)

Defect Density	**Prob. No**	**GA+CO**	**DP+CO**	**MIP+C**	**MIP+CO**	**MIP+COC**
Low	**1**	536.08	564.71	551.54	538.09	529.21
	2	579.02	619.09	621.09	583.6	585.89
	3	577.01	635.12	600.2	582.74	571.58
	4	547.53	606.21	580.45	568.59	539.52
	5	575.3	629.11	625.96	588.12	569.57
Average (Low)		**562.99**	**610.85**	**595.85**	**572.23**	**559.15**
Medium	**6**	1007.2	994.9	971.15	951.97	937.08
	7	978.3	967.71	951.68	935.37	912.76
	8	1017.2	990.89	969.43	951.97	941.09
	9	1004.9	959.12	984.31	930.79	891.29
	10	963.99	955.4	935.37	899.59	877.84
Average (Medium)		**994.32**	**973.6**	**962.39**	**933.94**	**912.01**
High	**11**	1447.7	1339.2	1363	1310.3	1310.3
	12	1416.2	1356.1	1325.2	1281	1310.3
	13	1425.7	1337.8	1325.2	1310.3	1262.1
	14	1424.8	1351.5	1332.1	1310.3	1285.1
	15	1448.6	1355.3	1354.4	1318.3	1336.7
Average (High)		**1432.6**	**1347.98**	**1339.98**	**1306.04**	**1300.9**
Average (Overall)		**996.64**	**977.48**	**966.07**	**937.4**	**924.02**

Recall that DP yields better results than MIP when no production targets are imposed. On the contrary, MIP has better solution quality when we have production targets. At first glance, this seems counter intuitive. Therefore, we examine the effects of production targets on MIP solution quality in an effort to find the reason of this improvement. In

Figure 5.4, amount of scrap glass produced is shown for the two MIP versions with and without production targets (MIP vs MIP+COC). For the first 200 iterations, the two MIP versions produce almost the same solution quality. Since we enforce balanced production distribution, after 200 iterations, MIP+COC starts to sacrifice some more glass to keep the production homogeneous.

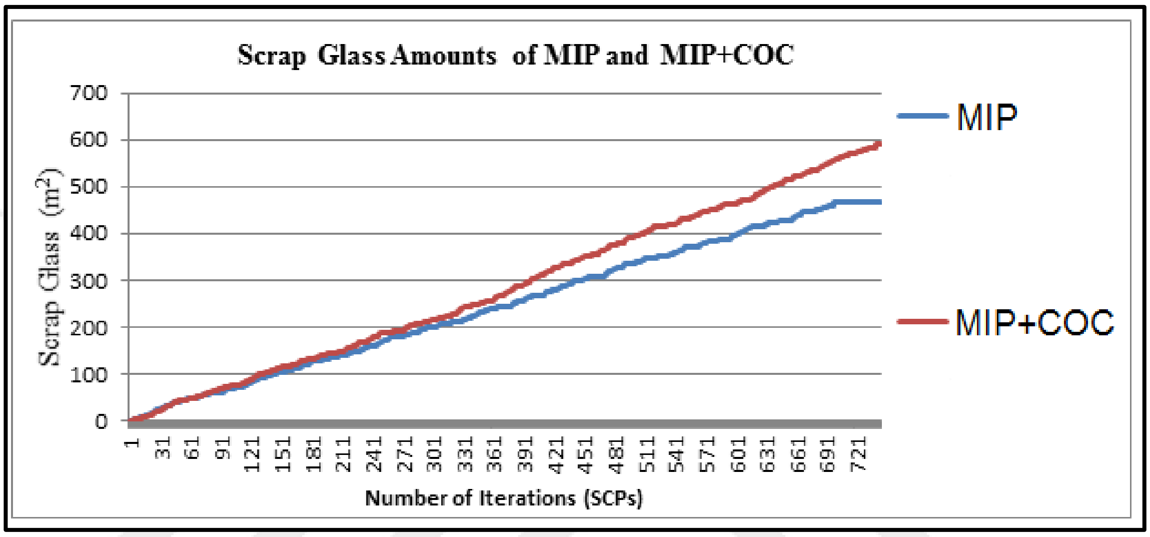


Figure 5.4. Scrap glass amounts (m^2) produced by two MIP versions (MIP vs. MIP+COC) over iterations on a problem with medium defect density ($L = 9$m)

Similarly, as you can see in Figure 5.5, after the 466th iteration MIP+COC reaches a production target of a product. Thus, MIP+COC starts to produce solutions by using only 4 different products which decreases solution quality. On the other hand, additional constraints and less number of available products create a significant advantage in terms of reduced problem size. In this new environment, MIP solves less complex problems in a smaller solution space. Therefore, the number of optimal solutions for MIP with production targets is relatively higher than MIP without production targets as shown in Figure 5.6.

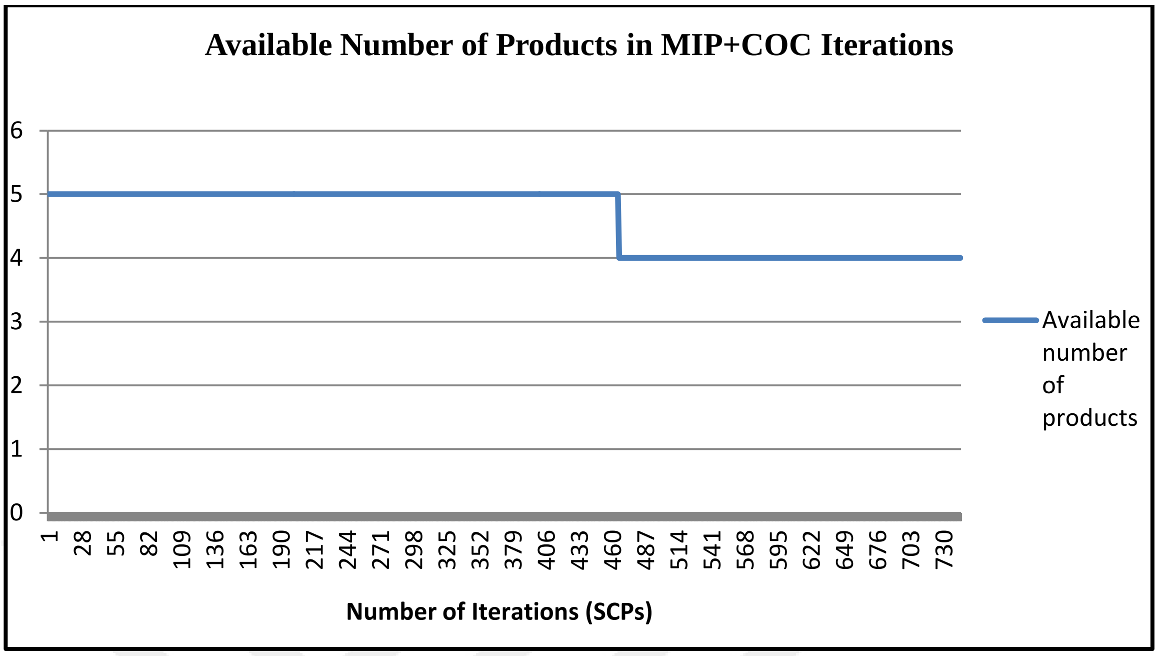


Figure 5.5. Change in the number of available products in MIP+COC over iterations on a problem instance with medium defect density under 3 seconds time limit ($L = 9$m)

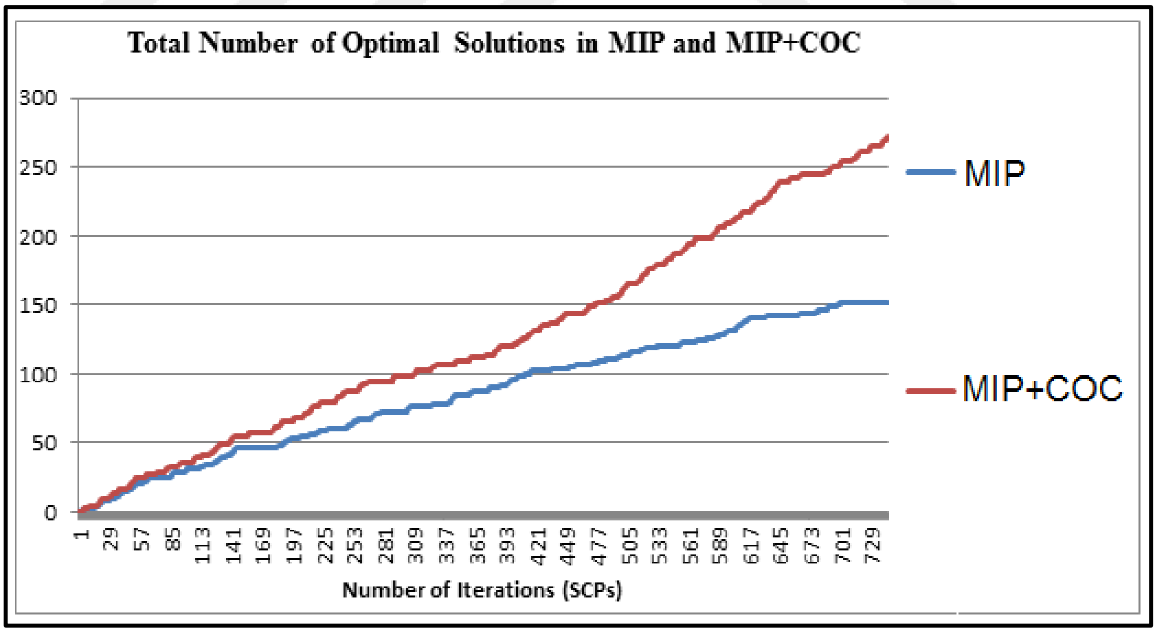


Figure 5.6. Number of optimal solutions in MIP and MIP+COC over iterations on a problem instance with medium defect density under a 3 seconds time limit ($L = 9$m)

Figure 5.7 shows the change of optimal solution percentages over time. Percentage of optimal solutions of MIP without production target has a declining trend. On the other hand, MIP with production targets produces more optimal solutions after iteration 466. This improvement is expected because one of the products reaches its production target and the number of available products decreases. Therefore MIP+COC starts to solve simpler problems and produces more optimal solutions.

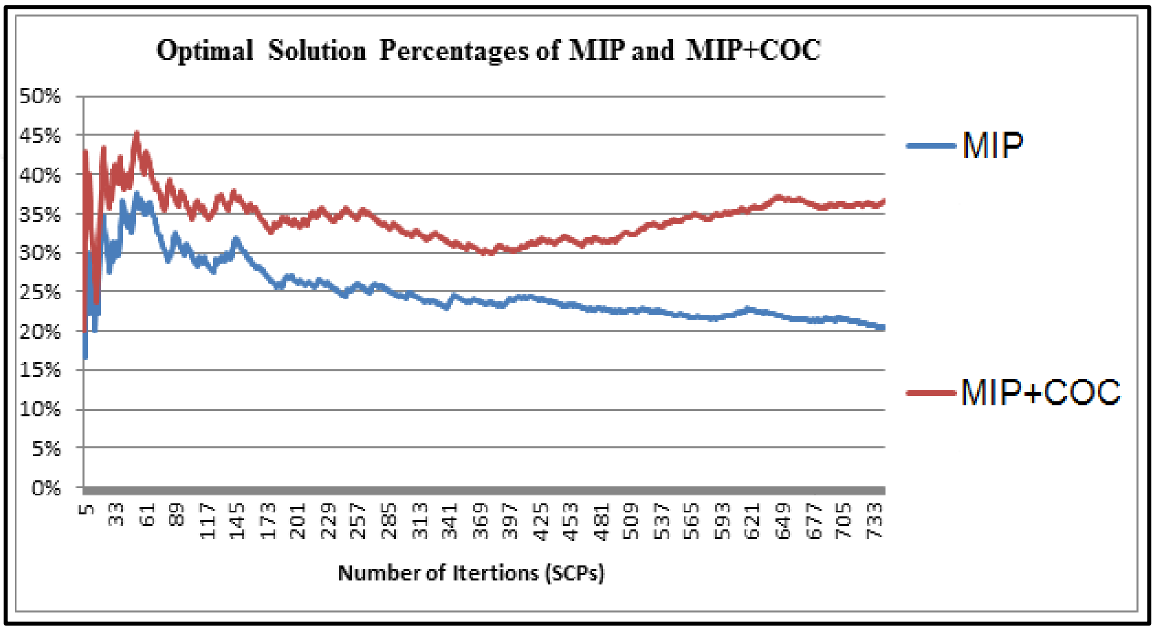


Figure 5.7. Optimal solution percentages of MIP and MIP+COC over iterations on a problem instance with medium defect density under a 3 seconds time limit ($L = 9\text{m}$)

We know that a high ratio of optimal solutions leads to better solution quality for the MIP model. MIP without production targets can only produce 20 per cent optimal solutions which is not enough to exceed the solution quality of DP. On the other hand, MIP with production targets produces 37 per cent optimal solutions which results in better solution quality than DP. This analysis suggests that the MIP version with production targets is capable of producing high quality results if sufficient CPU time can be allocated to solve the MIP model to optimality.

5.2.3. Computational Time

Table 5.7 shows total time spent by the methods in seconds for solving a 1,200m problem. Although using production targets in the objective function improves solution quality, it has no significant time advantage for the MIP model. However one can see that limiting the number of configurations has a positive effect on the computational time requirement of the MIP+COC model since it reduces the problem size. Even if DP+CO is the fastest method for $(L = 9\text{m})$ SCP length, by this reduction, MIP+COC can produce better results than DP+CO by spending almost the same computational time in problems with high defect density. Unfortunately the CPU time requirements of MIP+COC is significantly higher than DP+CO for instances with low and medium defect densities.

Table 5.7. Total solution times of solution methods for solving a 1,200m problem (in seconds)

Defect Density	**Problem Instance**	**GA+CO**	**DP+CO**	**MIP+C**	**MIP+CO**	**MIP+COC**
Low	**1**	3004.31	653.28	2594.08	2620.27	2366.41
	2	2914.63	682.49	2561.51	2564.72	2425.35
	3	2962.12	650.08	2538.29	2932.12	2453.08
	4	3092.68	664.13	2557.95	2279.85	2423.70
	5	2989.09	675.80	2563.26	2654.51	2410.30
Average (Low)		**2992.57**	**665.16**	**2563.02**	**2610.29**	**2415.77**
Medium	**6**	3290.66	1066.97	2385.35	2448.70	1908.87
	7	3226.92	1079.85	2362.34	2440.03	1929.35
	8	3371.39	1031.49	2435.32	2429.33	1931.36
	9	3205.75	1070.72	3353.44	2459.17	1987.29
	10	3180.09	1082.43	2391.40	2414.23	1932.55
Average (Medium)		**3254.96**	**1066.29**	**2585.57**	**2438.29**	**1937.88**
High	**11**	3778.28	1425.09	1900.54	1767.39	1469.18
	12	3603.92	1423.72	1848.62	1948.27	1558.12
	13	3801.80	1428.83	1812.07	1771.02	1508.09
	14	3810.84	1413.37	1856.71	1879.80	1498.02
	15	3752.78	1478.98	1763.54	1779.62	1393.66
Average (High)		**3749.53**	**1434.00**	**1836.30**	**1829.22**	**1485.41**
Average (Overall)		**3332.35**	**1055.15**	**2328.29**	**2292.60**	**1946.36**

5.3. COMPUTATIONAL RESULTS OF THE ADAPTIVE ALGORITHM

Previous experiments have shown that the performance of the α value depends on the defect density and the distribution of product targets. These two factors may not be constant during the production process. Therefore defining the value of α at the beginning of the production process and keeping it constant throughout the production process is not an effective way in that it ignores the changing conditions of the production line. This chapter presents the results of adaptive version of MIP where the value of α is updated throughout the production.

The aim of this experiment is to compare the solution quality of the adaptive algorithm with the solution qualities of the previous MIP versions and present its performance in comparison to others. Moreover, by this set of experiments, we try to find further options for improving solution quality of adaptive method by testing the parameters of the algorithm such as time limit and step size. Initial α value and initial production targets are other factors that may affect the solution quality. These factors are also tested in this section.

5.3.1. Experimental Design

Before testing the adaptive version of MIP, we have examined the effect of the update cycle length ($4k$) on α decisions. In Figure 5.8, the x-axis shows the value of k and y-axis shows the number of α updates. The red line shows the maximum number of possible α updates in the entire production run. For instance, when k = 13, there is a potential α update in each update cycle of $4k$ = 52 iterations and the online algorithm completes the whole production in 1355 iterations. Therefore there are 26 cycles. Since there is no update in the first cycle, the number of potential updates is 25. The blue line in Figure 5.8 shows the number of successful updates. Since we know that the best value of α for this instance is 88,000 from previous experiments, we evaluate each α update that brings it closer to 88,000 receives a + 1 point, updates in the other direction receive a - 1. For instance, at k = 13, there are exactly the same number of correct α update decisions as the incorrect ones. Thus, the score is 0 even if there are 25 potential updates. On the other hand, at k = 55, there are 5 potential updates all of which are correct. This study shows that k values less than 33 produce unstable results. On the other hand, k values higher than 50 generally does not make a decision of incorrect updates. The disadvantage of high k values is that there is a limited number of update chances, therefore update speed is relatively slow. In our experiments, we use the setting k = 33 where we observe the highest number of successful update in our initial experiments.

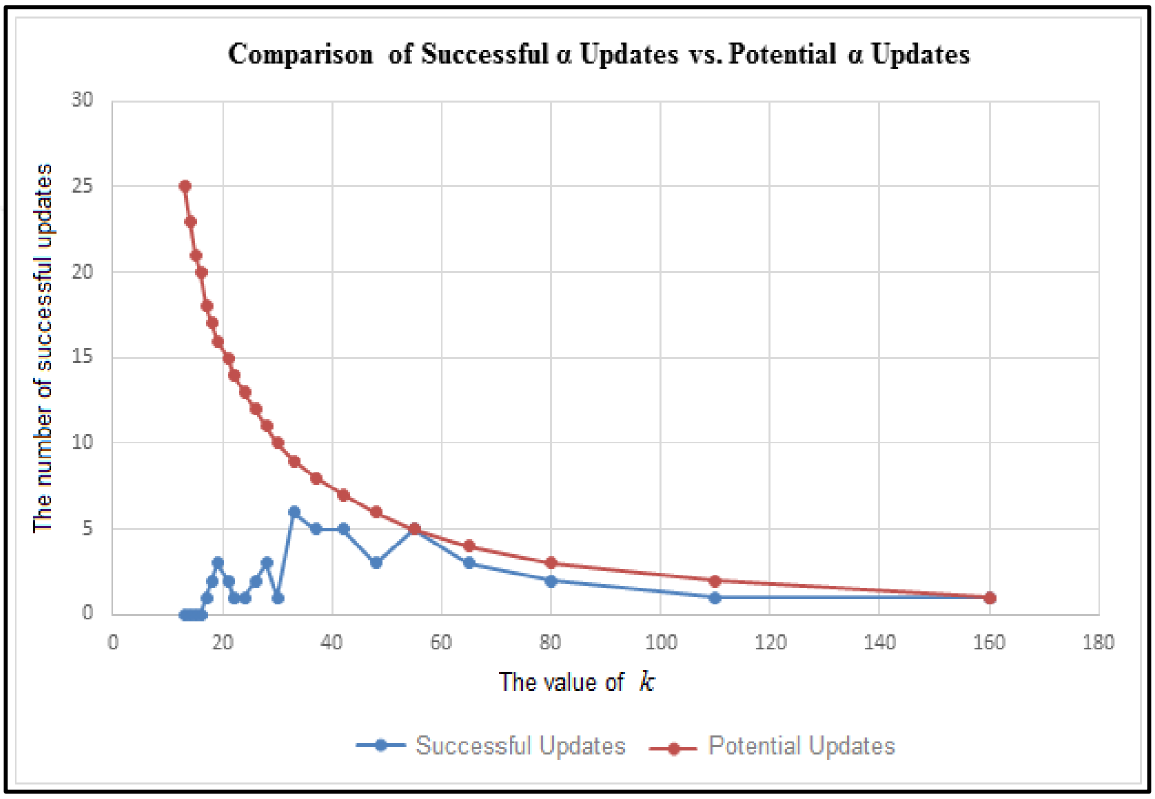


Figure 5.8. Comparison of successful α updates with potential α updates for various k values on a problem instance with medium defect density under a 3 seconds time limit ($L = 9$m)

All experiments are performed on a personal computer that has limited CPU performance comparing with industrial hardware. Operations which can be completed in 3 seconds with industrial hardware will take more time on a personal computer. This difference leads us to test the effects of time limit over the solution quality of the adaptive algorithm. Figure 5.9 shows scrap glass area in m^2 vs. time limit in seconds. When we inspect the solutions presented here, the performance with a 6 seconds limit appears satisfying in terms of solution quality. Moreover it is safe to assume that a computer twice as fast as the one used here will be available in an industrial application. Thus, a time limit of 6 seconds is used in our remaining experiments.

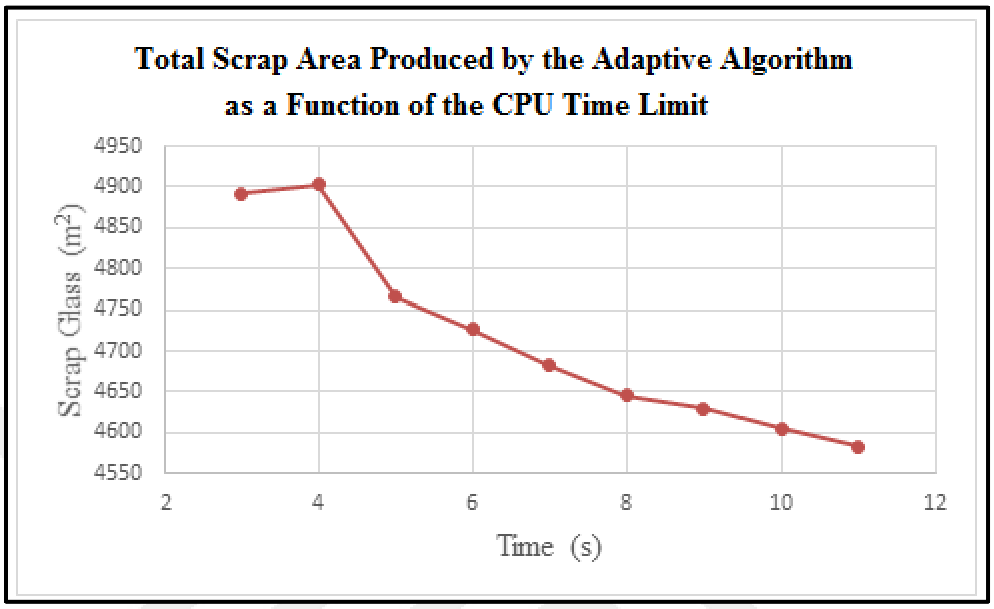


Figure 5.9. Total scrap glass area (m^2) produced by adaptive algorithm under different time limits on a problem instance with medium defect density ($L = 9$m)

Step size (ε) defines how fast the algorithm can update the α values. Larger step sizes improve solution quality faster, however precision of the α value is low. On the other hand, with smaller step sizes the solution quality improves slowly but the precision of the obtained α value is higher. Thus, one cannot conclude that smaller or larger step sizes are better. In our previous preliminary experiments, we used 10,000 as step size, but in this section we further tested this value to obtain better solution quality. Figure 5.10 shows the change in scrap glass area by step size increases. Since the best solution quality is obtained when step size equals to 8,000, this value is used in latter experiments.

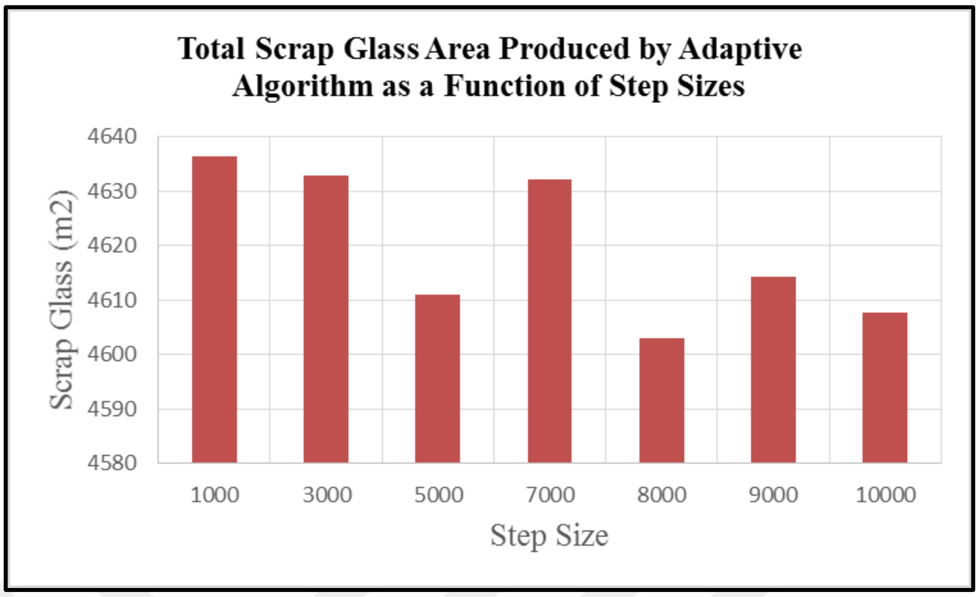


Figure 5.10. Total scrap glass area (m^2) produced by the adaptive algorithm under different step sizes on a problem instance with medium defect density ($L = 9$m)

The adaptive algorithm requires more data for better estimations. Moreover, it may take longer to reach an α range where the algorithm starts to produce better solution quality. Therefore, unlike the previous experiments, the length of the glass sheet for the problem instances in this section is set at 6,000m instead of 1,200m in the experimental design of the adaptive algorithm. The effect of configuration limit on MIP versions has already been tested in the previous experiments. Therefore, a configuration limit of 75, which produced good results for the earlier runs is used for all adaptive versions of the algorithm where MIP is used to solve the underlying static problem.

The main motivation for the adaptive algorithm is to provide an efficient production process by balancing the production according to production targets in order to prevent an early completion of any product. In our preliminary adaptive experiments, we started with the same production targets for all of the products. We expected that the adaptive algorithm to try to protect the initial balance along the whole production process. However, assigning the same production targets to all of the products is not a realistic approach. Therefore, to create a more realistic experiment and better observe the adaptive capability of the online algorithm, we start with random production targets for each product type for the problem instances in this section. In these experiments, we expect to see more production of products with higher production targets in order to prevent early completion of others. Table 5.8 shows the production targets of products used in adaptive algorithm experiments. For each problem instance, production targets of products are shown in the columns of the table.

Table 5.8. Initial production targets of products used in the adaptive algorithm experiments

Defect Density	**Problem Instance**	**Production Targets of Products**				
		1	**2**	**3**	**4**	**5**
Low	**1**	1581	2783	628	2290	1656
	2	3904	1080	1444	2206	727
	3	1684	1274	1465	2360	1454
	4	1972	2510	736	2104	1644
	5	3768	2201	618	1407	1675
Medium	**6**	1376	1449	1275	1394	1549
	7	3428	2193	690	1011	1177
	8	1609	2612	1099	1446	870
	9	2431	1645	590	1731	1637
	10	2166	2655	957	1594	685
High	**11**	2555	919	635	1057	1634
	12	3718	1722	640	952	645
	13	1543	1384	1026	1122	1210
	14	3193	1152	414	2185	737
	15	3363	1260	741	1215	801

In the previous experiments, we observed that the most efficient values of α were 88,000 and 100,000 for the MIP model with fixed α. When the adaptive algorithm starts with $\alpha = 0$, it spends some time to reach the reasonable α values. Therefore, it produces more scrap glass during this period. If the algorithm starts with another α value, let's say 90,000, it can find the efficient α values within a short time and does not produce unnecessary scrap glass. In order to test the effects of initial α value, we repeated the experiments with two different initial α values, 0 and 90,000.

5.3.2. Solution Quality

Our first adaptive experiments are conducted to observe the changes in α and determine the parameters that affect the solution quality of the algorithm. We started with a 3 seconds time limit and a step size of 10,000. These parameters are tested and updated in experimental design, but the initial results are also included in this section in order to present the evolution of the study.

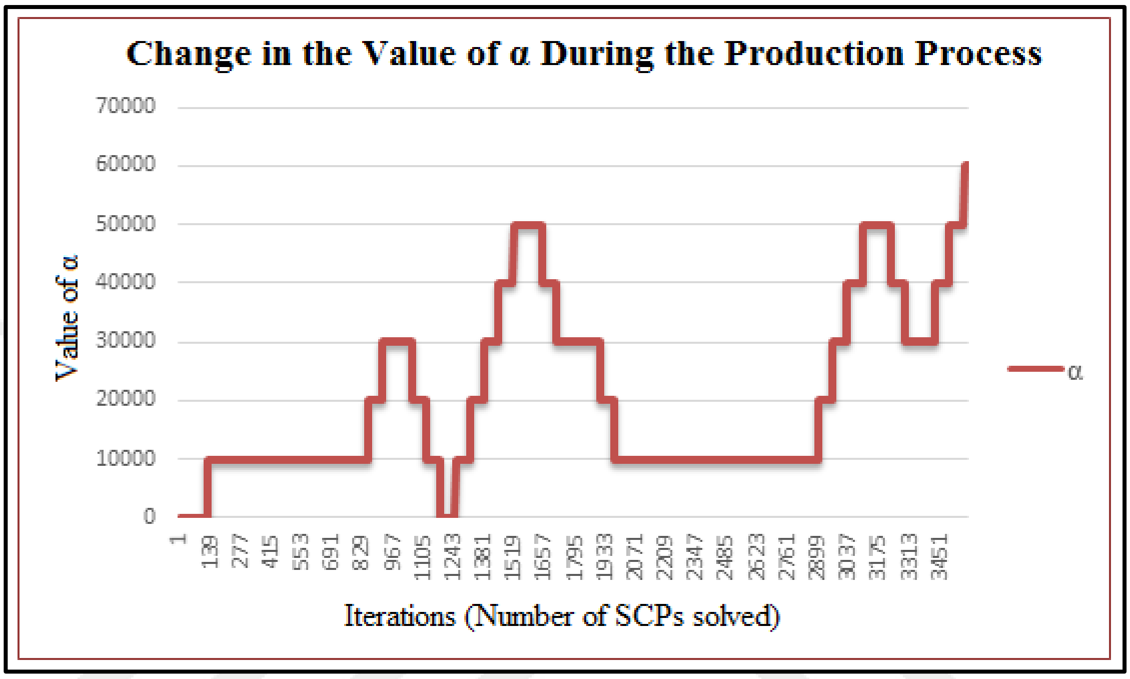


Figure 5.11. Change in the value of α during the production process on a problem instance with medium defect density under a 3 second time limit ($k = 33$, $stepsize = 10{,}000$, $L = 9$m)

Figure 5.11 shows that, value of α is continuously updated according changing conditions. Update of α values are affected by two main factors: production targets of the products and the defect density of the glass sheet. Figure 5.12 and Figure 5.13 show the α updates performed by the adaptive algorithm on problem instances with low and high defect densities respectively. One can see that the algorithm adapts itself with various update decisions by considering the existing circumstances.

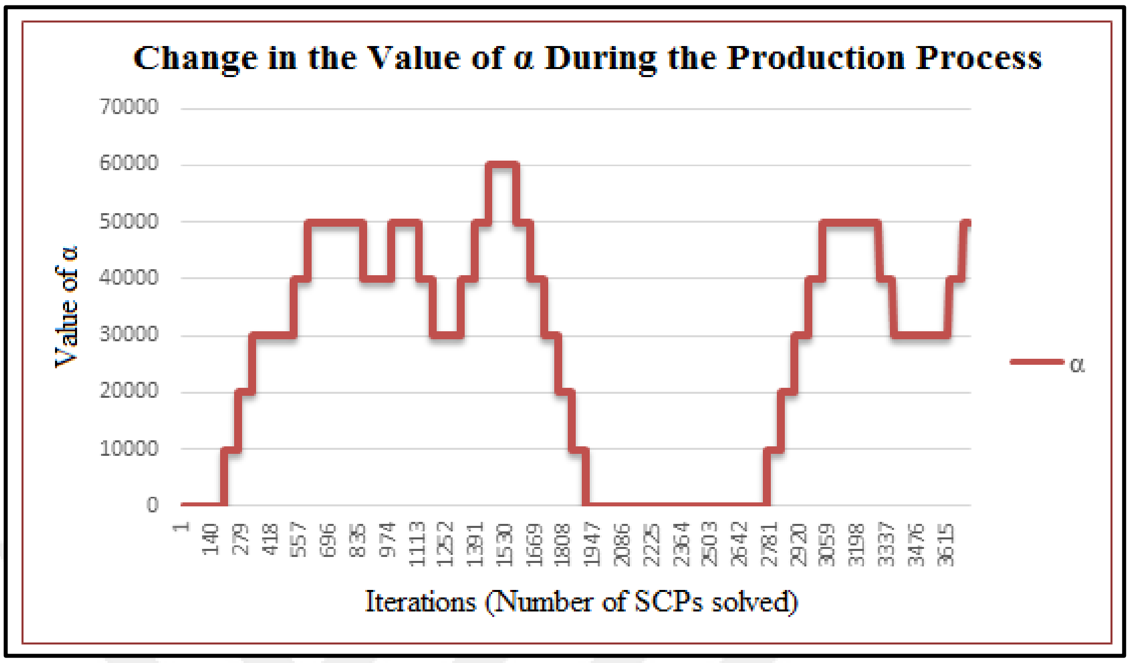


Figure 5.12. Change in the value of α during the production process on a problem instance with low defect density under a 3 second time limit ($k = 33$, $stepsize = 10{,}000$, $L = 9$m)

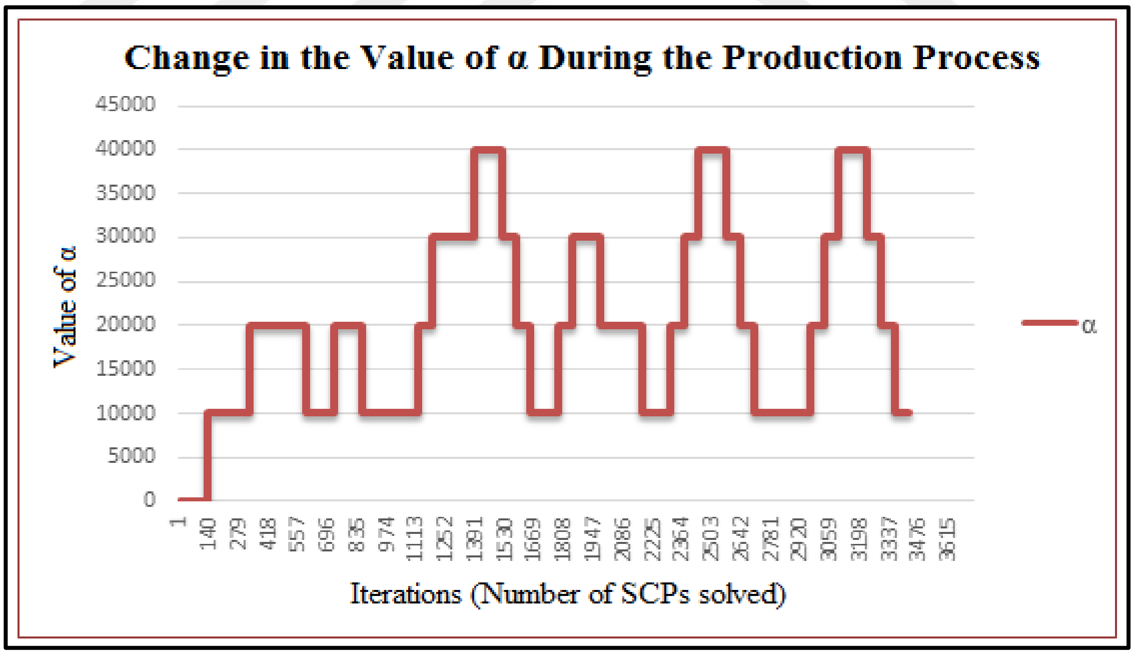


Figure 5.13. Change in the value of α during the production process on a problem instance with high defect density under a 3 second time limit ($k = 33$, $stepsize = 10{,}000$, $L = 9$m)

The preliminary results which are presented above show us the parameters that affect the algorithm and lead us to design the main experiments of this section. The results of this experiment in terms of solution quality are summarized in Table 5.9. This table shows the solution qualities of the adaptive model where α is updated continuously and the MIP+COC version that is run using a fixed α value. There are two columns for both models. The only difference between these columns is the α values used to initialize the run: first column shows the results with an initial $\alpha = 0$ and second one shows with initial $\alpha = 90{,}000$. All values are scrap glass amounts in m^2.

Table 5.9. Solution qualities of MIP+COC and the adaptive algorithm on problem instances with a glass sheet length of 6,000m under a 6 seconds time limit ($k = 33$, $stepsize = 10{,}000$, $L = 9$m)

Defect Density	**Problem Instance**	**MIP + COC**		**Adaptive**	
		α = 0	**α = 90000**	**α = 0**	**α = 90000**
Low	**1**	3455.20	3446.60	3290.90	3282.71
	2	3530.50	3268.30	3133.70	2900.97
	3	2969.20	2940.30	2933.70	2905.15
	4	3359.10	3287.80	3254.30	3185.22
	5	3639.30	3465.20	3303.60	3145.56
Average (Low)		**3390.66**	**3281.64**	**3183.24**	**3083.92**
Medium	**6**	4437.20	4772.10	4718.90	5075.06
	7	5728.10	5572.40	5545.00	5394.28
	8	4528.70	4923.60	4931.60	5361.63
	9	5298.80	5186.60	5225.50	5114.85
	10	5172.90	4934.20	4993.10	4762.70
Average (Medium)		**5033.14**	**5077.78**	**5082.82**	**5141.70**
High	**11**	7347.90	7306.70	7312.40	7271.40
	12	7928.60	7856.50	7838.20	7766.92
	13	6824.70	6728.50	6725.10	6630.30
	14	7768.90	7648.70	7714.00	7594.65
	15	7717.40	7663.60	7653.30	7599.95
Average (High)		**7517.50**	**7440.80**	**7448.60**	**7372.64**
Average (Overall)		**5313.77**	**5266.74**	**5238.22**	**5199.42**

Table 5.9 indicates that the adaptive algorithm provides an average of 1.5 per cent improvement compared to the version with fixed α when the initial α values are 0 in both

versions. The adaptive algorithm updates the value of α continuously along the production process to reduce the scrap glass area. However, it takes some time to reach reasonable α values. During this interval, the adaptive algorithm produces some more scrap glass than necessary. Therefore, in order to observe the potential of adaptive version, we repeated the same experiment with initial $\alpha = 90{,}000$ which is a more appropriate value based on our previous findings. As shown in Table 5.9, choosing a good initial α value has a positive effect on solution quality. By eliminating the time spent using inappropriate α values, it is possible to improve the solution quality by almost another 1 per cent.

Figure 5.14 and Figure 5.15 show the change in the value of α during the production process when it is initialized at $\alpha = 0$ and $\alpha = 90{,}000$ respectively. In the experiments of the MIP version with fixed α, we observed that the most efficient values of α change between 88,000 and 100,000 depending on the defect density. Therefore, when initialized at $\alpha = 0$, the adaptive algorithm increases the α value to obtain better solution quality as shown in Figure 5.14. It completes the production with a final α value of 24,000 on the average. On the other hand, when initial $\alpha = 90{,}000$, which is an efficient value, the adaptive model tries to preserve this value during the production. Even if there are small changes in the α value, adaptive algorithm completes the production with a final α of 92,333.33 on the average which is close to the initial value of α.

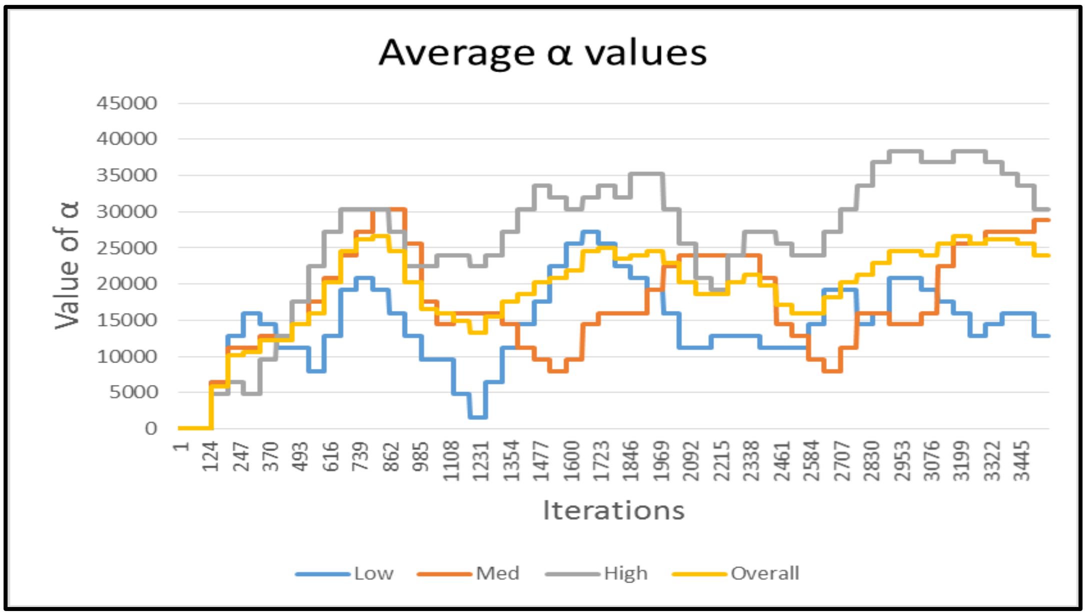


Figure 5.14. Change in the value of α during the production process when initial $\alpha = 0$

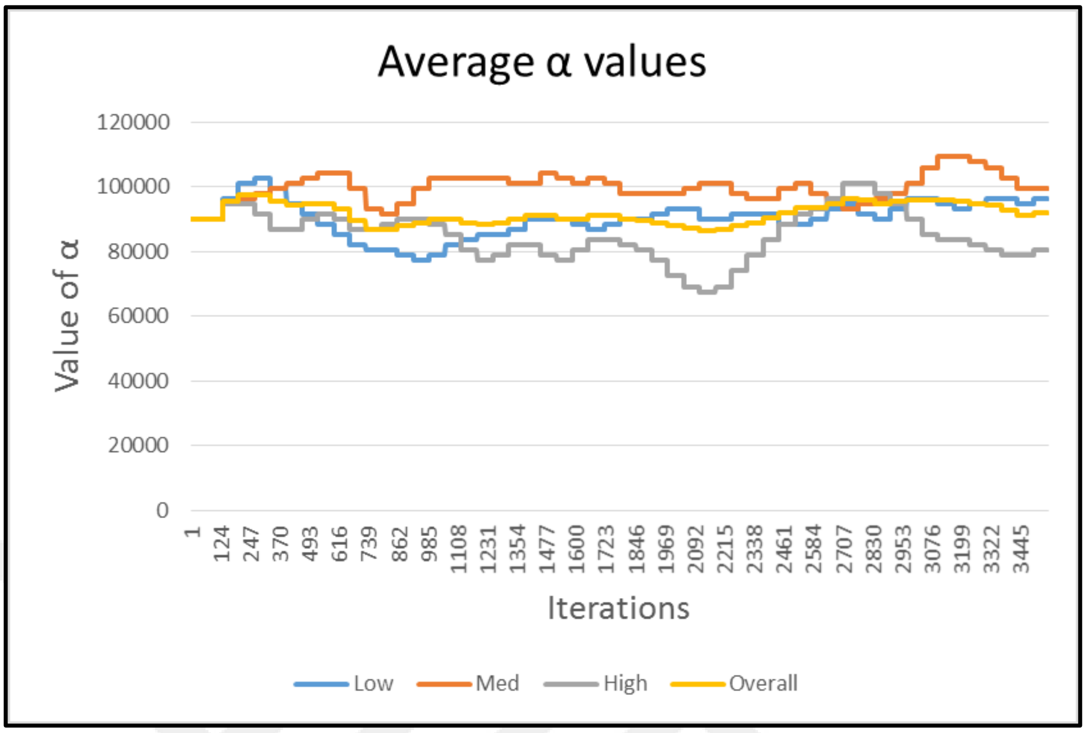


Figure 5.15. Change in the value of α during the production process when initial $\alpha = 90{,}000$

5.3.3. Computational Time

The only difference between the MIP versions in this experiment is the value of α which is a coefficient in objective function. Therefore, we do not observe significant computational time difference among the algorithm versions. In Table 5.10, computational times are listed in seconds. On the other hand, we observe that computational times depend on defect densities. Low defect densities require more time since the number of possible configurations is high. On the contrary, less computational time is required due to the limited solution space when defect density is high.

Table 5.10. Computational times of MIP+COC and adaptive algorithm in seconds on problem instances with length 6,000m under a 6 seconds time limit ($k = 33$, $stepsize = 8{,}000$, $L = 9$m)

Defect Density	**Problem Instance**	**MIP + COC**		**Adaptive**	
		α = 0	**α = 90000**	**α = 0**	**α = 90000**
Average # of iterations		**3795**	**3810**	**3859**	**3851**
Low	**1**	58511.07	59127.26	57856.06	58465.36
	2	58279.25	60489.68	55954.16	58076.41
	3	59069.55	59594.09	58309.32	58827.11
	4	60068.83	60635.52	59169.33	59727.54
	5	57441.78	58311.22	55607.75	56449.43
Average (Low)		58674.10	59631.55	57379.33	58309.17
Medium	**6**	44912.46	55058.77	51055.29	62589.34
	7	51949.08	50730.70	49312.66	48156.11
	8	46383.36	54609.57	49002.10	57692.75
	9	56437.34	56318.06	51687.03	51577.79
	10	53622.35	56585.51	51001.61	53819.95
Average (Medium)		50660.92	54660.52	50411.74	54767.19
High	**11**	38751.81	38475.47	37302.44	37036.43
	12	39374.86	40150.83	39102.18	39872.78
	13	41067.72	41771.58	39135.93	39806.68
	14	41899.31	41921.64	40486.58	40508.16
	15	40445.28	39668.26	39101.57	38350.37
Average (High)		40307.80	40397.56	39025.74	39114.88
Average (Overall)		49880.94	51563.21	48938.93	50730.41

5.4. OVERVIEW OF COMPUTATIONAL RESULTS

One of the primary goals of the study was to test the CBF rule, which is often used in the industry. For this reason we have designed our experiments with two methods that use the CBF rule (GA and DP) and another one that does not use this rule (MIP). In terms of solution quality, the MIP model is the best-performing method among the three, which clearly shows that the CBF rule produces suboptimal solutions despite its prevalence in the glass industry.

Under the CBF rule, we compare the solution qualities of our solution methods with the solution quality of EE. We observed that GA and DP yield significantly better quality solutions than EE. The improvements of GA and DP are 18 per cent and 25 per cent on the average respectively.

When we compare the two methods that use the CBF rule, we observe that DP produces a higher solution quality than GA. However, the time required to produce the solutions of DP varies considerably depending on the underlying SCP lengths (L). In addition, we cannot impose a time limit on DP. These two factors pose a risk that DP cannot produce solutions within the allowed time when higher SCP lengths such as 12m are used. For this reason, the practical use of DP is more likely to be with conservative SCP lengths such as 6 or 9m. On the other hand, although GA produces lower solution quality than DP, it guarantees to provide feasible solutions within the allowed time. Hence, we can say that GA has the practical usage for any SCP length. However, the solution quality of GA deteriorates as the SCP length increases. To summarize, when the CBF rule is used, it is recommended to use DP as long as it does not pose the risk of not terminating within allowed time, and to use GA if this risk exists. In the production environment, one way to eliminate the risk of DP not terminating in the allowed time is to run the GA version in parallel to DP as a back-up so that the GA solution is used whenever DP fails to terminate in time.

If we remove the CBF rule, we face a more complicated problem in the form of a MIP model. As expected, we need more time to solve this complex problem. However, whether we use the CBF rule or not, we have to solve the problem within the same time limit. The best choice is always to use the optimal solution if we can find it within the time limit. If not, we can use the best feasible solution we have so far. If the model does not even produce a feasible solution, we have to use a fall-back heuristic. Using this three-step-methodology, we can guarantee to reach a solution in the allowed time. Our computational results indicate that the solution quality depends which of these three solutions we use predominantly. We observe high quality solutions when the model produces mostly optimal solutions, but the solution quality is be relatively poor in cases where the model often resorts to the fall-back heuristic. In an effort to increase the percentage of optimal solutions produced by MIP, we propose to reduce the complexity of the problem by limiting the total number of configurations. We achieve this by removing some configurations present in the model that have a relatively limited effect on the cutting decision because they are in the last part of the SCP length and will be re-evaluated in the next iterations. The computational results demonstrate that reducing the number of configurations did not lead to a significant improvement in problems where the SCP length is set at 6m. SCPs with $L = 6$m, which is a relatively short length, are simple enough to produce an optimal solution within the allowed time. However, since the SCPs with 9m and 12m lengths are more complex, the limiting the number of configurations has a positive effect on the result for these settings. With this modification, we were able to record an average improvement of 7 per cent in the MIP model. However, DP still has a solution quality which is 3 per cent higher than MIP model despite the improvement in the MIP solutions.

Production targets are an important consideration for all of the solution methods so that they can be used in real life settings. When we incorporate production targets into the online algorithm only as constraints, we observe that some products are produced more frequently than others and reach their production targets much sooner. In the later stages of production, this situation reduces the number of products to choose from, which results in more scrap glass. Therefore we added a term for production balancing into the objective function that penalizes high differences in achievement of production targets among products. Using a properly determined objective function coefficient for production balancing provides 3 per cent additional improvement in solution quality. When this adaptive algorithm is used with a MIP model that has a limited number of configurations, we were able to improve the solution quality further by another 1.5 per cent. When we compare the three methods under production targets, it is observed that the MIP model yields an average of 7.5 per cent better solution quality than GA and 5.5 per cent than DP.

In an effort to create an adaptive method that updating the value of α based on the current conditions of the production line, we created an adaptive version of MIP and conducted experiments to find efficient parameters for this adaptive model. Based on initial experiments, we set the cycle length at $4k$ = 132 iterations (k = 33 iterations), CPU time limit at 6 seconds and step size at 8,000 and used a configuration limit of 75 as before. To better observe the adaptive behaviour of the algorithm, we used longer problem lengths (6,000m) and randomly set production targets for these experiments. The results of experiments showed that the adaptive model can provide an average of 1.5 per cent improvement in solution quality compared to the MIP+COC version where the value of α is constant during the production process. Moreover, starting with a good α value provides an additional 1 per cent improvement in solution quality. Overall, these results indicate that the adaptive method can generate slightly better results than the version with a fixed α value. Based on these results, we can conclude that the adaptive algorithm is capable of adjusting the value of α to the conditions of the float line, which suggests that it can run autonomously without the intervention of a human operator.

6. EXTENSIONS AND FURTHER APPLICATION AREAS

Both the online algorithm and the methods for solving the underlying static cutting problem proposed in this thesis can be generalized to solve many other cutting problems in the literature. However, as we will demonstrate with various examples in this section, these problems should have a structure that can be transformed to the one investigated here. In order to enable this transformation, the original problem should have the following features:

- It should be possible to reduce the original problem to a one-dimensional one.

In the glass cutting problem, this is achieved by having a limited number of cutting wheels whose positions are known and fixed during the production run and all cuts being limited to guillotine cuts. In order to use the GA, DP and MIP based solution methods proposed here, one should either have a one-dimensional problem in its original form or should be able to limit the number of possible cutting configurations along the second dimension to reduce it to one dimension.

- All products are defined and their sizes are known in advance.

This allows us to create the set of possible configurations and reduce the problem to one dimension. This can be done as a preprocessing step at the beginning of the production run so that the configuration set is known in advance. In fact, even when product definitions and sizes are subject to change, it is sufficient to have the up to date information for the products to be cut in the next static cutting problem.

One should also note that the algorithm proposed in this thesis works on an online problem where the stock can be assumed to be of infinite length. However, the problem is solved by decomposing the problem into overlapping cutting problems of fixed length. Therefore the same approach would be valid for online problems where individual stocks of fixed size problems are cut into products in an online fashion. In fact, our solution approach does not require that each static cutting problem is solved over a fixed stock size L. Therefore the same approach can be used when the problem involves variable stock size. To summarize, the approach proposed here is quite flexible and can be applied to many cutting problems of an online nature.

In this section, the reader can find how similar cutting problems can be solved by the methodology proposed in this thesis. To demonstrate the limitations of our approach, we also include some other problems where our methodology cannot be used to produce a solution.

6.1. CUTTING OF PAPER ROLLS WITH DEFECTS

[6] studies on a cutting problem from the paper industry. In this problem, a defective paper roll is slit to appropriate widths by paper mills, however defective area cannot be used and should be removed from the paper roll (see Figure 6.1).

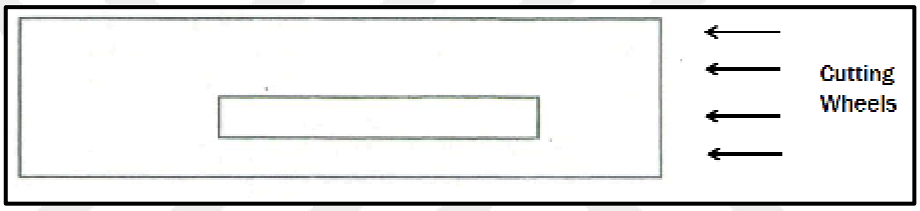


Figure 6.1. Cutting paper roll where there is a rectangular defect [6]

This problem can be converted to our problem by minor modifications. First, we have to define one configuration for all available paper widths. Second, all defective areas are assumed to be filled with point defects of highest defect class which are not accepted in any of the configurations. Third, the width of the paper roll is assigned as our static problem length. Since the problem is not continuous, it will be enough to solve it once as a static problem (see Figure 6.2).

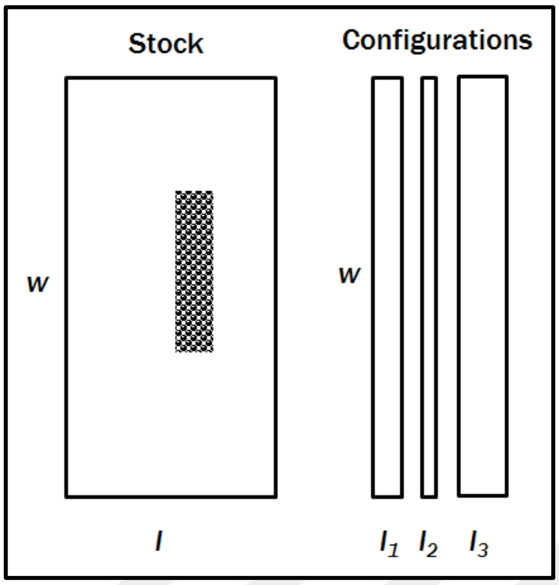
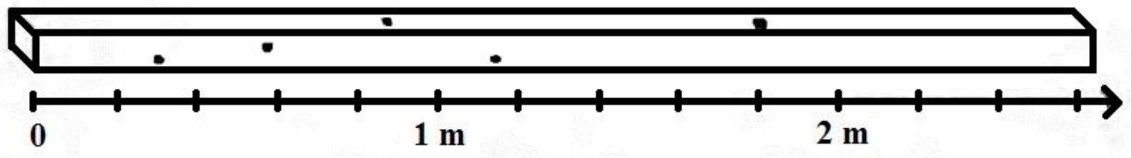
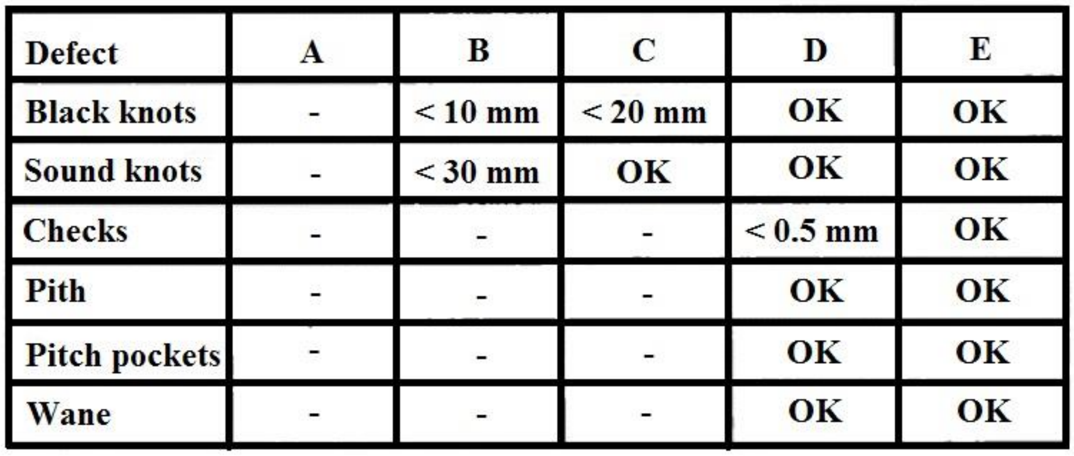


Figure 6.2. Transformation of the paper cutting problem

6.2. CROSSCUT OPTIMIZATION OF WOODEN BOARDS

[32] applies a dynamic programming method to a cutting problem that arises in the wood industry. In this problem, window parts with different sizes are cut from a wooden stock. All product sizes are known in advance. Similar to our problem, there are different defect types and window parts that have different quality requirements as shown in Figure 6.3. On the other hand, each window part can be placed at different rotations. The objective is maximizing the total value of products.





Defect	A	B	C	D	E
Black knots	-	< 10 mm	< 20 mm	OK	OK
Sound knots	-	< 30 mm	OK	OK	OK
Checks	-	-	-	< 0.5 mm	OK
Pith	-	-	-	OK	OK
Pitch pockets	-	-	-	OK	OK
Wane	-	-	-	OK	OK

Figure 6.3. Defects on wooden stock (top), defect types allowed for each quality class A – E with corresponding defect tolerances (bottom) [32]

The problem can be solved using our algorithm as a single static problem since all of our methods for solving SCP (GA, DP, MIP) are designed to handle problems with different defect types and products of various quality classes.

Problem is a three dimensional problem, therefore the first step is transforming it to one dimensional problem. Each side of wooden stock can be represented as a strip on a two dimensional plane as shown in Figure 6.4.

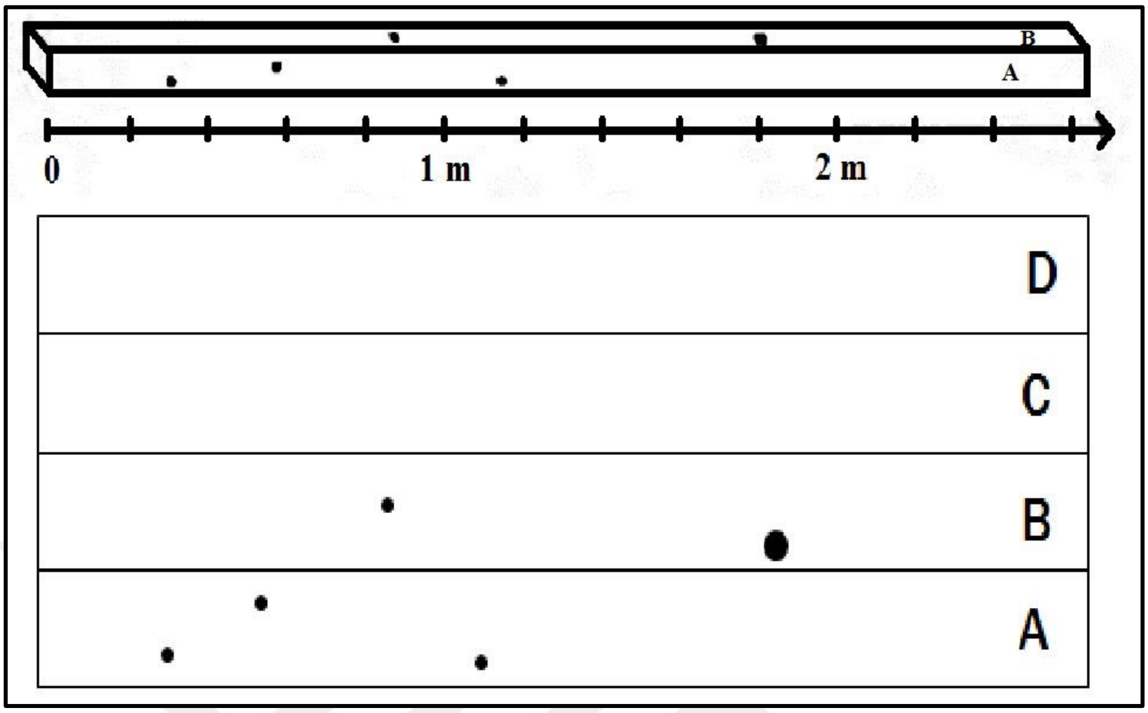


Figure 6.4. Transformation of wooden stock [32]

Similarly, each side of a product should be represented as a strip on the same plane. This transformation forms a two dimensional problem which corresponds to the original one. Here, the important point is defining all of the possible rotations of window parts as a unique configuration. In Figure 6.5, strip A can be used for the front surface of the shown product. However, it is also possible to rotate the product to cut its front surface from strip D if there is a critical defect on strip A. Therefore, all of the four cutting rotations should be defined as separate configurations. In others words, each product can be represented by multiple configurations in our model to reduce the problem to one dimension.

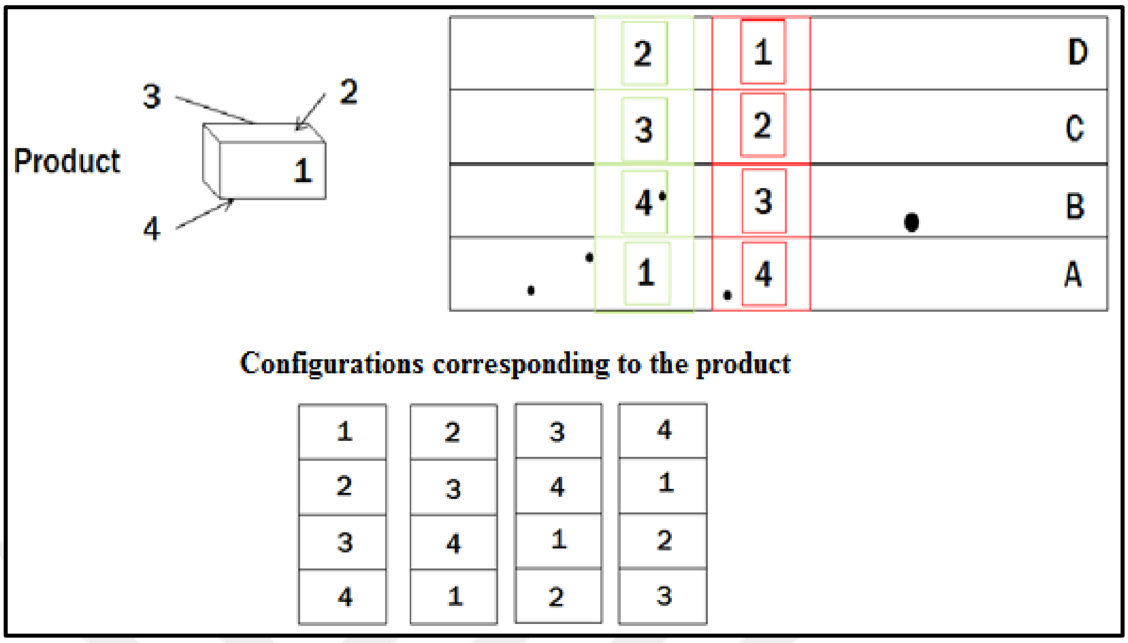


Figure 6.5. Transformation of products to configurations

6.3. CROSSCUT OPTIMIZATION IN A WOOD PROCESSING MILL

Another wood cutting problem is solved using dynamic programming by Fathi and Kainfar [9]. The problem arises in the lumber industry where cubic blocks with predefined dimensions and surface characteristics are cut from a wood stock. The problem is three dimensional and defects can be seen on each side of the stock. Each product has a quality requirement and can be cut by using different orientations as shown in Figure 6.6. The problem is different from the one explained in Section 6.2 in term of stock shapes and the orientation options. The crosscut area of the problem has a rectangular shape whereas the one in Section 6.2 has a square crosscut area.

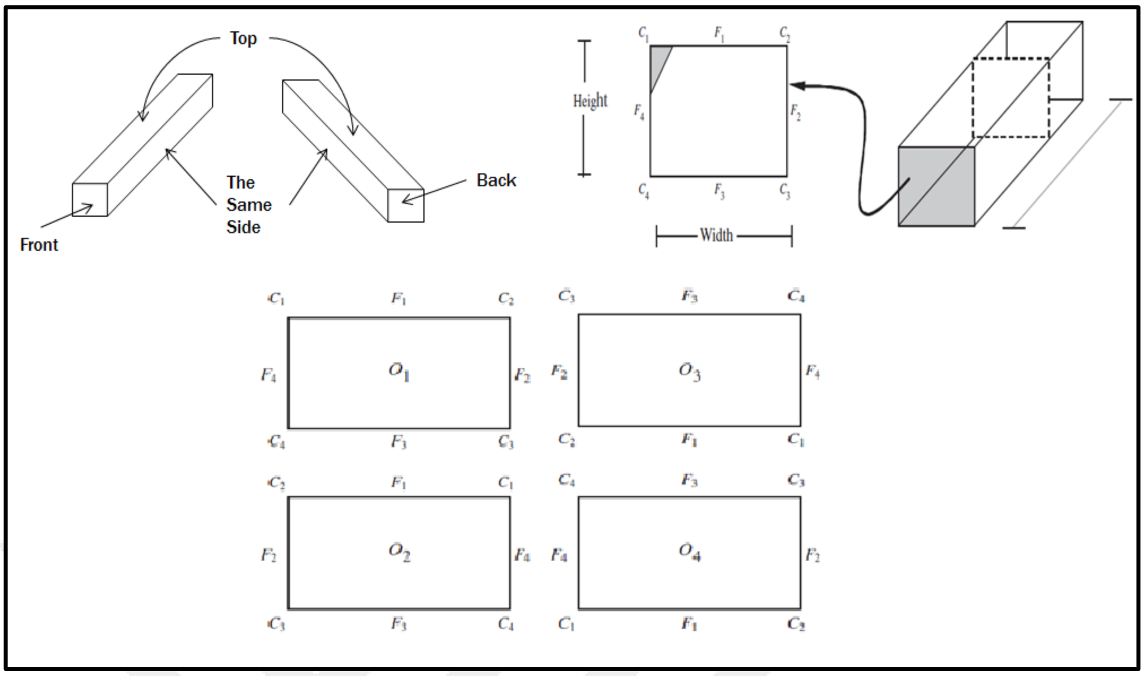


Figure 6.6. Cutting options (*O₁, O₂, O₃, O₄*) for different orientations [9]

First we have to transform the problem to one dimension. By treating each surface of the stock as a parallel strip on the two dimensional plane, the problem can be transformed into two dimension. This plane is the stock of our static problem. Similarly, we have to transform all of the products to a two dimensional plane with all of the possible orientation variations. Each variation corresponds to a configuration in the model, so one product is represented by more than one configuration. This representation provides another dimension reduction and creates a one-dimensional problem which is similar to ours. Since the width of all configurations are the same as the width of the stock in the final model (which is equal to the circumference of three dimensional stock), we can use our solution methodology (see Figure 6.7).

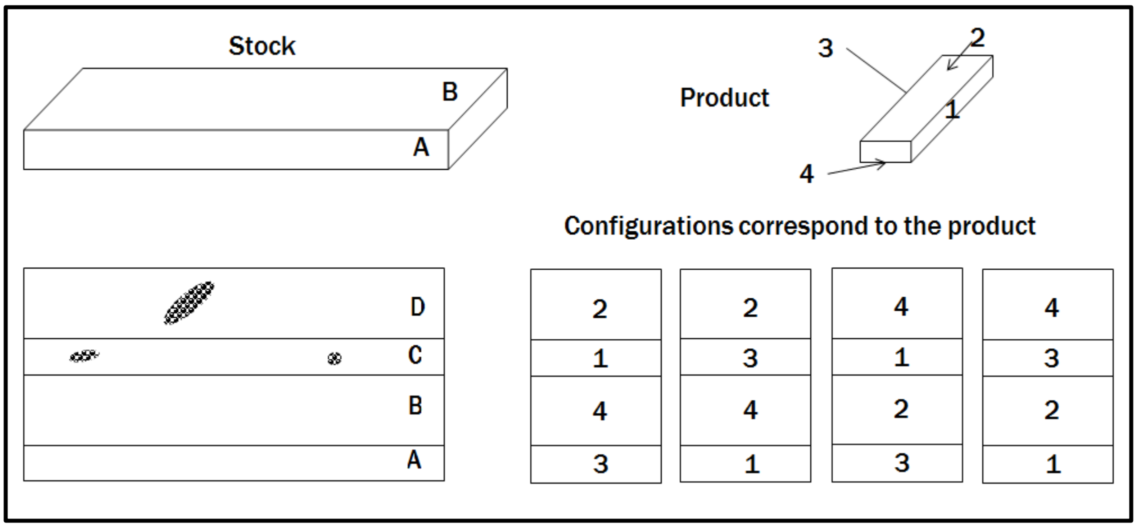


Figure 6.7. Transformed wood cutting problem

6.4. ROLLER BLIND PRODUCTION

[23] studies a cutting stock problem in roller blind production. Although there are no defect types or quality classes in this problem, the problem structure is similar to ours. They work on a continuous problem with two-stage guillotine cuts and rectangular products. However, after the online cutting is completed additional cuts are applied in an offline fashion to produce final products. In the offline cutting process, products can be placed such that defects on the stock is avoided. Although this detail increases the number of possible configurations significantly, the problem can be solved using our approach.

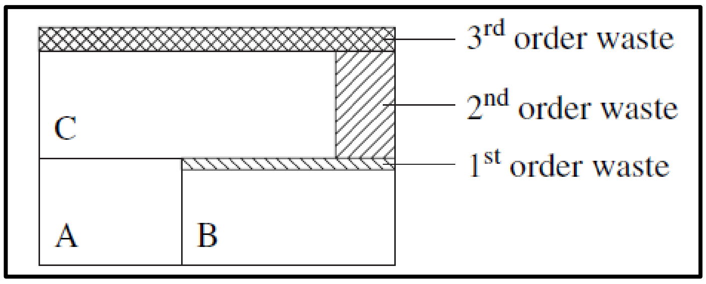


Figure 6.8. A feasible configuration that can be cut in two stages [23]

Figure 6.8 shows a feasible configuration. In the second cutting stage, the configuration is cut into products A, B and C by sacrificing some material. The value of the configuration is set as the total value of the nondefective products minus the value of discarded area. The second cutting stage allows various cutting orientations for a configuration, some of which are shown in Figure 6.9. It is possible to shift the products within the available area of configuration so that the model can avoid any defect on products. In Figure 6.9, defects are represented as circles. Depending on the defect location, any orientation of the configuration can be used to gain advantage. However, this necessitates all orientations to be defined as separate configurations in the model, which increases the problem size significantly.

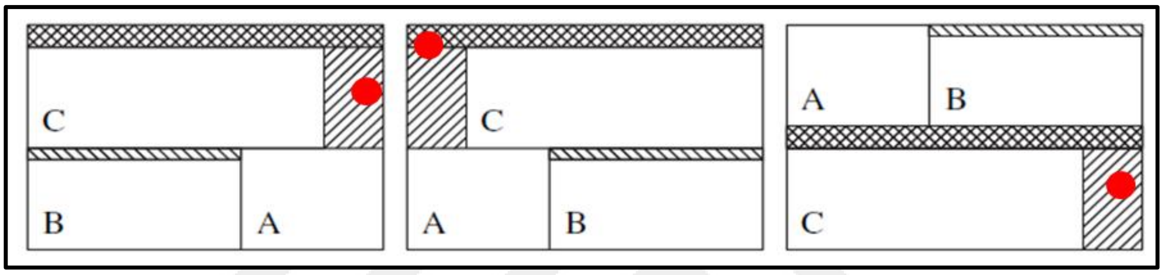


Figure 6.9. Different orientations of a feasible configuration

6.5. BUN SPLITTING

In [8], Glass and Oostrum study a defective cutting stock problem in cake manufacturing. A set of buns are baked in a fixed-sized rectangular tray and then cut into smaller pieces consisting of multiple buns by guillotine cuts for packaging. Figure 6.10 and Figure 6.11 display two alternative cutting schemes for cutting a tray of size 9 x 6 into packages of size 3 x 2.

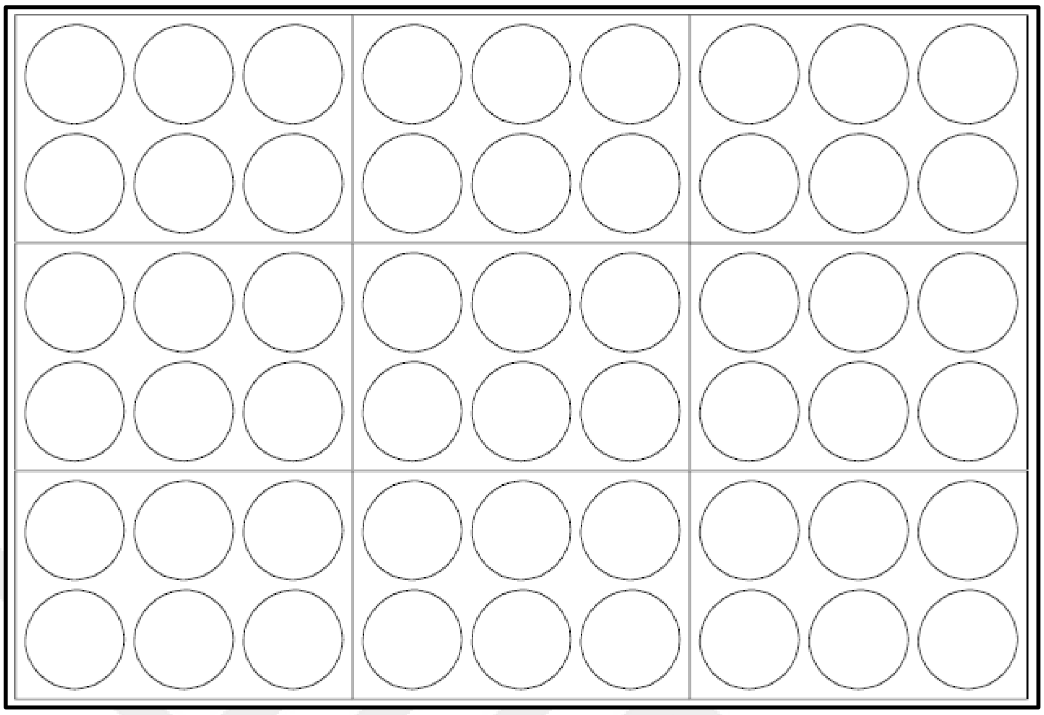


Figure 6.10. Bun splitting (one of the basic cutting schemes) [8]

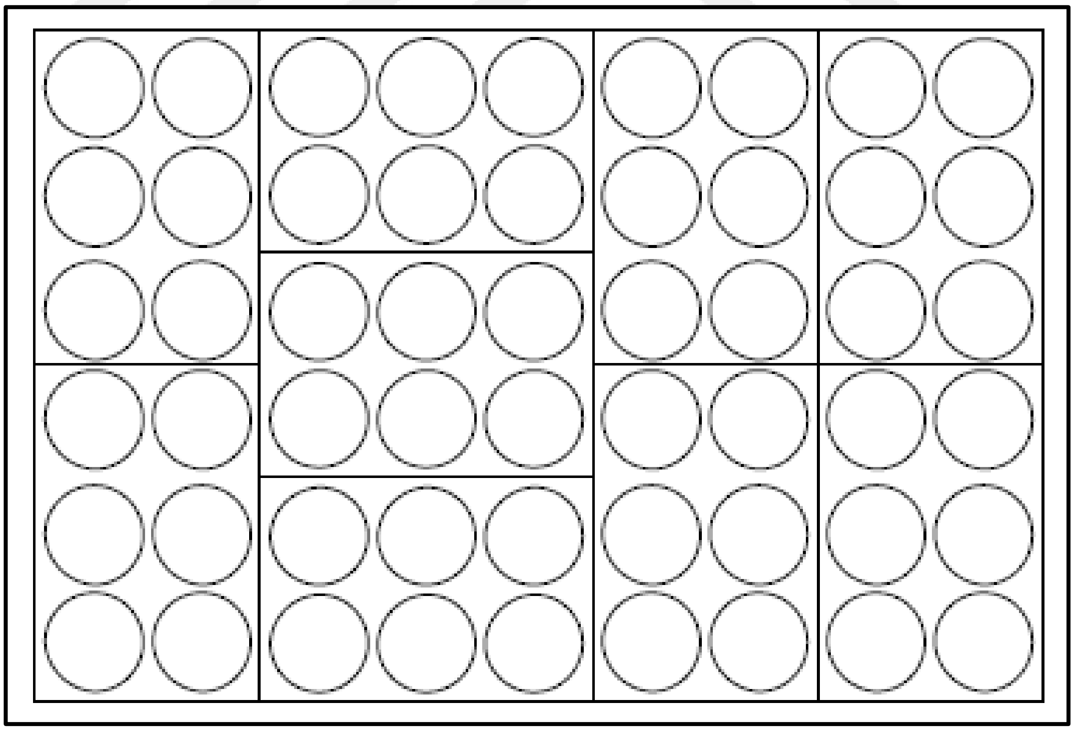


Figure 6.11. Bun splitting (another basic cutting scheme) [8]

However some of the buns are burnt or misformed during the baking process. (To simplify the problem, the authors assume that there can be only one burnt cake on a tray.) These buns are considered as defective and should be thrown away after the splitting stage. Therefore, a second cutting operation is needed after splitting to remove the defective bun as shown in Figure 6.12. In this figure, two 2 x 3 packages with a single defective bun are displayed. Secondary cuts are applied to each 2 x 3 package to create a different set of smaller pieces: the first one results in one 4-bun piece and a 1-bun piece, whereas the second one creates two 2-bun pieces and a 1-bun piece. At the packaging stage, it is necessary to combine these smaller pieces into a 2 x 3 packages (For instance, a 4-bun piece can be combined with a 2-bun piece to form a package). Thus the problem has an online nature where the pieces that are to be cut should be decided based on a continuously changing set of small pieces (consisting of 1, 2, 3 or 4 buns) available for combining.

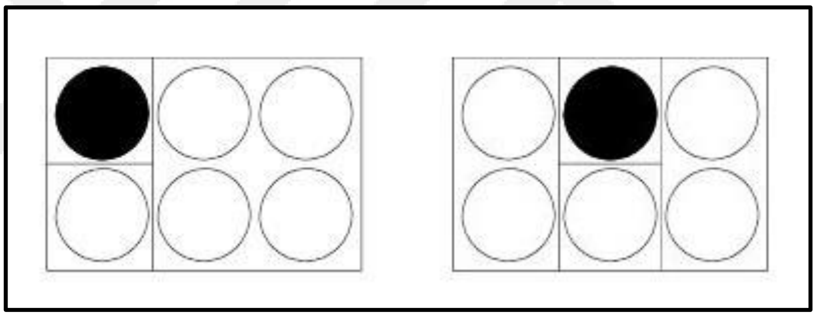


Figure 6.12. Removal of burnt or misformed buns (sub-cutting schemes) [8]

The problem is an online problem with the one studied in this paper. Our methodology can be applied to cake manufacturing with minor modifications. Moreover, using our methodology, we can solve a more general version of the problem in which there can be more than one burnt cake on a tray; which is much more realistic than the single defect per tray assumption in [8]. The authors of [8] also acknowledge that there may be more than one defective bun on a tray. Furthermore, since two (or more) burnt buns are likely to be close to each other, it is possible to have more than one burnt bun even on a 2 x 3 piece to be processed in the second stage. Using the algorithm proposed in this thesis, we can remove this assumption and generalize the problem to one with multiple defects. In order to solve the problem using our approach, we can assume that we solve each tray as a separate SCP during the online algorithm. We can reduce the two dimensional cutting problem for each tray to a one dimensional one by defining configurations along the longer side of the rectangular tray (the x-axis). Each configuration defines a pattern consisting of (possibly partial) packages along the other axis (the y-axis). Depending on its position along the x-axis and the position of the defective bun(s) on the tray, each configuration has a value based on the type of pieces that can be cut from it. The algorithm can then find the best permutation of configurations that match the available pieces to form 2 x 3 packages.

6.6. A DECAYED WOOD CUTTING PROBLEM WITH NON-GUILLOTINE CUTS

[25] suggests a genetic algorithm for a problem from the wood industry. The problem is not a continuous problem and requires cutting stocks in order to produce products with various dimensions. No defect types or quality classes are used; all defective areas should be removed. Although the problem seems similar to ours, we cannot use our methodology to solve it. The reason is that we are using guillotine cuts in all of our solution methods, however cuts are not restricted to guillotine ones in this problem. Figure 6.13 shows a feasible solution for this problem where the shaded area represents a defective region. One can see that solution in Figure 6.13 cannot be produced by any method that uses only guillotine cuts. This prevents the problem from being reduced to one dimension, which is essential for our approach.

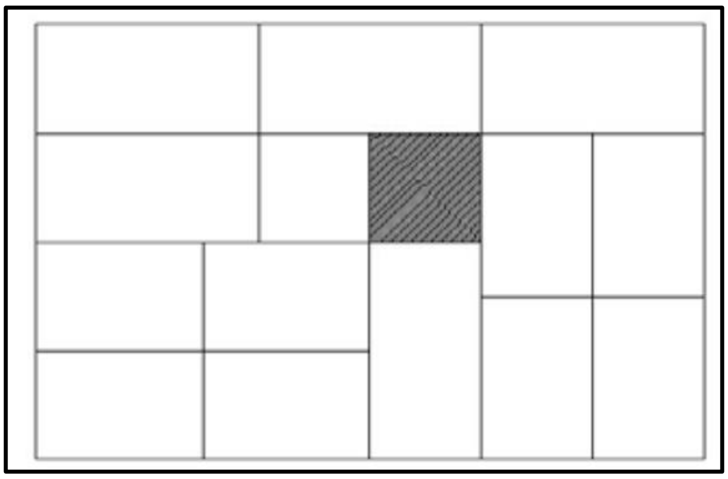


Figure 6.13. A feasible solution for decayed wood board cutting problem [25]

6.7. A GENERIC TWO-DIMENSIONAL CUTTING PROBLEM WITH ARBITRARY CUT POSITIONS ALONG BOTH AXES

[26] studies a two dimensional cutting problem that can be encountered in various industries. The problem is a static problem with unlimited product demands. All of the defective areas should be removed, hence no defect types or quality classes are defined. The authors suggest a dynamic programming based heuristic to solve the problem. Unlike our problem, it focuses on removal of defective areas as shown in Figure 6.14 and more importantly non-guillotine cuts are allowed in this removal process.

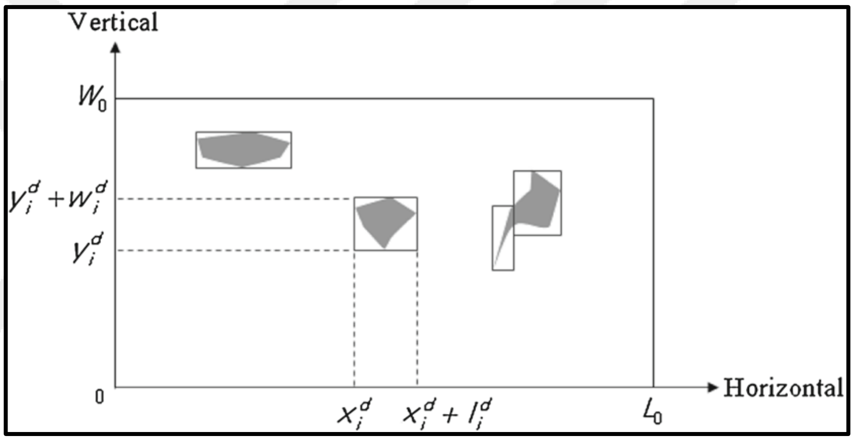


Figure 6.14. Removal of defective regions [26]

One of the main assumptions of our methodology is the fixed positions of the y-cutting wheels, which allow us to transform the problem from two dimensions to one dimension. In this problem, cutting decisions are possible on any point along both x and y axes. This flexibility creates infinitely many possible configurations, which prevents us from using our methodology.

6.8. LOG BUCKING AND LUMBER MANUFACTURING

The problem in [4] is a static cutting problem from the wood industry. Each product has specific dimensions and a quality class and value of a product depend on both of these two features. The objective is to maximize the total value of products produced by cutting the tree. The authors suggest a dynamic programming based method to solve the problem.

There are two important differences between our problem and the problem studied in their paper. First, their stocks are not rectangular as shown in Figure 6.15.

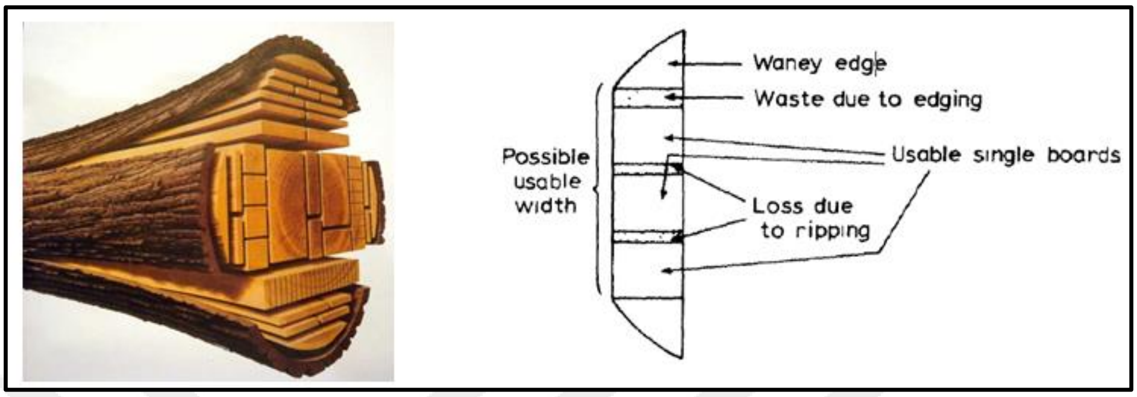


Figure 6.15. Stocks used in log bucking and lumber manufacturing [4]

Second, they have the option of cutting the tree at different angles (see Figure 6.16). These two differences prevent the problem from being reduced to a single dimension. Therefore our algorithm is not applicable to this problem.

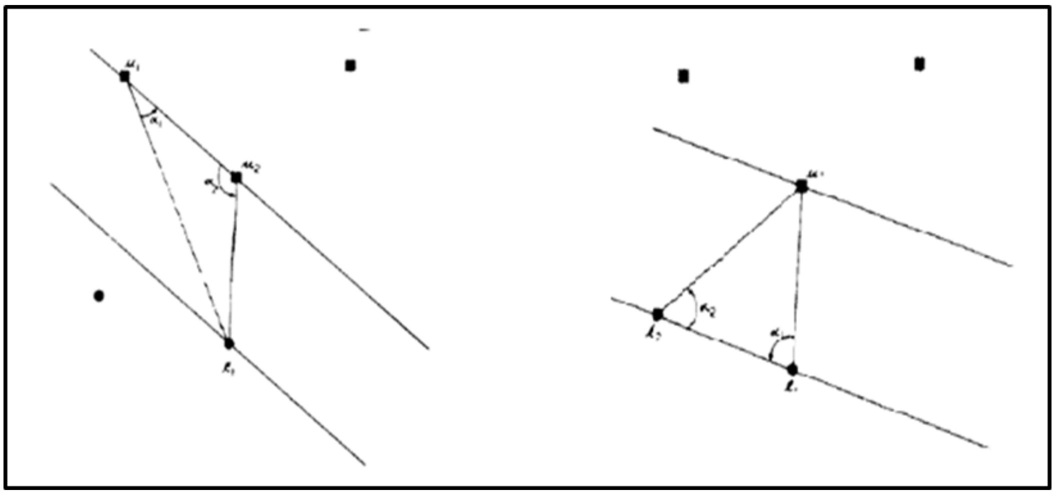


Figure 6.16. Two cutting options of the same stock with different cutting angles [4]

7. CONCLUSION AND FUTURE WORK

In this thesis, we developed an algorithm for the online glass cutting problem that arises in flat glass production. Problem contains defect points of varying severity and product types with different quality classes. Glass sheet is cut horizontally and vertically with guillotine cuts to produce rectangular products. The goal of the problem is to place products on the glass sheet in order to minimize the total scrap glass area and maximize the total value of produced products. Since the problem is an online problem, computational time is the most significant restriction for making a cutting decision: the decision maker has just a few seconds to determine the next product to cut.

We developed a solution algorithm that decomposes the online problem into a series of static cutting problems with a fixed length. In each iteration, the algorithm solves one static problem and implements only a small portion of it. This prevents the decision maker from making myopic decisions.

In order to solve static problems in each iteration, two different approaches can be used. In the first approach where the scrap cuts are limited to CBF, we propose a genetic algorithm (GA) and a dynamic programming (DP) algorithm. On the other hand, when scrap cuts can be made at any arbitrary position, these two solution methods can no longer be used. For this version, we use a MIP model with a preprocessing algorithm and two heuristics, one for performance improvement and another as a fall-back back heuristic to be used in case the MIP does not yield a feasible solution.

In the initial versions of all methods, production targets are assumed to be unlimited. However in practical usage, production targets are one of the main restrictions of a cutting problem. Therefore, we developed new versions of all methods in order to consider production targets. First, we included production targets into MIP model as constraints. Similarly, GA and DP methods accept the product as scrap glass if its production target has already been achieved. In our experiments, we noticed that early completion of any product reduces solution quality significantly in the latter parts of the glass sheet. Thus, in order to balance the production distribution according to production targets, we added a new term into the objective functions of all methods with a coefficient, α.

After we completed the experiments with the new versions, we observed that the suitable values of α depend on the current defect density and the distribution of unfilled production targets. Both values change continuously during the production process. Therefore, using a fixed value for α is not the best way to balance the production distribution. These results inspired us to create the final version of MIP, the adaptive algorithm. This adaptive algorithm solves the problem in parallel with two different α values. The first α value is the one currently used in production, and the second α value is used for a parallel test run. If the second α produces better results, then the algorithm updates the first α with the second one. Otherwise, the first α continues to be used in production and a new α value is tested as a challenger.

7.1. SUMMARY OF FINDINGS

In this thesis, we developed an online cutting problem with defects that has practical significance for the glass industry. The algorithm also provides a general framework for online cutting problems that arise in other industries. Especially the static sub problem solved at each iteration of the algorithm is general enough to adapt to cutting problems with defects encountered in other industries.

Our study starts with two methods for solving static problems, GA and DP methods, both of which use CBF rule. First results show that, both methods have better performance than EE, which is a common practice in the glass industry. Even if GA has better solution quality than DP in low defect density and 6m of static problem length, DP dominates GA in all other cases. On the other hand, computational time of DP highly depends on the defect density and the static problem length, but we cannot assign a time limitation to DP. This poses the risk of not producing a result within the allowed time limit. On the contrary, GA always produces a result within a given time interval. However, when the static problem length is high, solution quality reduces significantly. To sum up, under the CBF rule, we recommend using DP due to its superior solution quality but limiting the static problem length at a conservative value to minimize the risk of not terminating within the allowed time. Even though the risk is minimal, it is always advisable to have a back-up plan in case DP does not yield a solution in time. In fact, in a production environment it would be a good idea to run GA as a back-up procedure in parallel with DP and the solutions generated by GA can be used whenever DP fails to terminate.

In our studies, we questioned the optimality of the results obtained under the CBF rule. We have provided two counterexamples which prove suboptimality of the CBF rule. In both examples, we were able to find a better result than the one produced by DP. These cases prove that CBF rule provides suboptimal results rather than optimal solutions. These two counterexamples inspire us to create another algorithm that does not rely on the CBF rule.

The MIP model which eliminates the CBF rule is used in conjunction with a fall-back heuristic so that it can guarantee to generate a solution within the allowed time. However using suboptimal solution reduces the solution quality. We showed that high optimal solution ratio can be achieved by reducing the problem size of MIP. By eliminating some of the noncritical configurations, we were able to reduce the problem size. Our experiments showed that 7 per cent improvement on average is possible when the number of configurations is limited.

In the production environment, each product has a different demand and priority. During production, cutting decisions also depend on these parameters. At first, we used production targets as constraints in our algorithm. Then, we included them into the objective function with α coefficient in order to balance the production distribution. Including production targets into the objective function provides 3 per cent improvement in all methods on the average. We also experimented with limiting the number of configurations in the MIP model in this new version and obtained an additional 4.4 per cent improvement in solution quality. When production targets are incorporated into the algorithm, we observed that the MIP model yields 7.5 and 5.5 per cent better results compared to GA and DP.

In our final set of experiments, we have shown that balance production quantities of different products improves the solution quality. In order to provide a balanced production distribution, we used an additional term with α coefficient in objective function. However, α value was a fixed number. We developed an adaptive algorithm that updates this coefficient during production so that the algorithm can adapt itself to changing conditions such as changes in defect density or a filling of production targets. Experiments showed that a 1.5 per cent improvement in solution quality is possible by using the adaptive algorithm compared to the fixed α version. In addition, starting with a good initial α value improves the solution by 1 per cent more. The results indicate that the adaptive algorithm can be used without manual intervention to yield comparable and even superior results than the manually optimized α values and demonstrate that it can be used autonomously without the need for an operator.

As explained earlier, the scrap glass resulting from the production process is sent to recycle step and heated again to produce molten glass. This reheating process causes additional energy consumption. Moreover, the scanning and cutting processes are performed again, which further increases the production costs and reduces productivity. Therefore, reduction of scrap glass area is the top priority for float lines. In our conversations with em-glass, we were informed that even a 1 per cent improvement has a significant effect on profitability the profitability of a flat glass manufacturer. In this thesis, we have shown that more than 30 per cent improvement can be achieved by using our adaptive algorithm instead of the explicit enumeration approach currently adopted in the industry. When implemented on a production line, this 30 per cent improvement has the potential to translate into considerable savings.

7.2. CONTRIBUTIONS OF THIS THESIS

In this thesis, we focused on a cutting problem that is solved in real time in float lines. Contributions of our thesis to the literature are summarized below:

- New framework applicable to many online problems.

The glass cutting problem studied in this thesis is an online problem. We transform it to many smaller static problems that are required to be solved within a limited time. However, we implement only a small portion of the solution in order to avoid myopic decisions. The remaining part of the solution is revaluated with new defect information to improve solution quality. This look-ahead approach is a new framework that can be used in different online problems that arise in various industries.

- The problem is a unique problem in terms of different types of defects and various product quality classes.

There are various defect types on the glass sheet. In addition, there are different product quality classes. Each quality class is defined with upper limits of each defect types. Therefore, a product with a particular quality class is valuable only if all types of defects on the product are within the defined limits of that quality class. As a results, the value of a configuration depends on its location on the glass sheet. With all of these characteristics, the problem is a unique problem which has not been studied before. Consequently, our solution methodology does not simply address a simple defect removal process, instead we propose evaluation of all products according to their positions on the glass sheet and their quality classes.

- Solution approach proposed in this study can be generalized to solve other cutting problems.

The approach proposed here can be utilized for many other cutting problems where defects are present. Our study suggests a solution for a problem with multiple defects, which is a general form of the single defect problem. Moreover, the problems where defect removal is necessary can also be solved using our methods. If the definitions of quality classes are updated so that they accept zero defects, our algorithms can be used to create solutions that completely avoid placing products on the defective area. In Chapter 6, we demonstrate that other problems where the stock contains continuous regions of various quality levels or problems in which products can be rotated to avoid defects can also be addressed with the methods described in this thesis. Therefore, although we focus on a specific cutting problem from the glass industry, our approach is flexible enough to be utilized in the solution of many other cutting problems with defects.

- An adaptive algorithm which updates itself in real time is proposed.

In online problems with defects, production is a continuous process. In general, defects do not have a particular pattern and their distribution is not homogenous. Therefore, algorithm can face with various defect densities during the production. Similarly, production targets depend on market conditions and customer demands. These changing parameters affect the performance of the algorithm and the solution quality. In current float lines, production targets and product priorities are entered into the system manually by operators in real time. The adaptive model has the ability to update itself according to conditions, therefore it eliminates the need for human intervention. The novel update mechanism developed in this thesis has the potential to be implemented in other online problem settings with changing conditions.

- The adaptive algorithm can be used for other online multi-objective problems.

The adaptive algorithm proposed in this thesis is designed for a problem with two objectives. The algorithm tries to reduce scrap the glass amount and keeps the production balanced at the same time. In order to adapt to changing conditions, the model continuously checks the weight of the objective function term (α) and updates it as necessary. Updating the weight of production balancing is just one application area of this approach. In general, the model can be used to control and update the weights of an objective function term in any online multi-objective problem. From this perspective, we can view the model as a machine learning tool which observes changing conditions and creates efficient patterns for them in order to increase solution quality.

- CBF rule produces suboptimal results.

GA and DP methods solve the SCP under the CBF rule, which is widely believed to produce superior solution quality in the glass cutting industry. In our MIP model, we removed this rule and had the opportunity to test the optimality of the CBF rule. Using counter examples, we were able to demonstrate the suboptimality of this rule. Moreover, the results of our computational experiments showed us that the CBF rule increases the amount of scrap produced and causes considerable reduction in solution quality.

7.3. SUGGESTIONS FOR FUTURE WORK

In our study, we noticed that similar cutting applications are used in different industries such as paper, textile, metal, lumber, furniture and LCD. The look-ahead approach of our methodology can be adapted to other online cutting problems with defects from these industries. Moreover, as explained in Section 7.2, our approach for solving the static problem is a general solution methodology that can be implemented for various cutting problems. Therefore, these algorithms can be adapted to other static cutting problems with defects.

In this study we focus on glass cutting problems and generalize our solution to other cutting applications in different industries. However, the novel update mechanism used in our adaptive algorithm is quite general and does not have any context specific features. Therefore this approach can be used in other areas where an adaptive behaviour is desired. For instance, online job scheduling problems where scheduling of jobs have to be done according to the changing priorities of the production environment can be a new application area.

Our adaptive algorithm analyses the current data and learns how to update the parameters in order to improve solution quality. To achieve this, it estimates the future results by interpreting the results that have already been produced. With these features, our adaptive approach can also be viewed as a machine learning algorithm. This adaptive approach can be used in any online setting as a machine learning tool that learns the good parameter settings while solving the problem. In particular, our approach provides an effective mechanism that allows the algorithm to set the relative weights of two objectives of a multi-objective problem. In future studies, the adaptive algorithm can be used to solve other online multi-objective problems to update the weights of the objective function terms continuously according to changing conditions. This new approach can be implemented for many other online multi-objective problems to automate the solution process and eliminate manual intervention.

REFERENCES

1. Ochshorn J. Wall sections: Curtain walls and glazing systems, Cornell University ARCH 2614/5614 Lecture notes; [cited 2018 4 April]. Available from: https://courses.cit.cornell.edu/arch262/notes/11b.html.

2. Chauny F, Loulou R. Lp-based method for the multi-sheet cutting stock problem. *INFOR.* 1994;32(4):253–264.

3. Lu HC, Huang YH. An efficient genetic algorithm with a corner space algorithm for a cutting stock problem in the tft-lcd industry. *European Journal of Operational Research.* 2015;246:51–66.

4. Faaland B, Briggs D. Log bucking and lumber manufacturing using dynamic programming. *Management Science.* 1984;30:245–257.

5. Ghodsi R, Sassani F. Real-time optimum sequencing of wood cutting process. *International Journal of Production Research.* 2005;43(6):1127–1141.

6. Aboudi R, Barcia P. Determining cutting stock patterns when defects are present. *Annals of Operations Research.* 1998;82:343–354.

7. Özdamar L. The cutting-wrapping problem in the textile industry: optimal overlap of fabric lengths and defects for maximizing return based on quality. *International Journal of Production Research.* 2000;38(6):1287–1309.

8. Glass CA, van Oostrom JM. Bun splitting: a practical cutting stock problem. *Annals of Operations Research.* 2010;179:15–33.

9. Fathi Y, Kianfar K. An efficient model for the crosscut optimisation problem in a wood processing mill. *International Journal of Production Research.* 2012;50(2):485–497.

10. Dowsland KA, Dowsland WB. Packing problems. *European Journal of Operational Research.* 1992;56:2–14.

11. Dyckhoff H. A typology of cutting and packing problems. *European Journal of Operational Research.* 1990;44:145-159.

12. Bischoff EE, Wascher G. Cutting and packing: special issue. *European Journal of Operational Research.* 1995;84:503–505.

13. Trigos F, López EM. A vulcanising decision planning as a particular one-dimensional cutting stock problem with limited part-related tooling in make-to-order industrial environments. *International Journal of Production Research*. 2017;55(10):2881–2896.

14. Kim BI, Ki Y, Son D, Bae B, Park JS. An algorithm for a cutting problem in window frame production. *International Journal of Production Research.* 2016;54(14):4327–4339.

15. Huang YS, Wang SA, Fang CC. Optimization of the lcd optical film cutting problem. *International Journal of Production Research.* 2017;55(15):4411–4435.

16. Wäsher G, Haussner H, Schumann H. An improved topology of cutting and packing problems. *European Journal of Operational Research.* 2007;183:1109-1130.

17. Gilmore PC, Gomory RE. Multistage cutting stock problems of two or more dimensions. *Operations Research.* 1965;13:94–120.

18. Hahn SG. On the optimal cutting of defective sheets. *Operations Research.* 1968;16:1100–1114.

19. Gilmore PC, Gomory RE. The theory and computation of knapsack functions. *Operations Research.* 1966;14:1045–1074.

20. Russo M, Sforza A, Sterle C. An improvement of the knapsack function based algorithm of gilmore and gomory for the unconstrained two-dimensional guillotine cutting problem. *International Journal of Production Economics.* 2013;145:451–462.

21. Russo M, Sforza A, Sterle C. An exact dynamic programming algorithm for largescale unconstrained two-dimensional guillotine cutting problems. *Computers & Operations Research.* 2014;50:97–114.

22. Herz JC. Recursive computing procedure for two-dimensional stock cutting. *IBM Journal of Research and Development.* 1972;16:462–469.

23. Gelder ER, Wagelmans APM. The two-dimensional cutting stock problem within the roller blind production process. *Statistica Neerlandica.* 2009;63(4):474–489.

24. Twisselmann U. Cutting rectangles avoiding rectangular defects. *Applied Mathematics Letters*. 1999;12:135–138.

25. Wenshu L, Dan M, Jinzhuo W. Study on cutting stock optimization for decayed wood board based on genetic algorithm. *The Open Automation and Control Systems Journal*. 2015;7:284–289.

26. Afsharian M, Niknejad A, Washer G. A heuristic, dynamic programming-based approach for a two-dimensional cutting problem with defects. *OR Spectrum*. 2014;36(4):971–999.

27. Neidlein V, Vianna ACG, Arenales MN, Wäscher G. Two-dimensional guillotineable-layout cutting problems with a single defect - an and/or-graph approach. *In: Fleischmann B, Borgwardt K-H, Klein R, Tuma A, editors. Operations Research Proceedings 2008 Part 3,* Berlin-Heidelberg. 2009;85–90.

28. Sculli D. A stochastic cutting stock procedure: cutting rolls of insulating tape. *Management Science.* 1981;27(8):946–952.

29. Sarker BR. An optimal solution for one-dimensional slitting problems: a dynamic programming approach. *The Journal of the Operational Research Society*. 1988;39:749–755.

30. Sweeney P, Haessler RW. One-dimensional cutting stock decisions for rolls with multiple quality grades. *European Journal of Operational Research.* 1990;44:224–231.

31. Aksu DT, Durak B. A dynamic programming algorithm for the online cutting problem with defects and quality grades. *IFAC-PapersOnLine.* 2016;49(12):17–22.

32. Astrand E, Rönnqvist M. Cross cut optimization of wooden boards based on automatic defect detection. *Operational Research Society of New Zealand Proceedings*. 1993;268–275.

33. Durak B, Aksu DT. A genetic algorithm based solution for the online cutting problem with defects. *Proceedings of the International Conference on Industrial Engineering and Engineering Management, 2015 IEEM,* Singapore. 2015;1805 - 1809.

34. Durak B, Aksu DT. Dynamic programming and mixed integer programming based algorithms for the online glass cutting problem with defects and production targets. *International Journal of Production Research*. 2017;55(9):1-14.