THE CUTTING STOCK PROBLEM WITH DIAMETER CONVERSION IN THE CONSTRUCTION INDUSTRY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DENİZ ALTINPULLUK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2019

Approval of the thesis:

THE CUTTING STOCK PROBLEM WITH DIAMETER CONVERSION IN THE CONSTRUCTION INDUSTRY

submitted by **DENİZ ALTINPULLUK** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _______________

Prof. Dr. Haldun Süral
Supervisor, **Industrial Engineering, METU** _______________

Examining Committee Members:

Assoc. Prof. Dr. Canan Sepil
Industrial Engineering, METU _______________

Prof. Dr. Haldun Süral
Industrial Engineering, METU _______________

Assoc. Prof. Dr. Sedef Meral
Industrial Engineering, METU _______________

Assoc. Prof. Dr. Ferda Can Çetinkaya
Industrial Engineering, Çankaya University _______________

Assist. Prof. Dr. Sakine Batun
Industrial Engineering, METU _______________

Date:

ABSTRACT

THE CUTTING STOCK PROBLEM WITH DIAMETER CONVERSION IN THE CONSTRUCTION INDUSTRY

Altınpulluk, Deniz
M.S., Department of Industrial Engineering
Supervisor: Prof. Dr. Haldun Süral

July 2019, 85 pages

The one-dimensional cutting stock problem has been widely used for reinforcement steel bar (rebar) in the construction industry. Diameter sizes of rebars are determined by the structural designer to provide tensile strength to the structure, and they can be changed if the ratio of cross-section area of rebars to the concrete area stays constant. The decision maker can decide to convert diameter size to generate better cutting patterns. Besides, the time dimension is considered by assuming the multi-period structure of rebar demand. We study the cutting stock problem that decides diameter sizes, cutting times, and cutting patterns to minimize the usage of raw material and holding cost. It differs from the classical cutting stock problem as it is not known how many pieces should be cut until diameter sizes are chosen. We propose a pseudo-polynomial formulation and a genetic algorithm to solve the problem. Computational results are provided.

Keywords: construction industry, cutting stock, lot-sizing, genetic algorithm

ÖZ

İNŞAAT SEKTÖRÜNDE ÇAP TAHVİLLİ STOK KESME PROBLEMİ

Altınpulluk, Deniz
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Haldun Süral

Temmuz 2019 , 85 sayfa

Bir boyutlu çubuk kesme problemi inşaat sektöründeki betonarme demirlerinin kesimi için sıklıkla uygulanmıştır. Yapı tasarımcısı gerekli çekme mukavemetini sağlayabilmeleri için kullanılan demirlerin çap büyüklüklerini belirler. Çap büyüklükleri, kullanılan betonarme demiri sayısı değiştirilerek betonun kesit alanındaki demirin alanı sabit kalmak şartıyla farklı büyüklüklere dönüştürülebilmektedir. Karar verici çapların büyüklüklerini birbirlerine tahvil ederek daha iyi kesme kalıpları oluşturabilir. Demir taleplerinin farklı dönemlerde olabileceğini varsayarak, zaman boyutu da hesaba katılmıştır. Malzeme kullanımını ve envanter tutma maliyetini enazlayacak şekilde çap büyüklüklerine, kesim zamanlarına ve kesme kalıplarına karar verilen bir stok kesme problemi üzerinde çalışılmıştır. Bu problemde ne kadar parça kesileceği, çap büyüklüğü belirlenmediği için klasik stok kesme probleminden farklılaşmaktadır. Çözüm önerisi olarak pseudo-polinomiyal bir formulasyon ve genetik algoritma yaklaşımı önerilmektedir. Sayısal sonuçlar verilmiştir.

Anahtar Kelimeler: inşaat sektörü,stok kesme, kafile büyüklüğü, genetik algoritma

To Mom and Dad...

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. Dr. Haldun Süral for his support during my MS study. He helped me to learn how to conduct a research and motivated me to improve my study.

I would like to thank the examining committee members of my thesis Assoc. Prof. Dr. Ferda Can Çetinkaya, Assoc. Prof. Dr. Canan Sepil, Assoc. Prof. Dr. Sedef Meral, and Assist. Prof. Dr. Sakine Batun for their feedbacks about my research study.

I would like to thank my family members, Habip Altınpulluk, Yıldız Altınpulluk, and Özgür Altınpulluk, for their great support during my life. I feel very lucky for having such a perfect family. I wish to thank Nur Banu Demir for her support throughout my MS thesis. She was always patient and helpful.

I would like to thank Cem İlhan being a great inspiration for this study. My friends Ali Naif İnceefe, Baran Kaya, Esin Özden Doğan, and Ahmet Sencer Koyuncu helped me a lot to understand the basics of the construction industry and provided real data to conduct computational experiments. Without their support, it would be a lot harder to define my problem based on the application area.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vi

ACKNOWLEDGMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xii

LIST OF FIGURES xiv

LIST OF ABBREVIATIONS xvi

CHAPTERS

1 INTRODUCTION 1

2 LITERATURE REVIEW 5

2.1 Cutting Stock Problems 5

2.1.1 Open Dimension Problems 7

2.1.2 Waste Minimization Problems for Reinforcement Steel Bar . . 11

2.1.3 Cutting Stock Problems with Time Dimension 12

2.2 Solution Methods for 1-Dimensional Cutting Stock Problems 13

2.2.1 Formulations 13

2.2.2 Exact Methods 14

2.2.3 Heuristic and Metaheuristic Approaches 16

3 PROBLEM DEFINITION 19

3.1 Single-Period Cutting Stock Problem with Diameter Conversion in the Construction Industry 19

3.1.1 Problem Formulation 19

3.1.2 Mathematical Formulations 24

3.1.2.1 Kantorovich Formulation 24

3.1.2.2 Arc-Flow Formulation 26

3.1.2.3 Reflect Formulation 28

3.2 Multi-Period Cutting Stock Problem with Diameter Conversion in Construction Industry 31

3.2.1 Problem Formulation 31

3.2.2 Mathematical Formulation 33

3.2.2.1 Kantorovich Formulation for Multi-Period Case 33

3.2.2.2 Reflect Formulation for Multi-Period Case 35

4 SOLUTION APPROACHES 39

4.1 Solution of Mathematical Formulations 39

4.1.1 Kantorovich Formulation 39

4.1.2 Arc-Flow Formulation 42

4.1.3 Reflect Formulation 45

4.2 Genetic Algorithm Approach 48

4.2.1 Representation Scheme 48

4.2.2 Initial Population Generation 49

4.2.3 Fitness Function 52

4.2.4 Parent Selection 54

4.2.5 Crossover Operators 54

4.2.6 Mutation Operators 55

4.2.7 Forming Population for Next Generation 56

4.2.8 Termination Criteria 56

4.2.9 Pseudocode of the Genetic Algorithm 58

5 COMPUTATIONAL RESULTS 59

5.1 Data Collection & Instance Generation 59

5.2 Computational Results for the Single-Period Cutting Stock Problem with Diameter Conversion in the Construction Industry 61

5.3 Computational Results for the Multi-Period Cutting Stock Problem with Diameter Conversion in the Construction Industry 62

5.4 Parameter Selection for Genetic Algorithm 67

5.5 Contributions of 1.5-Dimensional and Multi-Period Cutting Stock Problem 72

6 CONCLUSION 75

REFERENCES 79

APPENDICES

A 83

LIST OF TABLES

TABLES

Table 3.1 Example orders 22

Table 3.2 Conversion table 23

Table 3.3 Notation of the Kantorovich formulation for 1.5-D CSP 24

Table 3.4 Notation of the arc-flow formulation for 1.5-D CSP 27

Table 3.5 Notation of the reflect formulation for 1.5-D CSP 30

Table 3.6 Notation of the Kantorovich formulation for the 1.5-D MPCSP . . . 34

Table 3.7 Notation of the reflect formulation for the 1.5-D MPCSP 36

Table 5.1 Properties of instances 60

Table 5.2 Number of instances used in the experiments 61

Table 5.3 Comparison of reflect and arc-flow formulations for 1.5-D CSP . . . 63

Table 5.4 Comparison of the reflect formulation and genetic algorithm for 1.5-D MPCSP 65

Table 5.5 Genetic algorithm parameters' levels to be enumarated (%) 67

Table 5.6 Average gap values in different parameter settlement 68

Table 5.7 Average gap values in different mutation percentages 69

Table 5.8 Average gap values and run times in different population sizes . . . 69

Table 5.9 1.5-D CSP % gain comparison to 1-D CSP 73

Table 5.10 1.5-D MPCSP % gain comparison to 1-D CSP 74

Table A.1 Example of a small instance for first-fit algorithm 83

Table A.2 Example of first-fit algorithm without improvement 84

Table A.3 Example of first-fit algorithm with improvement 84

LIST OF FIGURES

FIGURES

Figure 1.1 Rebars used in concrete 1

Figure 2.1 Typology of cutting and packing problems [1] 7

Figure 2.2 Coil slitting operation [2] 9

Figure 2.3 Input and ordered sheets for 1.5-D CSP in plastic industry [3] . . 11

Figure 3.1 Rebars used in construction industry 21

Figure 3.2 Example for network representation of the arc-flow formulation [4] 26

Figure 3.3 Example for network representation of the reflect formulation [5] 29

Figure 4.1 Graph generation example 46

Figure 4.2 Chromosome representation 49

Figure 4.3 Example conversions of base heuristic 51

Figure 4.4 Example conversions of base heuristic 51

Figure 4.5 Example of a crossover operation 55

Figure 4.6 Mutation example 56

Figure 5.1 Gap values over time when the number of item is 250 for four different population sizes 70

Figure 5.2 Gap values over time when the number of item is 500 for four different population sizes 71

Figure 5.3 Gap values over time when the number of item is 1000 for four different population sizes 71

Figure 5.4 Gap values over time when the number of item is 2000 for four different population sizes 72

LIST OF ABBREVIATIONS

1-D	1-Dimensional
2-D	2-Dimensional
CSP	Cutting stock problem
1-D CSP	One-dimensional cutting stock problem
1.5-D CSP	1.5-dimensional cutting stock problem
2-D CSP	Two-dimensional cutting stock problem
1.5-D MPCSP	1.5-dimensional multi-period cutting stock problem
B&B	Branch & Bound
B&P	Branch & Price
BPP	Bin packing problem
DP	Dynamic Programming
GA	Genetic Algorithm
GCG	Generic Column Generator
GDP	Gross Domestic Product
LP	Linear Programming
MIRUP	Modified integer round-up property
WCPR	Worst-case performance ratio

CHAPTER 1

INTRODUCTION

The construction industry is one of the leading industries in the world. It constitutes a significant part of the gross domestic products of countries. According to the Turkish Central Bank report [6], construction accounts for 5.7% of Turkey's GDP. If its direct and indirect impacts on other sectors are considered, the share of the construction sector in the Turkish economy reaches 30%. This industry not only aims to construct real estates such as houses, apartments, skyscrapers, and business offices but also includes the construction of public structures such as bridges, roads, airports, etc. Decreasing costs of construction industry contributes to economies of countries considerably. In addition, decreasing the waste amount of materials used in constructions is beneficial to the environment as well. Although developing technologies provide cost reduction in the construction industry significantly, there are always chances to improve the existing system by planning its activities efficiently.

One of the major cost items in the construction industry is reinforcement steel bar (rebar) which is widely used to provide necessary tensile strength to the structures. Figure 1.1 shows the reinforcement steel bars used in concrete. It is crucial to use

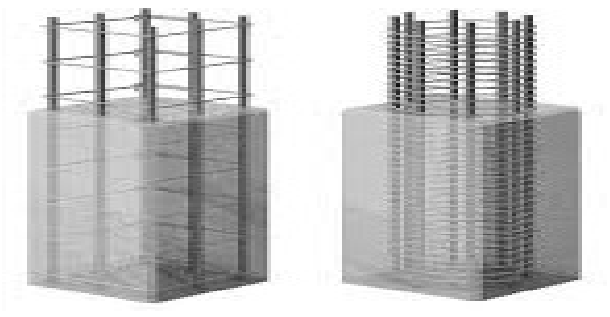


Figure 1.1: Rebars used in concrete

rebars efficiently to decrease the cost of structures and to prevent wasting natural resources. Reinforcement steel bars with standard length are sold in different diameters in the market. These rebars are ordered in different sizes of length depending on the geometric sizes and shapes of the structural elements. Each part of the structures requires different sizes of these rebars in terms of diameter and length, and they can be obtained by cutting large rebars into smaller pieces. The need for an efficient cutting plan arises in order to decrease trim losses for every structure where reinforcement steel bars are used. The one-dimensional cutting stock problem (1-D CSP) appears when cutting patterns are generated so that the usage of raw material is minimized. Suppose that there are m item types of rebars, each of them has length w_j and demand d_j, k different diameter sizes, and a sufficiently large number of stocks (j = 1,2,...,m). Stocks are generally sold with 12 meters in the market. Therefore, stock capacity is 12 meters where several items are located with length of w_j. The objective is to cut d_j copies for each item type j using the minimum number of stocks so that the total length of items for each stock does not exceed the capacity. However, using the 1-D CSP needs to group items whose diameter sizes are equal. It separately solves every group of the problem for each diameter size. Therefore, the number of the 1-D CSP on hand is equal to the number of diameter sizes used in the construction project. However, this approach misses the advantage of convertibility of diameter sizes to each other in designing structural elements.

Diameter sizes and the number of rebars required are determined in a way to obtain necessary tensile strength. The chosen diameter size and the number of rebars needed for each structural element can be converted into different sizes if some conversion rules are followed. These rules are discussed in Chapter 3. Since the diameter sizes are convertible, considering both decisions of choosing diameter sizes and generating cutting patterns together would yield more efficient results to avoid trim loss. Briefly, the problem in this thesis is to determine cutting patterns and diameter sizes to minimize the usage of reinforcement steel bars. Since the diameter is a different dimension of the problem in addition to the length dimension, the problem differs from the classical 1-D CSP. On the other hand, sizes of the diameter dimension are unknown and convertible to different sizes. Therefore, determining the appropriate diameter size is a part of the decision process. It means that diameter sizes are decision variables of the problem and the problem differs from the classical two-dimensional cutting stock problem (2-D CSP).

The 1-D CSP and 2-D CSP have taken particular attention in the literature. However, a few studies were published similar to our problem, and the problem is defined as the 1.5-dimensional cutting stock problem (1.5-D CSP). We propose a pseudo-polynomial arc-flow formulation, particularly the reflect formulation of Delorme and Iori [5] to solve the problem. It is a quite powerful technique even for large size problem instances because it uses the half capacity of the stock size rather than the total capacity. Therefore, it partially handles the weakness of the former pseudo-polynomial formulation of arc-flow [4].

Most construction projects may include more than one building such as multiple unit housing projects, holiday sites, public building complexes, etc. The need for rebars may occur in different time periods during the construction phase of these projects. Especially, one huge building projects may need different diameter rebars in different time periods according to the business plan. As a result, the decision maker of the construction project should decide the times of procurement and amount of rebars before rebars are needed. The aim of the decision maker is to determine the time periods of the cutting operations and cutting patterns of the items so that trim loss and inventory holding costs are minimized. This problem is called as the 1.5-dimensional multi-period cutting stock problem (1.5-D MPCSP). To benefit from the diversity, the decision maker can decide cutting the demands of the latter periods by paying inventory holding costs. Therefore, there is a trade-off between inventory holding cost and trim loss. The problem is to determine cutting patterns, diameter sizes, and cutting periods of each rebar at the same time. To solve the 1.5-D MPCSP, the reflect formulation [5] is proposed for small and moderate size problem instances. However, obtaining exact solutions by using the commercial solver CPLEX is difficult for moderate and large size instances with this formulation. Therefore, we propose a Genetic Algorithm (GA) approach for solving moderate and large size problem instances. The proposed algorithm can reach small optimality gap values within a reasonable amount of time. Moreover, the parameters of the GA are finetuned. We compared the performance of the GA under different parameters in terms of solution quality and computation times.

The computational experiments are based on the real projects in the construction industry such as hospitals, apartments, business center, and public buildings. Our computational experiments question the following issues:

1. To what sizes can the single-period and multi-period 1.5-D cutting stock problems be solved?
2. Do the different properties of the construction projects (such as the number of items, the number of different diameter sizes, etc.) affect the solution qualities and computational times?
3. How does the multi-period structure affect the solution quality compared to the single-period structure?
4. How does a different solution approach affect solution quality and computational time for single-period and multi-period problems?
5. What is the advantage of using the genetic algorithm over the reflect formulation?
6. How does the parameter setting of the genetic algorithm affect the solution qualities and computational times?
7. Are the conversions in the diameter sizes and holding inventory cost beneficial in the industry?

The organization of the next chapters is as follows. In Chapter 2, the studies related to the cutting stock problems and their solution approaches will be discussed. In Chapter 3, the problem definitions, properties of the problems, background information, and mathematical formulations are provided separately for the single-period and multi-period cutting stock problems with diameter conversion in the construction industry. Exact and heuristic solution approaches to solve the 1.5-D CSP and the 1.5-D MPCSP will be presented in Chapter 4. The computational results for each solution approaches are compared in terms of solution quality and computational times and the discussions based on experimental results will be provided in Chapter 5. Finally, the conclusion and further research directions will be shared in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

In this section, we conduct a literature review for the cutting stock problem and its variants. We summarize two published typologies about cutting and packing problems. Furthermore, we classify the studies in two main sections according to their relevances. The first section reviews the cutting stock problems while the second section summarizes solution methods for the cutting stock problems.

2.1 Cutting Stock Problems

In the cutting stock problem, there are m item types, with length w_j and demand d_j (j = 1,2,...,m), and a sufficiently large number of identical stocks with capacity c. The aim is to obtain d_j copies of each item type j using the minimum number of stocks so that the total length of items for each pattern does not exceed the capacity. The cutting stock problem was early formulated by Kantorovich [7]. Although his formulation is weak and solves small size instances, it helps to understand the problem structure. Gilmore and Gomory [8] suggested a concept of column generation for the cutting stock and bin packing problems (BBP) inspiring from Dantzig and Wolfe [9]. This approach brought a breakthrough in solving the cutting stock problems. Since enumerating all feasible cutting patterns is prohibitively time consuming, it generates valid patterns iteratively and adds them to the problem according to their contribution to the objective function. With the help of the column generation method, the large scale cutting stock problems became solvable in a reasonable time. Furthermore, many studies have been conducted in course of time to attack the cutting stock problem (CSP) and its variations.

Dyckhoff [10] classified cutting and packing problems by integrating different kinds of problems according to several classification criteria. Washer et al. [1] developed a new typology partially based on Dyckhoff's ideas. They extended a variety of the problems and introduced new classification criterion which is the shape of the small items. Basically, small items are laid out on large objects. For example, small items refer to orders which are cut from large stocks in the cutting stock problems. The cutting and packing problems can be classified according to five main features: dimensionality, kind of assignment, assortment of small items, assortment of large objects, and shape of small items. Dimensionality distinguishes the problems according to one, two, and three dimensional or even larger dimensions. Kind of assignment divides the problems in terms of their assignment types such as input minimization and output maximization. There are three main categories in assortment of small items such as identical small items, weakly heterogeneous assortment, and strongly heterogeneous assortment. To illustrate, the cutting stock problems can be classified as weakly heterogeneous assortment since small items can be grouped into relatively few classes. On the other hand, the bin packing problems are in the class of strongly heterogeneous assortment because all items differ from each other. Assortment of large objects classifies the problems as one large object and several large objects. The last criterion is the shape of small items in the typology. It distinguishes the shape of small items in terms of their geometric shapes. Basic problem types can be defined by using two criteria which are kind of assignment and assortment of small items. Figure 2.1 shows the basic problem types according to the typology of Washer et al. [1].

Although the exact classification of our problem was not addressed in the typology, we consider our problem in the class of open dimension problems according to this typology. Open dimension problems have been rarely discussed in the literature compared to other problem types. In the class of open dimension problems, small items are accommodated by one or several large objects, but at least one dimension is regarded as a variable of the problem. It means that fixing this variable dimension must be a part of decision processes in solving the problem. It generally occurs when two dimensional small items are laid out on one or more very long rectangular large objects, whose width is fixed and its length is tried to be minimized. In addition, open

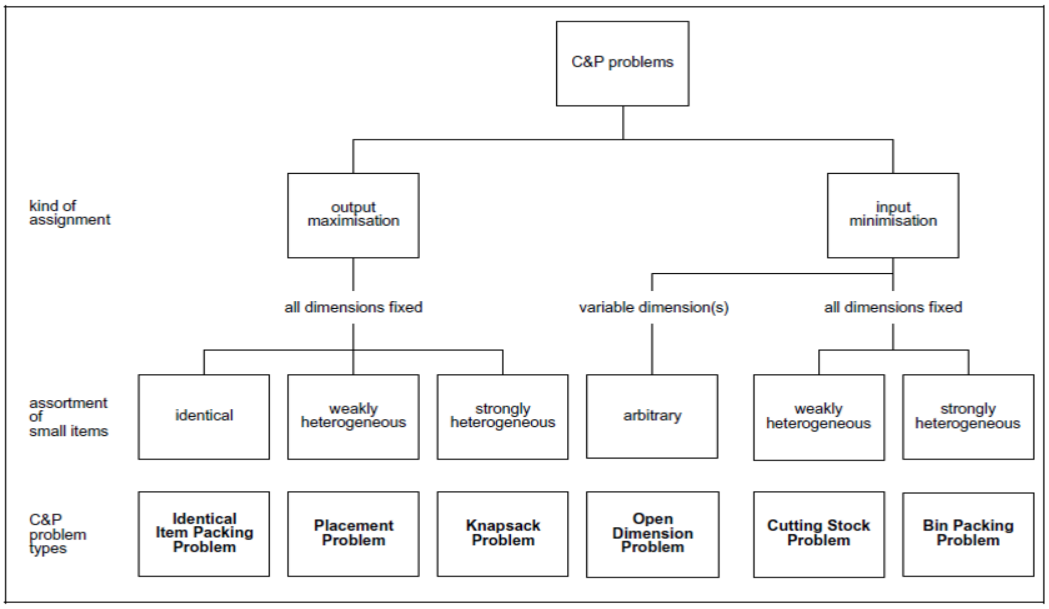


Figure 2.1: Typology of cutting and packing problems [1]

dimension problems appear in the industries that use rolled materials such as paper rolls or coils. Width of the large rolls is generally fixed, but their lengths or diameters vary in distinct sizes. The aim is basically to minimize inputs by determining cutting patterns and selecting appropriate stock sizes.

Considering discussion about the typology of cutting and packing problems, we claim that the problems considered in this thesis can be classified as an input minimization, open dimension with more than one large object, and cutting stock problem. Moreover, this class of problems is also defined as the 1.5-dimensional cutting stock problem (1.5-D CSP) in the literature.

2.1.1 Open Dimension Problems

Haessler [11] studied the 1.5-dimensional cutting stock problem in 1978 and called the problem 1.5-dimensional coil slitting problem that occurred in the metal industry. In classical 1-D CSP, larger stocks are cut into smaller parts to satisfy customer demand. It is desired to generate optimal cutting patterns to minimize the usage of raw material. Similarly, the 2-D CSP tries to achieve the same goal, but orders are specified in two dimensions. In coil slitting problem, coils are rolled on length dimension, and their widths are much smaller than their lengths. Therefore, orders should be obtained by cutting these rolled larger items alongside their long dimension. In Heassler's problem, each order has a particular width and weight, and these orders can be obtained from larger coils in the inventory. Inventory coils also vary in width and weight. Figure 2.2 depicts the inventory coils, order coils, and slitting operation. However, demand requirements for one dimension are specified indirectly. To clarify, demand is stated in terms of the total weights and desired width instead of specifying the length, width, and required number as being in the classical 2-D CSP. Demand can be satisfied by summing up the weights of small items. Since there are multiple coils in the inventory, it is not known which order should be satisfied from which of them. Therefore, the number of pieces needed for satisfying demand for an item is not known until the sizes of inventory coils, which will be cut, have been selected. Coils in the inventory may vary according to their weights because the diameter of coils may change. Thus, the number of required coils to satisfy demand may vary according to the selection of the coils.

In Haessler's problem [11], orders are defined by specifying number, widths, and weights of coils. To formulate he used pounds and widths. Parts are cut according to the width and brought together in order to satisfy weight per inches of width. To satisfy weight per inches of width for a specific order, weights of some equal width parts should be summed up, and the total weight of the parts should be between the upper and lower limits of the order for weight per inches of width.

Haessler's problem [11] is similar to the problem to be considered in this thesis study in terms of several aspects. First, both problems have orders whose lengths (widths) are specified clearly but the size of the second dimension is denoted indirectly. While the measure of the second dimension is the total area of rebars in our problem, the total weight per inches of width is the measure of order in Haessler's problem. Hence, the diameter of rebars and weights of coils can be changed as long as the total requirements specified for both problems satisfied. Second, the number of pieces that must be cut is not known until which stocks should be cut that are chosen for both problems. In our problem, diameter size can be changed by decision maker within an allowable range provided that the ratio of rebar cross-section area to the concrete area remains constant. Therefore, the number of pieces to be cut is uncertain until

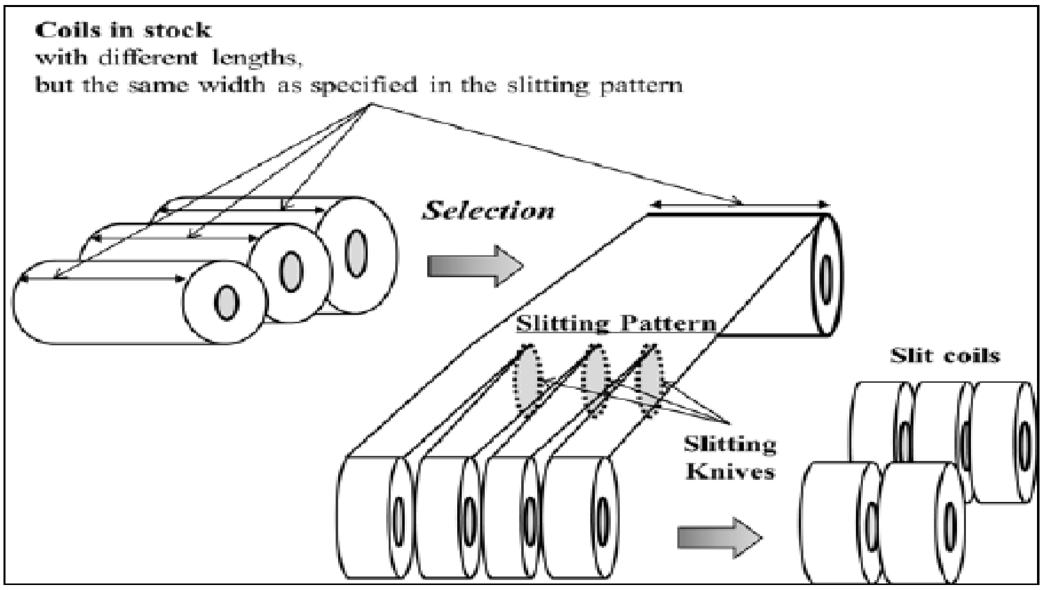


Figure 2.2: Coil slitting operation [2]

diameter size is chosen for each structural element. Similarly, weights of larger coils vary in different sizes. The number of required coils, that satisfy the order, is not known unless inventory coils selected.

There are some differences between these two problems. Firstly, rebars with different diameters cannot be brought together to satisfy area requirements apart from Haessler's case. To clarify, the total weight requirement of a particular item can be satisfied from several large objects by summing up weights of small items whose width are same in coil slitting problem. However, only one type of diameter should be chosen to satisfy the demand of each structural element in our problem. It is not desired to satisfy demand by using rebars with more than one kind of diameter in one structural element because of the structural rules and operational simplicity. In addition, diameter conversion is also restricted by several rules. It can be made for an order of structural element within an allowable range, generally, two steps up and down. However, an order of the coils could be met from the multiple coils in the inventory whose weight may vary. It means that two or more coils in the inventory can be used to satisfy particular order as long as the requirement of weight per inches of width is satisfied. Secondly, coils in the inventory may have different widths, but the rebars are cut from a constant 12 meters stock length. Therefore, the coil slitting problem has multiple stock length nature.

The coil slitting problem can be classified as an input minimization, open dimension with more than one large object, and the cutting stock problem with multiple stock sizes according to the typology of Washer et al [1].

Han and Chang [2] also studied the coil slitting problem. They tried to minimize slitting loss and overproduction by selecting appropriate large objects when the cutting patterns are given. Apart from Haesler, they attempted to deal with overproduction penalty. They developed a pseudo-polynomial time algorithm to attack the problem. However, they only dealt with the selection of coils in the inventory by not considering pattern generation.

Gasimov et al. [12] studied the 1.5-dimensional assortment problem in the production of corrugated boxes in the paper industry. Orders are cut from paper rolls in a rectangular shape. Their large objects are sufficiently long sheets assuming infinite for practical purposes. Moreover, their problem has multiple stock size nature. There are several stock sizes available in the market and it is beneficial to keep different sizes of stock in the inventory in order to achieve the lower trim loss level. Nevertheless, increasing the number of different roll sizes causes several types of costs to be incurred such as costs of operation, stocking, and handling. Therefore, there is a trade-off between inventory costs and material utilization in terms of trim loss. The problem is also an open dimension cutting stock problem with strongly heterogeneous assortment of large objects. They proposed the multi-objective mixed integer linear programming model. They generated all possible cutting patterns for the formulation.

Song et al. [3] studied a real-life 1.5-dimensional cutting stock problem that appears in the plastic industry. The problem is the selection of a subset from the set of stock rectangles to minimize the total production cost. It includes waste of material and production time under limitations of cutter knife changes, machine restrictions, and due dates. There are the number of available input sheets whose lengths vary in length dimension, and their widths are the same. They are desired to cut along its width, but input sheets are smaller than the orders as shown in Figure 2.3. Therefore, demand can be met by getting together more than one input sheets to obtain the necessary

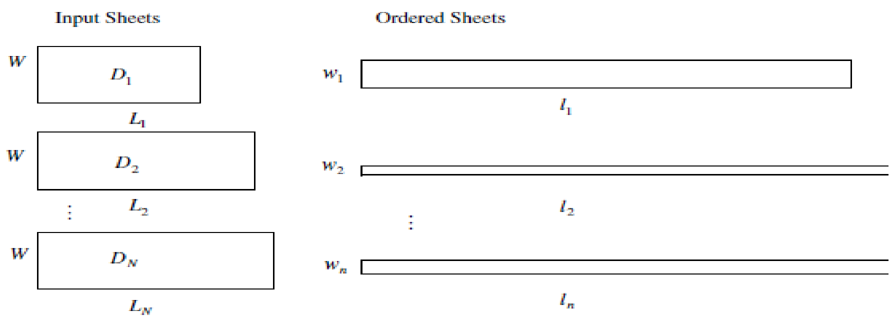


Figure 2.3: Input and ordered sheets for 1.5-D CSP in plastic industry [3]

length and cutting them into the desired width. This problem is also similar to our problem because the number of pieces that will be cut is not known until the input sheets are chosen. Besides, the width of sheets is the same for all input sheets similar to our problem. However, demand can be met from any input material by bringing them together to satisfy the length requirement apart from our problem. In our case, demand can be met by bringing rebars together to satisfy area requirements as in their problem, but rebars should be from the same diameter for each structural element. In addition, the diameter of rebars should also be within an allowable range.

2.1.2 Waste Minimization Problems for Reinforcement Steel Bar

Reducing waste of reinforcement steel bar has taken particular attention in the literature since it is one of the major cost items in the construction industry. The 1-D CSP occurs while generating cutting patterns to satisfy rebar requirements. On the other hand, several designing and managerial issues can be combined with CSP. A recent study published by Nadoushani et al. [13] dealt with minimizing cutting waste of reinforcing steel bars by considering designing issue, lap splicing, which affects ordered lengths of reinforcement steel bar. Since there is flexibility in terms of length of rebars for some particular structural elements, they proposed to take into account this flexibility while generating cutting patterns in order to achieve better utilization level of materials. In other words, lengths of some ordered rebars are changed within the range that structural design code allows. This study also has similarities with our problem because changing the design of structures leads to better utilization of material. However, they dealt with changing the length of the rebars rather than diameter as in our problem. Another recent study published by Zheng et al. [14] considered rebar material costs related to trim loss and rebar installation costs including labor hours used in rebar stock processing, delivering, placing, and tying. These installation costs are directly dependent on rebar layout arrangement plan. Authors claim that benefits can be obtained from a trade-off between reducing waste and lowering the total cost by identifying the rebar layout arrangement plan and generating the rebar procurement plan, cutting plan, and crew installation plan. Benjaoran and Metham [15] studied the effect of demand variations on steel bars cutting loss and experimentally showed how the distribution of pieces of length ordered affects material utilization. Although there are several studies published in the journals related to the construction industry to attack the 1-D CSP for reinforcement steel bar, we reviewed solution methods to solve 1-D CSP in the last section in detail.

2.1.3 Cutting Stock Problems with Time Dimension

In this subsection, we review the cutting stock problem related to the time dimension. There are some articles addressing the time dimension of the cutting stock problem in the literature. Time dimension appears as a significant factor that affects the costs of production and operations. Since there are several cost components such as setup costs, holding costs, and tardiness costs, it is beneficial to take time dimension into account for the cutting stock problems. The main idea behind the time dimension of cutting and packing problem is to exploit the trade-offs between trim loss and costs related to time. Recently, Poldi and Araujo [16] considered the multi-period one-dimensional cutting stock problem with holding costs and trim loss and adopted an arc-flow model with the multi-period structure. Reinertsen and Vossen [17] studied the 1-D CSP with due dates which arises in supplying of reinforcement steel bars to customers. It tries to minimize total tardiness costs and trim loss simultaneously. Sometimes, tardiness may be more important than trim loss. In addition, there are a lot of studies in the literature which dealt with setup and holding costs in the cutting stock problems. We refer to the review paper recently published by Melega et al. [18], which classifies the lot-sizing and cutting stock problems. We also refer to another review study recently published about operations scheduling for waste minimization in Hesran et al. [19].

2.2 Solution Methods for 1-Dimensional Cutting Stock Problems

There are a lot of solution methods to solve the BPP and the CSP in the literature. We refer to the review paper published by Delorme et al. [20]. It reviews successful methods in the literature in solving the one-dimensional CSP and BPP and classifies these methods into four main groups such as upper and lower bounding techniques, pseudo-polynomial formulations, enumeration algorithms, and branch & price (B&P) approaches. They also provided the computational performances of these methods. Note that, we are given m items each having an integer weight w_j (j = 1,2,...,m), and an unlimited number of identical bins of integer capacity c in the bin packing problem. The aim is to pack all items by minimizing the number of bins used. However, all items are demanded in one piece apart from the cutting stock problem.

We restrict our review of solution methods with prominent studies and those which are most relevant with and inspiring our solution methods. Remaining parts of this section are organized in three main classes, namely, formulations, exact methods, and heuristics & metaheuristics.

2.2.1 Formulations

Pseudo-polynomial formulations are widely used to solve the CSP. They are useful tools for solving large size problems. However, their strengths are highly dependent on the problem parameters. When the stock capacity increases, number of variables and constraints increase. We reviewed several pseudo-polynomial formulations for the CSP.

The one-cut formulation independently developed by Rao [21] and Dyckhoff [22] is an early pseudo-polynomial formulation for the CSP. The idea behind this formulation is dividing the stock length into two pieces. The left piece is used for an item and the right piece is residual that can be used either for producing other items or another item.

The DP-flow was proposed by Cambazard and O'Sullivan [23] for the BPP, but it can be extended to the CSP. It uses classical dynamic programming approach.

Carvalho [4] proposed an arc-flow formulation and implemented a branch & price method. He formulated the CSP as a minimum flow network problem. Items are represented on normal arcs and wastage is represented on loss arcs. He provided three efficient criteria to reduce network size. His formulation is very powerful to attack the 1-D CSP and the BPP if the stock capacity is not very large. Note that his work is mentioned in detail at Chapter 4. Brandão and Pedroso [24] proposed an alternative arc-flow formulation and an algorithm of graph compression for reducing network size. Delorme and Iori [5] suggested the reflect formulation which is an enhanced version of the arc-flow formulation. Since these formulations are pseudo-polynomial, they become weak when the capacity of stock length increases. It is because of that increase in stock capacity will lead to an increase in network size considerably. However, the reflect formulation uses half-length of the stock capacity so that the number of nodes and arcs reduce substantially. Hence, it partially handles the weakness of arc-flow formulation and becomes a powerful tool to attack the BPP and the CSP. The reflect formulation is mentioned in detail in Chapter 4.

2.2.2 Exact Methods

Branch & price method is widely used to obtain integer optimal solution for the CSP and the BPP. It is basically based on combining two well-known methods, column generation, and branch & bound. Although its implementation is difficult, it is a very successful tool for solving these kinds of problems.

Set covering formulations have been used to solve the cutting stock problems by enumerating all possible patterns because it gives a strong LP relaxation as a lower bound. On the other hand, generating all possible cutting patterns grows exponentially in number. Therefore, it is prohibitive to enumerate patterns even for moderate size instances. Column generation method, invented by Gilmore & Gomory [8] for the CSP, is based on the set covering formulation but it generates patterns iteratively and adds them to the model according to their contribution to the objective function. Although this method gives very strong lower bound, the resulting solution may not be an integer. Integer round-up property was conjectured for the CSP in the seventies. Optimal integer solution can be obtained by rounding up the LP relaxation of set covering formulation. However, it was shown by Marcotte [25] that there are some instances that violate integer round-up property. On the other hand, a modified integer round-up property (MIRUP) was conjectured by Scheithauer & Terno [26]. MIRUP claims that the rounded-up value of LP relaxation plus one is greater than or equal to the optimal integer solution. Until now, there are no instances shown yet that MIRUP property does not hold. Therefore, it is still an open conjecture for the CSP.

Branch & Price method has been used to obtain an optimal integer solution for the CSP and the BPP. It is based on column generation method, but if the obtained solution is fractional, branch & bound is implemented to escape fractional solution. Therefore, column generation is repeatedly implemented at the nodes of branch & bound tree until an optimal integer solution is found.

Vance [27] proposed a branch & price method for solving the binary cutting stock problem by using branching rules developed by Ryan & Foster [28]. Then, Scheithauer and Terno [29] developed a method by combining MIRUP conjecture and branch & price. Vance [30] compared the two branch & price methods: Danztig-Wolfe decomposition on the Kantorovich formulation and set covering formulation used by Gilmore & Gomory for column generation. It is shown that both methods provide the same LP relaxation value. On the other hand, both methods need different branch & bound approaches. She proposed branching decisions for both formulations and provided a comparison of experimental results. Valério de Carvalho [4] implemented column generation on the arc-flow formulation. Belov and Scheithauer [31] proposed a very powerful branch & price approach which is quite successful in solving well-known instances in the literature.

There are a few enumeration algorithms for solving the CSP and the BPP exactly such as branch-and-bound and constraint programming approaches. An early branch-and-bound algorithm for the BPP was conducted by Eilon and Christofides [32]. It is based on combining the best-fit decreasing algorithm and standard LP relaxation, but it cannot solve large instances. Martello and Toth [33] proposed a powerful branch-and-bound algorithm, called MTP, by using better heuristics and improved lower bounds.

Another successful branch-and-bound algorithm for the BPP, known as BISON, was developed by Scholl et al. [34]. They combined tabu search and some tools from MTP to compose their algorithm. Constraint programming approaches were also used to attack the BPP and the CSP. Shaw [35] proposed a new dedicated constraint based on a set of pruning and propagation rules. Cambazard and O'Sullivan [23] used pseudo-polynomial formulations in constraint programming approach. Schaus et al. [36] introduced a filtering rule based on cardinality considerations.

2.2.3 Heuristic and Metaheuristic Approaches

There are several heuristics presented in the literature to solve the BBP and the CSP. However, there is no exact polynomial time algorithm that can solve these problems. There are basically three types of heuristics discussed in the literature: approximation algorithms, lower bounds, and metaheuristics.

Best-fit, first-fit, and next-fit algorithms are basically used as a simple approximation algorithm in the literature. It is shown that in the study of Semi-Levi [37], next-fit algorithm's worst-case performance ratio is 2 and it is worse than others. On the other hand, the first-fit and best-fit algorithms have the same worst-case performance ratio (WCPR), which is 17/10, but it can be improved by sorting items according to their lengths in descending order. The new algorithms are called best-fit decreasing and first-fit decreasing, and their WCPR is 3/2 for the BPP. Even though their WCPR is equal, the first-fit algorithms obtain solutions more quickly.

There are some heuristics to calculate lower bound on the BPP and the CSP. LP relaxation of the Kantorovich formulation can be used as a lower bound, although its WCPR is poor. Martello and Toth [38] provide a better lower bound and its WCPR is 2/3. There are also metaheuristics developed for the BPP and the CSP. The large majority of metaheuristics have been modified for the CSP such as the tabu search, genetic, simulated annealing, etc. As a prominent work, we refer to Quiroz-Castellanos [39] who developed a very successful genetic algorithm for the CSP.

The main contribution of this thesis is to integrate Operations Research tools into the cutting stock problem for the construction industry. The 1.5-D CSP in the construction industry is a novel application that considers diameter conversion and cutting patterns simultaneously. To the best of our knowledge, convertibility of diameter into different sizes makes the problem unique and the problem based on a real case in the construction industry is firstly examined. Moreover, we considered the multi-period case of the problem where decision maker must deal with the trade-off between trim loss and inventory holding cost by having an option of cutting the demands of the latter in the previous periods. To the best of our knowledge, this is the first study that takes both multi-period and the 1.5-D CSP structure into account together.

CHAPTER 3

PROBLEM DEFINITION

In this chapter, we define the 1.5-dimensional cutting stock problem with diameter conversion in the construction industry. Background information, problem properties, and mathematical formulations are provided separately for the single-period and multi-period problems in Sections 3.1 and 3.2, respectively. The single-period cutting stock problem does not consider the time dimension. On the other hand, the multi-period cutting stock problem considers the case where orders appear in different time-periods and decision maker should compromise the trade-off between trim loss and inventory holding cost.

3.1 Single-Period Cutting Stock Problem with Diameter Conversion in the Construction Industry

In this section, we will define the single-period cutting stock problem with diameter conversion in the construction industry. In this problem, the decision maker considers the selection of diameter sizes and cutting patterns for one period. We consider Kantorovich, arc-flow, and reflect as mathematical formulations of the problem and modify them for this problem.

3.1.1 Problem Formulation

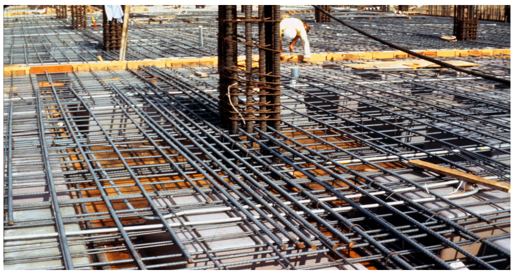
Reinforcement steel bar is widely used in the construction industry. It provides necessary tensile strength to the structure when it is placed in concrete with a sufficient amount. It is important to manage the usage of rebar for construction companies because it constitutes a significant cost of construction. To illustrate, rebars are used in all concretes for different purposes such as stirrup, shear wall, slab, column, and beam. Rebars are generally used in several diameter sizes in the industry: 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, and 32 in millimeters. However, some special structures may need 36 and 40 in millimeters. Rebars are usually 12 meters because it is the maximum amount that trucks can carry. Figure 3.1 shows reinforcement steel bars used in the construction industry.

Each structural element requires different sizes of rebar in length according to its geometric shape and place. Order of rebars is met by cutting 12m length stocks into the desired length. Cutting processes are conducted by workers on a workbench in the construction zone. The 1-D CSP occurs during the cutting processes. Therefore, cutting patterns should be generated according to the cutting plan to minimize the usage of materials. On the other hand, the number and diameter sizes of rebars are determined by the structural designer to provide the necessary tensile strength. After a load of concrete is calculated, necessary cross-section area between rebar and concrete is determined for each structural element. In addition, there are some rules that should be followed for locating rebars into concrete such as spacing where the distance between rebars in the concrete should be calculated. Therefore, a designer should determine the diameter size and the number of rebars in concrete by considering these rules. In general, a few diameter sizes are appropriate to be chosen to satisfy these rules at the same time. Required number of rebars is calculated to satisfy the necessary cross-section area using the formula:

$$\text{Required number of rebars} = \frac{\text{Necessary cross section area}}{\pi(\text{radius})^2}$$

Only one diameter size can be chosen for each structural element due to structural rules and operational simplicity. Therefore, the necessary cross-section area cannot be satisfied by using more than one size of diameter.

There are some flexibilities in determining diameter size and the required number of rebars. These sizes and numbers can be converted to different sizes if some conversion rules are followed. The primary condition that needs to be satisfied is that the necessary ratio of rebar cross-section area to the concrete area should stay con-


(a)

(b)

Figure 3.1: Rebars used in construction industry

stant. It means that the sum of the areas of the rebars in concrete should be constant. Therefore, a decision maker can change the required number of rebar by varying the diameter size. The conversion formula is given below:

$$(\text{chosen diameter})^2 * \text{required \# of rebar} = (\text{new diameter})^2 * \text{ new required \# of rebar}$$

The number of required rebar can be decreased by increasing its diameter size and vice versa if equality above holds true. However, diameter size cannot be converted more than two step sizes because of construction standards. To illustrate, if diameter size is initially chosen as Ø14, it can be converted to diameter sizes as Ø10, Ø12, Ø16, and Ø18.

The following example shows that instead of using 100 rebars of Ø16, the decision maker can select to use 64 rebars of Ø20 as given below.

$$(16)^2 * 100 = (20)^2 * 64$$

Note that, it may be beneficial to consider the conversion when generating cutting patterns because better cutting patterns can be generated using conversion. Cutting pattern denotes how items are cut from stock by combining number of items in different length in the stock capacity. Suppose that orders are assumed to be as shown in Table 3.1.

Note that, if the conversion is not considered during the pattern generation process, best patterns should be as follows: six times 3-7 and a 2-2-2. However, the last order can be converted to the diameter 12 by using conversion formula 3x14x14=Rx12x12.

Table 3.1: Example orders

Diameter (mm)	**Length (m)**	**Required #**
12	3	6
12	7	6
14	2	3

R is found as 4.08 but rounded to 5 in order to stay on the safe side. In this case, cutting patterns can be generated as five times 3-7-2 and a 3-7 with 2 m loss. As a result, usage of the material is decreased by using leftover parts of rebars from diameter 12.

Classical problems (1-D CSP) and (2-D CSP) have been widely studied in the literature. The 1-D CSP deals with the determination of the cutting patterns in a 1-dimensional space. The decision maker of the problem tries to find the best cutting patterns according to demand quantities for different sizes. The 1-D CSP has been widely used for determining cutting patterns of reinforcement steel bar in the construction industry. In the problem, the length of the rebar is fixed, and the pieces with different sizes are cut from the large stock to meet the demand of different sizes. The problem aims to minimize the usage of raw material by generating the best cutting patterns. Furthermore, the 2-D CSP considers both width and length requirements of the demands. The objective of this problem is the same as the 1-D CSP.

In the 1.5-dimensional cutting stock problem, sizes of one dimension are not fixed and can be changed by the decision maker by altering the required order of items. It means that the number of pieces being cut is not known until the sizes of the variable dimensions are fixed. The aim of the problem is to minimize the usage of materials by determining cutting patterns and choosing the proper sizes of variable dimension. The early definition of the problem is made by Haessler [11] at 1978 on coil slitting problem in the industry.

In our problem, diameter sizes are convertible to each other by applying conversion rules and the number of pieces that should be cut is not known until diameter size is chosen for each structural element. The decision maker should determine which part of demands are met from which diameter of rebar and cutting patterns to minimize the usage of materials. Since choosing diameter sizes and generating cutting patterns affect each other, the decision should be made simultaneously. Table 3.2 shows the requirements of items in their original diameter and possible conversion in different diameters. For example, 48 pieces in Ø12 are demanded for 4th item and its demand can be converted to 108, 70, 36 and 27 in quantity if diameter size is chosen as Ø8, Ø10, Ø14, and Ø16 respectively. Hence, there are flexibilities in diameter sizes and their corresponding quantities. Note that, each item has a range of convertibility specified by the designer. Mostly two steps of sizes to up or down are allowable but sometimes it may be tighter or even not convertible. The decision maker can convert these requirements to generate better cutting patterns so that better utilization level of materials is achieved.

Table 3.2: Conversion table

Items	**Dia.**	**Length**	**Quant.**	**Ø8**	**Ø10**	**Ø12**	**Ø14**	**Ø16**	**Ø18**	**Ø20**
1	8	8.50	808	**808**	518	360	-	-	-	-
2	8	3.45	202	**202**	130	90	-	-	-	-
3	10	6.65	265	415	**265**	185	136	-	-	-
4	12	7.00	48	108	70	**48**	36	27	-	-
5	12	9.15	105	237	152	**105**	78	60	-	-
6	14	8.05	203	-	398	277	**203**	156	123	-
7	14	3.15	1685	-	3303	2294	**1685**	1291	1020	-
8	16	1.50	24	-	-	43	32	**24**	19	16

* Dia: Diameter (mm), Length: Stock length (m), Quant: Quantity (units)

The problem that we consider differs from the classical 1-D CSP because there is an additional diameter dimension. On the other hand, it also differs from the classical 2-D CSP as sizes of the diameters are not fixed and can be changed by decision maker. The cutting stock problem with diameter conversion in the construction industry can be defined as the 1.5-D CSP since order sizes of diameter are not fixed and can be converted by the decision maker by changing the required number of rebars.

3.1.2 Mathematical Formulations

Different versions of mathematical formulations for the single period cutting stock problem with diameter conversion in the construction industry will be presented in this subsection.

3.1.2.1 Kantorovich Formulation

Kantorovich [7] formulated the cutting stock problem in 1939. Although the formulation is weak and cannot solve even small size instances, it is helpful to understand the problem. We modified his formulation according to the structure of the 1.5-D CSP. Since it is known that LP relaxation of this formulation is very bad, we provide this formulation to a solver that applies a branch & price algorithm by using Dantzig-Wolfe decomposition to be mentioned in Chapter 4.

Table 3.3: Notation of the Kantorovich formulation for 1.5-D CSP

	Sets:
I	Set of items
J	Set of diameters
U_i	Set of eligible diameters for converting item i
K	Set of patterns
	Parameters:
c_j	cost of unit stock with diameter j
b_{ij}	demand of item (in units) i if it is cut from stock with diameter j
w_i	length of item i
W	stock length capacity
	Decision Variables:
y_{ijk}	number of times item i is cut in pattern k with diameter j
x_{jk}	$= \begin{cases} 1, & \text{if pattern } k \text{ with diameter } j \text{ is used} \\ 0, & \text{otherwise} \end{cases}$
m_{ij}	$= \begin{cases} 1, & \text{if diameter } j \text{ is chosen for item } i \\ 0, & \text{otherwise} \end{cases}$

$$Min \quad \sum_{j\in J}\sum_{k\in K} c_j x_{jk} \tag{3.1}$$

$s.t.$

$$\sum_{k} y_{ijk} \geq b_{ij} m_{ij} \qquad \forall i \in I, \forall j \in J \tag{3.2}$$

$$\sum_{i} w_i y_{ijk} \leq W x_{jk} \qquad \forall j \in J, \forall k \in K \tag{3.3}$$

$$\sum_{j\in U_i} m_{ij} = 1 \qquad \forall i \in I \tag{3.4}$$

$$y_{ijk} : integer \qquad \forall i \in I, \forall j \in J, \forall k \in K \tag{3.5}$$

$$x_{jk} \in 0-1 \qquad \forall j \in J, \forall k \in K \tag{3.6}$$

$$m_{ij} \in 0-1 \qquad \forall i \in I, \forall j \in J \tag{3.7}$$

There is an additional dimension, diameter, apart from the original Kantorovich formulation for the 1-D CSP. Objective function (3.1) minimizes the total cost of rebars used. Constraints (3.2) assure to satisfy the required number of rebars according to chosen diameter sizes. Constraints (3.3) create feasible patterns by combining items, whose chosen diameter sizes are same, in stock length capacity W. They also ensure that if item i is cut in pattern k with diameter j, corresponding variable x_{jk} should get value 1. Constraints (3.4) guarantee that each item's demand should be satisfied from only one diameter size. Constraints (3.5)-(3.7) are the binary and integrality restrictions.

The Kantorovich formulation has a kind of pseudo-polynomial structure because the number of patterns k increases pseudo-polynomially if the amount of order is increased. Therefore, instances with high demand cause the model to have more decision variables. Although the formulation can be solved by branch & price method, its pseudo-polynomial and symmetrical structure make difficult to obtain a solution in a reasonable time.

3.1.2.2 Arc-Flow Formulation

Carvalho [4] proposed the arc-flow formulation which is quite successful in solving the cutting stock problems. Given stock length capacity of W and item sizes $w_1, w_2, \ldots, w_m$. He formulated the problem as the minimum flow network problem on graph $G = (V, A)$ with $V = 0, 1, 2\ldots, W, A = (i, j) : 0 \leq i < j \leq W$ and $j–i = w_d$ for every $d \leq m$. There are item arcs and loss arcs used in the graph. Item arcs represent the items while loss arcs represent the trim loss. There should be an item arc between two vertices if there is an item size equal to arc size. In addition, loss arcs such that $(k, k+1), k = 0, 1\ldots, W–1$ are used to denote the unused part of stock length. The network consists of nodes between 0 and the possible combination of the total length of items in the pattern till stock length. There is a pattern generated if and only if there is a path between 0 and W. Figure 3.2 shows a generic graph structure of the arc-flow formulation. In the figure, arcs (0,3), (1,4), and (2,5) represents the item-1, while arcs (0,2), (1,3), (2,4), (3,5) represent the item-2 according to their length. Loss arcs such as (0,1), (1,2), (2,3) represent the unused part of the pattern. Any path from 0 to 5 denotes the possible patterns. For example, the path, (0-2-4-5), indicates the pattern that includes 2 copies of item-2 and one unit of trim loss. Table 3.4 shows the notation of the arc-flow formulation for 1.5-D CSP and 1.5-D CSP formulation is given as:

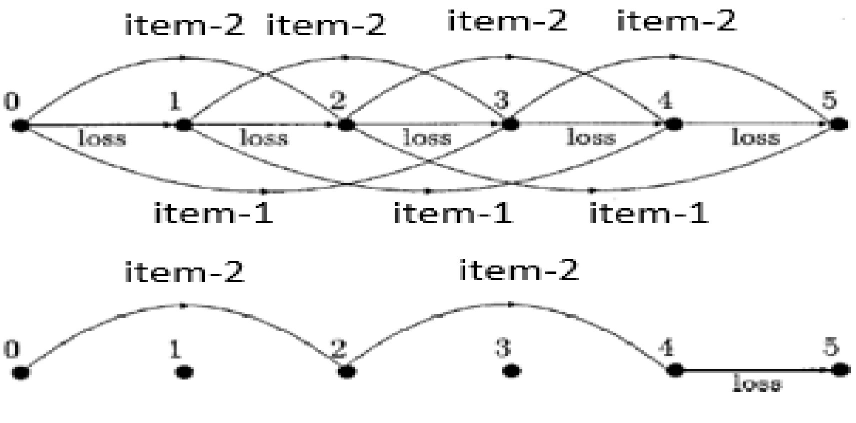


Figure 3.2: Example for network representation of the arc-flow formulation [4]

Table 3.4: Notation of the arc-flow formulation for 1.5-D CSP

	Sets:
I	Items
J	Diameters
d, e, f	Nodes
n	Last node
	Parameters:
w_i:	lenght of item i
b_{ij}:	demand of item i in diameter j
c_j:	unit cost of stock with diameter j
	Decision Variables:
x_{dej}:	flow between node d and e with diameter j
z_j:	total flows emanating from vertex 0 in diameter j
m_{ij}	$= \begin{cases} 1, & \text{if diameter } j \text{ is chosen for item } i \\ 0, & \text{otherwise} \end{cases}$

$$Min \quad \sum_j c_j z_j \tag{3.8}$$

$s.t.$

$$\sum_e x_{0ej} = z_j \qquad \forall j \in J \tag{3.9}$$

$$\sum_d x_{dej} - \sum_f x_{efj} = 0 \qquad \forall e, \forall j \in J \tag{3.10}$$

$$\sum_d x_{dnj} = z_j \qquad \forall j \in J \tag{3.11}$$

$$\sum_{d,d+w_i} x_{d,d+w_i,j} \geq b_{ij} m_{ij} \qquad \forall i \in I, \forall j \in J \tag{3.12}$$

$$\sum_j m_{ij} = 1 \qquad \forall i \in I \tag{3.13}$$

$$m_{ij} \in 0-1 \qquad \forall i \in I, \forall j \in J \tag{3.14}$$

$$z_j : integer \qquad \forall j \in J \tag{3.15}$$

$$x_{dej} : integer \qquad \forall d, \forall e, \forall j \in J \tag{3.16}$$

Objective function (3.8) minimizes the total flows z_k, emanating from node 0, which represents the total bars used associated with the unit cost of stock c_k according to diameter k. Constraints (3.9) - (3.11) are flow balance equations. Constraints (3.12) ensure that the number of arcs that represents item d should be greater than or equal to demand the amount of item d if diameter size k is selected for item d. Constraints (3.13) provide that every item should be cut from only one diameter size of stocks. Constraints (3.14)-(3.16) are binary and integrality restrictions.

3.1.2.3 Reflect Formulation

In order to formulate the problem, we use the knowledge accumulated in the literature. Carvalho's [4] arc-flow formulation is a pseudo-polynomial formulation because it has $O(mc)$ variables and $O(m + c)$ constraints. Thus, it becomes weak when the stock capacity increases. To handle this weakness, Delorme and Iori [5] developed the reflect formulation. It uses half of the stock capacity so that the number of arcs and nodes are reduced substantially. Therefore, this paves the way for the model to possess a few number of constraints and variables. It is a more powerful formulation for solving the cutting stock problems. The computation times are also improved by Delorme's [5] reflect formulation.

The reflect formulation has properties as listed below:

1. It uses vertices as in the normal patterns but from 0 to $\dfrac{W}{2}$ and extra vertex, called R, whose corresponding size is $\dfrac{W}{2}$.
2. The formulation converts each item arc (d, e) in the arc-flow formulation whose $d < \dfrac{W}{2}$ and $e > \dfrac{W}{2}$ into arc $(d, W\text{-}e)$.
3. It eliminates all items and loss arcs (d, e) whose $d > \dfrac{W}{2}$.
4. It adds a last loss arc between right most vertex before R with R.

A pattern can be generated as a pair of two colliding paths. Both start from 0 to the same vertex, but only one of them can pass through the R. In other words, only one of them can include reflected arcs. Figure 3.3 depicts the reflection of paths.

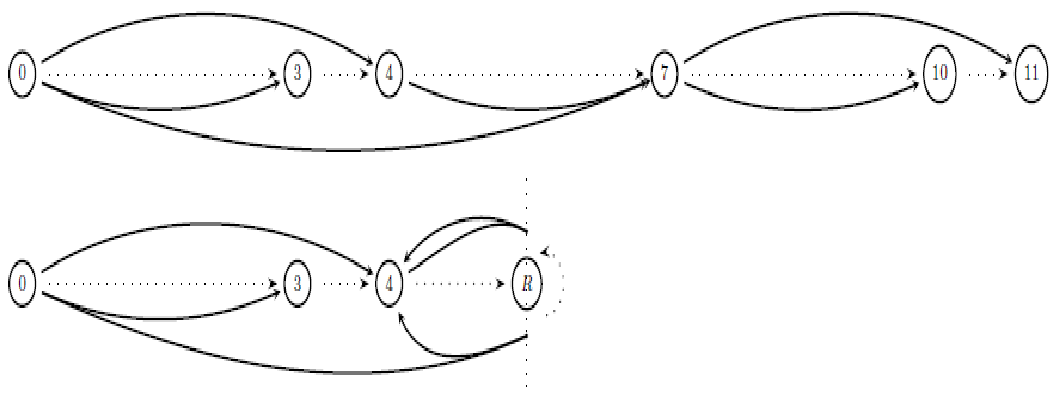


Figure 3.3: Example for network representation of the reflect formulation [5]

Multi Graph $G = (V, A)$ used in the reflect formulation consists of vertices such that $V = 0 \cup e \in N, 0 < e < \frac{c}{2} \cup \frac{c}{2}$. The set of arcs A includes two different arc types which are standard arcs, A_s, and reflected arcs, A_r. A_r includes reflected arcs such that they are converted from arc (d, e) such that $d < \frac{W}{2}$ and $e > \frac{W}{2}$ to $(d, W\text{-}e)$. As includes all other arcs in the arc-flow formulation such as normal item arcs and loss arcs whose e is less than $\frac{W}{2}$. Notations (d, e, r) and (d, e, s) represent arcs from d to e reflected and standard respectively, whereas (d, e, k) used for generic arcs from either A_s or A_r.

We modified this formulation by adding diameter dimension according to the structure of the 1.5-dimensional cutting stock problem based on sets, parameters and decision variables summarized in Table 3.5. The reflect formulation for our problem is provided.

Table 3.5: Notation of the reflect formulation for 1.5-D CSP

	Sets:
I	Items
J	Diameters
d, e, f	Nodes
A_r:	Set of reflected arcs
A_s:	Set of standard arcs
A_i:	Set of arcs, include both reflected and standard arcs, whose sizes correspond to the length of item i
r, s, k:	Arc type
$\delta_s^-(e)$:	denotes the set of standard arcs entering e
$\delta_r^-(e)$:	denotes the set of reflected arcs entering e
$\delta^+(e)$:	denotes the set of arcs emanating from e
	Parameters:
b_{ij}:	demand of item i in diameter j
c_j:	unit cost of stock with diameter j
	Decision Variables:
ξ_{dekj}:	arc between d and e with arc type k for diameter j
m_{ij}	$= \begin{cases} 1, & \text{if diameter } j \text{ is chosen for item } i \\ 0, & \text{otherwise} \end{cases}$

$$Min \quad \sum_{j} \sum_{(d,e,r)\in A_r} c_j \, \xi_{derj} \tag{3.17}$$

$s.t.$

$$\sum_{(d,e,s)\in\delta_s^-(e)} \xi_{desj} = \sum_{(d,e,r)\in\delta_r^-(e)} \xi_{derj} + \sum_{(e,f,k)\in\delta^+(e)} \xi_{efkj} \qquad e \in V - \{0\}, \forall j \in J \tag{3.18}$$

$$\sum_{(0,e,k)\in\delta^+(0)} \xi_{0ekj} = 2 \sum_{(d,e,r)\in A_r} \xi_{derj} \qquad \forall j \in J \tag{3.19}$$

$$\sum_{(d,e,k)\in A_i} \xi_{dekj} \geq b_{ij} m_{ij} \qquad \forall i \in I, \forall j \in J \tag{3.20}$$

$$\sum_{j} m_{ij} = 1 \qquad \forall i \in I \tag{3.21}$$

$$\xi_{dekj} : integer \qquad \forall (d,e,k) \in A, \forall j \in J \tag{3.22}$$

$$m_{ij} \in 0-1 \qquad \forall i \in I, \forall j \in J \tag{3.23}$$

Objective function (3.17) minimizes reflected arcs associated with unit cost of stock c_j according to diameter j. Since patterns are composed of two colliding paths, where the first one only includes standard arcs and the second one includes reflected arc, the number of total bars used equals the number of reflected arcs. Hence, minimizing the number of reflected arcs corresponds to minimizing the total bars used. Constraints (3.18) are a kind of flow balance equation, and they guarantee that amount of flow on standard arcs entering node e equals the sum of flow on every arc emanating from node e, and flow of reflected arcs entering node e. Constraints (3.19) are also a kind of flow balance equation, and they ensure that the amount of flow emanating from node 0 should be equal to twice amount of reflected arcs. It provides that patterns are composed of two colliding paths. Constraints (3.20) ensure that demand for items should be satisfied according to the selection of diameter type j. Constraints (3.21) ensure that every item should be cut from only one diameter type. Constraints (3.22) and (3.23) are binary and integrality restrictions.

3.2 Multi-Period Cutting Stock Problem with Diameter Conversion in Construction Industry

In this section, we will define the multi-period cutting stock problem with diameter conversion in the construction industry. In this problem, a decision maker should decide the selection of diameter sizes, cutting patterns, and cutting times by compromising the trade-off between trim loss and inventory holding cost. Mathematical formulations of Kantorovich and reflect are modified for this problem. and the problem is 1.5-D MPCSP.

3.2.1 Problem Formulation

Some construction projects can be composed of multiple buildings such as multiple unit housing projects, holiday sites, and public building complexes, etc. During the construction of these projects, reinforcement steel bars are required at different times because buildings are not constructed at the same time. In addition, some processes are repetitively pursued at different times according to the business plan due to the inherent properties of the construction. One of the major processes conducted repetitively is pouring of concrete for beams and columns at different times. Especially in huge projects such as skyscraper, bridge or multiple building complex, there are multiple repetitive works during construction and time intervals associated with them. Reinforcement steel bar is used in this process and embedded into concrete to provide tensile strength for each concrete pouring process. Rebars should be cut into the desired length before the process begins. Therefore, the need for rebars can occur at different times. To explain further, the completion time of huge construction projects might be more than one year. For example, during the construction of multiple buildings complexes, buildings are constructed sequentially according to some business plan to allocate resources such as labor and equipment efficiently. These different times can be defined as periods and demand for rebars should be satisfied in different periods throughout the project. If these orders are considered independently during generation of cutting patterns, it may lead to missing advantages of obtaining better solutions in terms of trim losses. On the other hand, aggregation of orders which appear in different periods needs keeping materials in the stock. In other words, if a cutting pattern is composed of items whose demands are in different periods, the cutting process should be completed before the closest deadline of the item in the pattern. It causes some items to be kept in inventory to meet future periods of demand. Therefore, it requires paying inventory holding cost for these items. Note that there are multiple periods in the construction projects, and periods have the following properties:

- Each period has its own demand for reinforcement bars from different diameters and lengths.
- Reinforcement bars should be ready before each pouring process begins. It means that orders of small items should be cut from large stocks before the pouring process of each period.
- Pouring process takes time and there should be waiting time after it is completed. Therefore, there should be enough time between periods to construct durable concrete. Sometimes, periods may get longer according to the business

plan of the project.

Increasing number of items in the cutting stock problems provides better utilization of stock materials. Therefore, aggregating orders of different diameters and periods increases the chance to generate better cutting patterns in terms of trim losses. However, aggregating orders of periods needs to hold inventory. To illustrate, suppose that there are five periods and each has its own demands for different items. If items, that are demanded for period four, are cut in period two to generate better-utilized patterns with items, that is demanded for period two. Items for period four should be kept in inventory for two periods. As a result, inventory holding cost will be incurred for two periods.

Briefly, it can be said that there is a trade-off between generating good patterns and holding cost. It may be beneficial to accept holding cost to reduce the cost of trim loss. It means that some reinforcement bars can be cut and stored to meet future demands. Formally, the problem is to determine which item should be cut at which period; select diameter for each item and generate cutting patterns for orders of each period in order to minimize the usage of raw material and holding cost.

3.2.2 Mathematical Formulation

We consider Kantorovich and reflect versions of mathematical formulations for solving the multi-period cutting stock problem with diameter conversion in the construction industry.

3.2.2.1 Kantorovich Formulation for Multi-Period Case

The Kantorovich formulation is not capable of solving small size instances within a reasonable time even for the single-period problem. However, it is useful to share this formulation for the purpose of understanding problem structure easily.

Table 3.6: Notation of the Kantorovich formulation for the 1.5-D MPCSP

	Sets:
I	Set of Items
J	Set of Diameters
U_i	Set of eligible diameters for converting item i
K	Set of Patterns
T	Set of Periods
	Parameters:
c_j	cost of stock per unit with diameter j
b_{ijt}	demand of item i if it is cut from stock with diameter j at period t
w_i	length of item i
h	holding cost for unit length per period
W	stock length
	Decision Variables:
y_{ijkt}	number of times item i is in the pattern k with diameter j at period t
In_{ijt}	amount of inventory from item i for diameter j in period t
x_{jkt}	$= \begin{cases} 1, & \text{if pattern } k \text{ is cut in diameter } j \text{ at period } t \\ 0, & \text{otherwise} \end{cases}$
m_{ij}	$= \begin{cases} 1, & \text{if diameter } j \text{ is chosen for item } i \\ 0, & \text{otherwise} \end{cases}$

$$Min \quad \sum_j \sum_k \sum_t W c_j x_{jkt} + \sum_i \sum_j \sum_t hc_j w_i In_{ijt} \tag{3.24}$$

$$s.t.$$

$$\sum_k y_{ijkt} = b_{ijt} m_{ij} + In_{ijt} - In_{ijt-1} \qquad \forall i \in I, \forall j \in J, \forall t \in T \tag{3.25}$$

$$\sum_i w_i y_{ijkt} \leq W x_{jkt} \qquad \forall j \in J, \forall k \in K, \forall t \in T \tag{3.26}$$

$$\sum_{j \in U_i} m_{ij} = 1 \qquad \forall i \in I \tag{3.27}$$

$$y_{ijkt} : integer \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall t \in T \tag{3.28}$$

$$x_{jkt} \in 0-1 \qquad \forall j \in J, \forall k \in K, \forall t \in T \tag{3.29}$$

$$m_{ij} \in 0-1 \qquad \forall i \in I, \forall j \in J \tag{3.30}$$

$$In_{ijt} : integer \qquad \forall i \in I, \forall j \in J, \forall t \in T \tag{3.31}$$

The objective function (3.24) minimizes the total cost of rebars and the inventory holding cost. The second term of the objective function indicates holding cost that is derived by considering length, diameter size, the number of items in the inventory, and the holding cost rate. Both term includes length parameters as W and w_i to make them comparable. Constraints (3.25) are the inventory balance equations. Constraints (3.26) provide to generate feasible cutting patterns. They also ensure that if item i is cut in pattern k with diameter j at period t, corresponding variable x_{jkt} should get value 1. Constraints (3.27) guarantee that each item can be cut from only one diameter size within allowable set U_i. Constraints (3.28)-(3.31) are binary and integrality restrictions.

3.2.2.2 Reflect Formulation for Multi-Period Case

We exploited from the reflect formulation to solve the 1.5-D MPCSP and modified it to the multi-period structure since it is a powerful tool to attack CSP unless stock capacity is quite large. In addition to the previous version, the time dimension is added to the formulation. The reflect formulation of the multi-period extension of our problem is provided below.

Table 3.7: Notation of the reflect formulation for the 1.5-D MPCSP

	Sets:
I	Items
J	Diameters
T	Periods
d, e, f	Nodes
A_r:	Set of reflected arcs
A_s:	Set of standard arcs
A_i:	Set of arcs, include both reflected and standard arcs, whose sizes correspond to the length of item i
r, s, k:	Arc type
$\delta_s^-(e)$:	denotes the set of standard arcs entering e
$\delta_r^-(e)$:	denotes the set of reflected arcs entering e
$\delta^+(e)$:	denotes the set of arcs emanating from e
	Parameters:
h:	holding cost for unit length per period
w_i:	lenght of item i
W:	stock length
b_{ij}:	demand of item i in diameter j
c_j:	cost of stock with diameter j
	Decision Variables:
ξ_{dekjt}:	arc between d and e with type k for diameter j at period t
In_{ijt}:	amount of inventory from item i for diameter j in period t
m_{ij}	$= \begin{cases} 1, & \text{if diameter } j \text{ is chosen for item } i \\ 0, & \text{otherwise} \end{cases}$

$$Min \quad \sum_j \sum_t \sum_{(d,e,r)\in A_r} Wc_j\,\xi_{derjt} + \sum_i \sum_j \sum_t hw_i c_j In_{ijt} \tag{3.32}$$

$s.t.$

$$\sum_{(d,e,s)\in\delta_s^-(e)} \xi_{desjt} = \sum_{(d,e,r)\in\delta_r^-(e)} \xi_{derjt} + \sum_{(e,f,k)\in\delta^+(e)} \xi_{efkjt} \qquad e \in V-0\ , \forall j \in J, \forall t \in T \tag{3.33}$$

$$\sum_{(0,e,k)\in\delta^+(0)} \xi_{0ekjt} = 2 \sum_{(d,e,r)\in A_r} \xi_{derjt} \qquad \forall j \in J, \forall t \in T \tag{3.34}$$

$$\sum_{(d,e,k)\in A_i} \xi_{dekjt} = b_{ij}m_{ij} + In_{ijt} - In_{ijt-1} \qquad \forall i \in I, \forall j \in J, \forall t \in T \tag{3.35}$$

$$\sum_j m_{ij} = 1 \qquad \forall i \in I \tag{3.36}$$

$$\xi_{dekjt} : integer \qquad \forall (d,e,k) \in A, \forall j \in J, \forall t \in T \tag{3.37}$$

$$m_{ij} \in 0-1 \qquad \forall i \in I, \forall j \in J \tag{3.38}$$

$$In_{ijt} : integer \qquad \forall i \in I, \forall j \in J, \forall t \in T \tag{3.39}$$

Objective function (3.32) minimizes the total cost of rebars and the total inventory holding cost. Constraints (3.33)-(3.34) are flow balance equations. Constraints (3.35) are inventory balance equations. They ensure that amount of items, which are cut in the current period and plus on-hand inventory from the previous period, should be equal to the demand of item for the current period plus the current inventory level for item i. Constraints (3.36) provide that every item should be cut from only one diameter. Constraints (3.37) - (3.39) are binary and integrality restrictions.

CHAPTER 4

SOLUTION APPROACHES

In this section, we will introduce our solution approaches to solve the 1.5-D CSP and the 1.5-D MPCSP. We implemented the Kantorovich formulation using SCIP-Generic Column Generator and CPLEX. On the other hand, we modified the reflect and arc-flow formulations for the 1.5-D CSP and the reflect formulation for the 1.5-D MPCSP. These formulations are implemented using only CPLEX. Furthermore, we proposed a genetic algorithm approach to solve large size instances of the 1.5-D MPCSP. The details of the solution approaches are discussed and organized as formulations in Section 4.1, and genetic algorithm approach in Section 4.2.

4.1 Solution of Mathematical Formulations

To solve our problem, we benefited from three different formulations namely, Kantorovich, arc-flow, and reflect. The implementation details of these formulations will be presented in this section.

4.1.1 Kantorovich Formulation

Branch & price (B&P) method is widely used in solving the cutting stock and bin packing problems to obtain an integer solution. B&P combines two well-known methods: Branch & Bound and Column Generation. Basically, the method applies column generation to the problem at the nodes of the branch & bound tree until an optimal integer solution is found. It is a powerful approach, but it requires significant efforts in the implementation. Therefore, we decided to use SCIP-Generic Column

Generator solver that uses branch & price algorithm logic.

The Kantorovich formulation is beneficial for understanding the problem structure easily, but it cannot be solved by using standard linear integer programming solvers for even small size instances. It is because of the weak LP relaxation of the formulation that does not give a good lower bound. To overcome this weakness, Gilmore and Gomory [8] invented a column generation method based on the set covering formulation, and its LP relaxation provides very good lower bound. Vance [30] showed that applying the Dantzig-Wolfe decomposition to the Kantorovich formulation gives the same lower bound with a set covering formulation. Since standard integer linear programming solvers cannot handle the Kantorovich formulation in a reasonable amount of time, we decided to use the SCIP-Generic Column Generator (GCG) solver due to two main reasons. First, the GCG automatically applies branch & price algorithm based on the Dantzig-Wolfe decomposition. Second, the implementation is quite easy compared to the manual implementation of the B&P methods, since the GCG handles many issues automatically such as column generation, tree management, branching decisions, etc. It is enough to provide information about how the problem should be divided into master and subproblems for the GCG.

The model should be divided into master and subproblems properly so that the GCG implements B&P method successfully. The Kantorovich formulation can be divided as constraints (3.2) and (3.4) in the master problem and constraints (3.3) are in the subproblems. We have J different diameter sizes and K different number of patterns provided initially. It can be observed that there would be J*K number of subproblems. However, the number of patterns, K, has no effects on the solution because all subproblems are same for each pattern k in K. Therefore, we conclude that there are J different types of subproblems and it is enough to solve these J subproblems at each iteration. GCG automatically detects these symmetries and aggregates subproblems into J different subproblems. Master and subproblems are divided as below:

Master Problem:

$$Min \quad \sum_{j \in J} \sum_{k \in K} c_j x_{jk} \tag{4.1}$$

$s.t.$

$$\sum_k y_{ijk} \geq b_{ij} m_{ij} \qquad \forall i \in I, \forall j \in J \tag{4.2}$$

$$\sum_{j \in U_i} m_{ij} = 1 \qquad \forall i \in I \tag{4.3}$$

$$y_{ijk} : integer \qquad \forall i \in I, \forall j \in J, \forall k \in K \tag{4.4}$$

$$x_{jk} \in 0 - 1 \qquad \forall j \in J, \forall k \in K \tag{4.5}$$

$$m_{ij} \in 0 - 1 \qquad \forall i \in I, \forall j \in J \tag{4.6}$$

Subproblem:

$$Max \quad ReducedCost \tag{4.7}$$

$s.t.$

$$\sum_i w_i y_{ijk} \leq W x_{jk} \qquad \forall j \in J, \forall k \in K \tag{4.8}$$

One can observe that the subproblems become knapsack problem in this partition. The GCG is a kind of black box solver, and it allows intervention in a limited manner. It uses the Dantzig-Wolfe decomposition method to implement branch & price. Not only it detects problem structure and suggests how the problem should be divided into the master and subproblems, but it also allows users to provide decomposition as well. We gave our decomposition structure to the solver and let it handle column generations, branching rules, tree management, etc. Nevertheless, the GCG could not handle moderate size instances. It is because of applying general frameworks to the problem, but it is needed to manage branch & price issues such as branching rules, symmetry breaking, etc. according to problem specific properties to solve large size instances successfully. Therefore, the general framework branch & price algorithm implemented by the GCG is not very successful for solving our problem. As a result, we decided to use network-based pseudo-polynomial formulations.

4.1.2 Arc-Flow Formulation

Pseudo-polynomial formulations have taken attention in solving the CSP and the BPP. There are several pseudo-polynomial formulations in the literature such as One-Cut, DP-Flow, and arc-flow. We focused on network-based formulations which are the arc-flow, introduced by Carvalho [4]. The cutting stock problems are handled as minimum flow network problem and items are represented by arcs in these formulations. However, the arc-flow formulation becomes weak when the capacity of stock length increases. Increasing stock length hardens problem because the network size grows enormously. Since stock capacity is fixed at 12 meters and precision used is 0.01 meters, the formulation is capable of solving instances in a reasonable time. To illustrate, item lengths increase in a way that 3.4 m, 3.41 m, 3.42 m, etc. However, it needs to generate network for implementing the formulation. As a result, it is crucial to generate a network efficiently to prevent symmetries in the solution space. Otherwise, both pre-processing time of generating network and solution time increase significantly.

Graph Generation:
The arc-flow formulation includes many symmetries in the solution space. Patterns that represent the same solution can be generated using different paths in the graph. Carvalho [4] proposes three criteria as mentioned below to decrease symmetries and network size.

Criterion 1:
An arc of size w_e, designated by $x(k, k + w_e)$, can only have its tail at a node k that is the head of another arc of size w_d, $x(k - w_d, k)$ for $w_d \geq w_e$, or, else, from node 0, i.e., the left border of the bin. It means items should be sorted in a non-increasing manner to avoid symmetry in the solution spaces. To exemplify, suppose that there are three items whose lengths are 5m, 4m, and 2m. To generate patterns with these items using 12m stock length, there are 24 (4!) ways of representing the solution in the graph such as 5-4-2-loss, 5-2-loss-4, loss-2-5-4, etc. Although item sequence in a pattern has no meaning to obtain low trim loss level, it appears in the graph as if they are different solutions. Carvalho's first criterion provides to avoid symmetry in the solution space by sorting them in non-increasing order. Therefore, only one solution can be valid which is 5-4-2-loss in the example.

Criterion 2:

All the loss arcs $x(k, k+1)$ can be set to zero for $k < w_m$. It means that loss arcs should begin from the node that the minimum size of item can fit. In other words, since patterns cannot start with loss arcs due to criterion 1, loss arcs from 0 to w_m, which is the smallest item length, should be set to zero initially. For example, if the minimum length is 4 in an instance, loss arcs (0,1), (1,2), (2,3), (3,4) should be set to zero.

Criterion 3:

Given any node k that is the head of another arc of size w_d $(w_d > w_e)$ or $k = 0$, the only valid arcs for size we are those that start at nodes $k + sw_e$, $s = 0, 1, 2, \ldots, b_e$–1 and $k + sw_e \leq W$, where b_e is the demand of items of size w_e. In other words, there should be no space between arcs.

We designed an algorithm to generate a directed acyclic graph for the arc-flow formulation by applying the criteria above. The pseudocode of this algorithm 1 is provided in Algorithm 1.

In the industry, the difference between two consecutive item's length is generally 0.05 or 0.01 meters, and stock length is 12 meters. To illustrate, if item's length increases 0.05 m spacing such as 3.4 m, 3.45 m, 3.5 m, there cannot be more than 240 types of item in terms of the item's length. Before generating a network for the arc-flow formulation, we expanded items' lengths into the range between 1 and 240. If item's length increases 0.01 m spacing such as 3.4 m, 3.41 m, 3.42 m, the maximum number of different item types in terms of length is 1200. Note that, precision used during design stage affects the network sizes significantly. If 0.01 m precision is chosen, it raises the number of nodes and arcs considerably. To generate an acyclic directed graph, the initial arcs set is introduced as 0-items' lengths. New arcs are added to obtain valid paths which represent possible patterns. To avoid adding redundant paths to the network, we used the information of last added items. The new arc is added to the tail point of a current incomplete path unless the length of the item is more than last added item of the current path and stock length capacity (C) is exceeded. Adding new items to the tail points of incomplete paths creates a number of new paths, which equals the number of added items, but the number of addable items decreases at every

Algorithm 1 Graph Generation Algorithm for the Arc-Flow

```
Step 1: Sort items in descending order
Step 2: Initialize paths as 0 - items' lengths set them as generation 1
    Keeps tail points of paths and last added items.
    Take generation 1 as current
    Do Step 3, Until no generation of path added
Step 3: Generate Graph
    For each tail point of current generation path
        For each item $i$
        If tail + length$_i \leq C$ and length$_i \leq$ last added item to tail point
            Generate arc as (tail point, tail point + item length)
            Add this arc to the network
            Keep item length as last added item for this path
            Keep new tail point of path of next generation
        Else
            Go to next tail point of current generation
            If there is no new tail point,
                Remove duplicated arcs in the network
                Take this new generation as current and return step 3
    If no generation added at last iteration, then stop.
Step 4: Create Loss Arcs
    Take all vertex in the network and sort them in ascending order
    Create loss arcs by connecting them each other consecutively.
Step 5: Expand graph into multi-graph by adding diameter dimension
    Take network created, set of arcs =(i,j)
    For each arc in the set convert it (i,j,k) for all k in K (set of diameter)
```

step. Adding processes continues until no addable items left. We keep arcs and add them into the network during generation processes. In order to prevent and avoid unnecessary iterations, duplications in the network are removed before starting to create a new generation.

To illustrate, Figure 4.1 shows a small example of creating networks using four items. The main idea of the algorithm is to generate paths whose arcs aligned in descending order. Note that vertex 50 can be reachable from by two paths, 0-50,0-30-50. It means the paths 0-30-50-90 becomes valid in the network despite nonincreasing order is desired. Hence, there are still some symmetries in the solution space.

We implemented the arc-flow formulation for our problem using CPLEX Concert Technology with C++. After the network is generated using Algorithm 1, it is provided to the model as problem parameters. Experimental results will be provided in Chapter 5.

4.1.3 Reflect Formulation

The reflect formulation, introduced by Delorme and Iori [5] as an enhanced version of the arc-flow, is used to solve our problems. Since an increase in stock length capacity causes that the arc-flow formulation becomes weak, Delorme and Iori [5] developed the reflect formulation which uses half of the length capacity in order to overcome this weakness. Therefore, it reduces the number of nodes and arcs in the network. It means that a new formulation is more powerful to solve large instances faster. Nevertheless, it is also a pseudo-polynomial formulation and it becomes weak if the capacity of stock length increases. On the other hand, the network structure includes many symmetries unless network used in the formulation is generated efficiently. In order to avoid these symmetries in the solution space and generate a network within a reasonable amount of time, we used a graph generation algorithm based on the ideas of Delorme and Iori [5].

Graph Generation:

The same logic with the arc-flow formulation can be used to avoid symmetry in the model. Mainly, sorting items according to their sizes in non-increasing order dur-

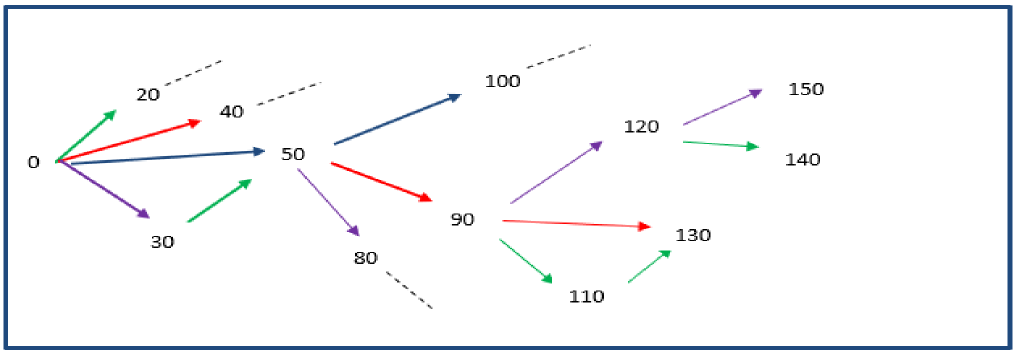


Figure 4.1: Graph generation example

ing generating graph eliminates redundant arcs and vertices, and it breaks possible symmetries.

Pseudocode of graph generation algorithm for the reflect formulation is provided in Algorithm 2. Note that similar to the arc-flow formulation, it is crucial to generate networks efficiently since obtaining solutions within a reasonable amount of time is significantly dependent on the network sizes. If these network-based formulations are implemented using an inefficient network, solution time increases considerably.

Apart from the arc-flow graph generation algorithm, arcs are reflected from the middle of stock length capacity in the reflect's algorithm. There are three types of arcs: standard, reflected, and loss arcs. The standard arcs can be generated like in the arc-flow's algorithm 1. They exist in the range between 0-capacity/2. If an arc has head in this range and its tail exceeds capacity/2, the arc should be reflected from the middle of the capacity. It should be $(i$, capacity-$j)$ rather than (i,j). If capacity is not an even integer, all items' length and capacity should be expanded by multiplying two.

We implemented the modified the reflect formulation for single and multi-period cases for our problem using CPLEX Concert Technology with C++. After the network is generated using Algorithm 2, it is provided to the model as problem parameters. Experimental results are discussed at Chapter 5.

Algorithm 2 Graph Generation Algorithm for Reflect

```
Step 1: Sort items in descending order
Step 2: Initialize paths
    Take items whose lengths $\geq$ C/2
    Generate reflected arcs as (0,C-length$_i$) for those items
    Remove these items from set of items
    Initialize paths as (0, length$_i$) set them generation 1
    Keep tail points of paths and last added items.
    Take generation 1 as current
    Do Step 3, Until no generation of path added
Step 3: Generate Graph
    For each tail point of current generation path
        For each item $i$
        If length$_i$ $\leq$ last added item to tail point
            If tail + length$_i$ $\leq C/2$
                Generate standard arc as (tail point, tail point + length$_i$, s)
                Add this arc to the network
                Keep item length as last added item for this path
                Keep new tail point of path of next generation
            Else
                Generate reflected arc as (tail point,$C$- tail point - length$_i$, r)
        Else
            Go to next tail point of current generation
            If there is no new tail point,
                Remove duplicated arcs in the network
                Take this new generation as current and return step 3
    If no generation added at last iteration, then stop.
Step 4: Create loss arcs
    Take all vertex in the network and sort them in ascending order
    Create loss arcs by connecting them each other consecutively.
Step 5: Expand graph into multi-graph by adding diameter dimension
    Take network created, set of arcs =(d,e,k)
    For each arc in the set convert it (d,e,k,j) for all j in J (set of diameter)
```

4.2 Genetic Algorithm Approach

In this section, we introduce a heuristic approach to solve the 1.5-D MPCSP. When problem size increases, solving the reflect formulation with CPLEX becomes ineffective. It has difficulty in solving problem instances more than 250 items. Therefore, we decided to design a genetic algorithm for our multi-period problem in order to reach a better solution quality within a reasonable amount of time. The parameters used in the algorithm are probabilities of crossover and mutation that are denoted by P_c and P_m, $Percent_{mutation}$ indicates the percentage of genes in a chromosome to be mutated, the $Percent_{best}$ that is the percentage to transfer the best parents to the mating pool directly, and $Percent_{elite}$ denotes the percentage of elite solutions at the initial population. While creating the initial population, we considered the feasible sets of diameter size and periods in our algorithm. Therefore, individuals in the population are always feasible. In the crossover operations, the offsprings are also always feasible. In the mutation operator, we also changed the genes according to their feasible sets. The details of the algorithm are provided in Sections 4.2.1-4.2.8. The pseudocode of the proposed algorithm is given in Section 4.2.9.

4.2.1 Representation Scheme

As a chromosome representation, we form a $2n$-dimensional array in order to represent diameter and period selection for an item, where n refers to the number of items in the problem. In the first n cells of the chromosome, chosen diameter sizes are represented at each cell. For this purpose, we use a discrete representation that a cell can only take the values of the diameter sizes which are 8, 10, 12...32. In the example provided at Figure 4.2, the first column of the chromosome has a value of 12, it means that the item-1 is assigned to be cut from the stock whose diameter size is Ø12. The second n dimension array is used for assignments of the items to the periods. The assigned period numbers are written to the chromosome to the related column. For example, the second n-dimension array includes '1' as a first entry. It means that the first item is assigned to period 1, and its demand should be cut at period 1. When we have n items, the chromosome representation is a $2n$-dimension array. Note that, periods are assigned for whole demand of each item. Cutting demand of items partially in different periods is not possible in this representation. We accept this weakness in advance because considering each unit of order partially lead to enormous computational effort.

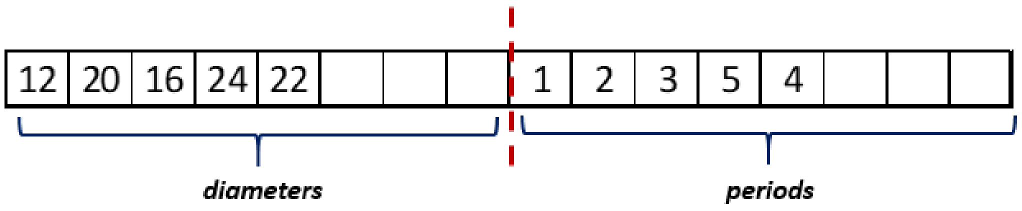


Figure 4.2: Chromosome representation

This representation is quite easy to decode. By looking at the value written in the chromosome, we can understand the assignments of the diameter sizes and periods to the items. In terms of memory, its requirement is small enough ($2n$-dimension array, n is the number of items).

4.2.2 Initial Population Generation

In the problem description, the designer initially assigns diameter size to the items. These are the desired diameter sizes for the items. On the other hand, the assigned diameter size can be changed as stated in Chapter 3, but it is a limited set. The assigned diameter size of the stock for the associated item should be within a range of four. To illustrate that, the assigned diameter size for an item is 16. Then, the diameter size of the associated item can be Ø12, Ø14, Ø16, Ø18, and Ø20. For each diameter size, we have a set of possible conversions.

While generating the initial population, we are using the information related to the possible diameter conversions. From the feasible set of conversions, we are randomly selecting one of the possible diameter sizes with equal probability.

For the period selection in the initial chromosomes, we are following similar steps. Deadlines of the cutting of items are previously determined according to the business plan. Therefore, the feasible set for the period assignment for the related item should be no later than the deadline. To illustrate that, the item i is required at period 3.

The feasible set for the cutting of item i can be performed at periods 1, 2, and 3. In the chromosome representation, we are selecting from the feasible values of the periods with equal probability. After determining the initial population, the values of the function can be calculated for each generated chromosome.

Instead of generating an initial population with entirely random chromosomes, we try to insert elite solutions to the population with small proportion so that the algorithm converges to better solution value more rapidly. Elite solutions can be achieved by implementing base heuristic which is derived using problem-specific information. The proportion of elite solution is restricted with a small rate to avoid premature convergence and let the algorithm explore better solutions. It can be said that the necessary diversity to explore better solutions is preserved since the majority of the population begins with random solutions. According to the preliminary results, using elite solutions in the initial population provides the algorithm to achieve better solution quality compared to the case where elite solutions are not included in the initial population. However, including elite solutions in the initial population may cause the algorithm to converge early if the percentage of elite solutions is increased overmuch. Therefore, we include the parameter levels of $Percent_{elite}$ in the experiments where parameters are fine-tuned at Chapter 5. The main idea behind base heuristic is aggregating diameter sizes of orders in order to generate better cutting patterns in terms of trim loss. Therefore, items are grouped by converting their original diameter sizes into some pre-determined diameter sizes. The main objective of the grouping step is decreasing the number of groups as much as possible so that each group has enough diversity to produce better cutting patterns. Since diameter conversion is restricted with a range of two steps up or down and there are 13 different diameter sizes in the industry, the number of grouping options is very limited. Orders can be aggregated within three groups as shown in Figure 4.3.

In this demonstration, orders are grouped in a way that 5-5-3, but there are also different versions of this partition such as 5-3-5 and 3-5-5. Arrows represent the conversion of diameter sizes. For instance, in the first group, orders whose original diameter sizes are 8-10-14, and 16 are converted to diameter 12. Besides, the group which consists of three diameter sizes can have different variants such as orders can be aggregated at 28 or 32. On the other hand, the groups which include five diameter sizes cannot have

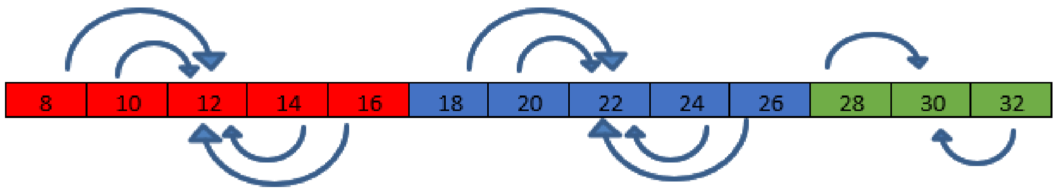


Figure 4.3: Example conversions of base heuristic

any variants because conversion is restricted with two steps. As a result, there are nine different versions of diameter aggregation if groups consist of numbers which are 5,5 and 3.

There is one more partition type to generate three groups as shown in Figure 4.4. In this partition, groups are made up of diameter sizes in number such as 4-4-5. There are also other versions of this partition such as 5-4-4 and 4-5-4. In addition, the groups of four diameter sizes have two different variants because each diameter sizes are convertible to each other. To illustrate, in Figure 4.4, diameter sizes 10, 12 and 18, 20 are the possible aggregation sizes since conversion is limited to two steps up or down. Hence, there can be 12 different types of diameter aggregation if groups consist of numbers such as 5,4 and 4. Ultimately, there are 21 different options to aggregate orders in three groups. In addition, choosing original diameter sizes for all item is the 22nd option as an elite solution. After assigning diameter sizes to the items, original deadlines are assigned as period information for all elite solution.

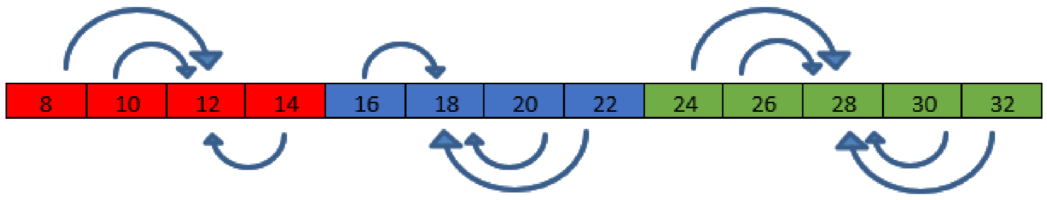


Figure 4.4: Example conversions of base heuristic

These 22 options derived from aggregation are evaluated, by calling CPLEX to solve

1-D CSPs, according to their objective function values, and the best solutions are transferred to the initial population as elite solutions with the proportion of $Percent_{elite}$. As a result, the algorithm can exploit these elite solutions so that it converges to better objective function value.

4.2.3 Fitness Function

When the chromosome representation is available, we have separate cutting stock problems for each diameter size and period, which gives D x P different 1-D cutting stock problems (assuming there are D different diameter sizes and P different periods). We can solve D x P different cutting stock problems for each chromosome and use their objective function value at the fitness function. The objective function values of the CSPs only give the material cost, but we also need to calculate the inventory holding cost. In the fitness function, we sum the objective function values of the CSPs and inventory holding costs. By this way, we can evaluate the quality of each chromosome.

To solve different cutting stock problems, we used two different techniques. The first one is the solving problems using original reflect formulation and calling CPLEX for each subproblem. However, we realized that it is time-consuming to call CPLEX D x P times for each chromosome. Although individual solutions can be obtained rapidly using the reflect formulation, calling it for each D x P subproblems leads to huge computation time at each iteration. Since problems for each chromosome are not dependent on each other, we applied parallel computing to solve the CSPs simultaneously by assigning threads to each problem. The implementation of parallel computing increased the speed of the algorithm as approximately two times faster compared to the serial implementation. However, the time consumed for solving each chromosome is still enormous to obtain solutions in a reasonable amount of time. Therefore, to get the solutions faster, we implemented a heuristic approach which is called first-fit decreasing algorithm with pre-allocated items to solve the CSPs separately, and evaluated the fitness function. Our preliminary results showed that solving CSPs approximately is very helpful to reduce computation times without sacrificing much from the solution quality. As a result, we decided to continue with this approach in order to evaluate the fitness function.

First-fit, best-fit, and next-fit algorithms are widely used to obtain approximate solutions rapidly in CSP and BPP. It is shown that worst-case performance ratio (WCPR) of the first-fit and the best-fit algorithm is 17/10 while next-fit's WCPR is 2 in terms of solution quality. Moreover, WCPR of first-fit and best-fit algorithm can be improved to 3/2 by sorting items in descending order as shown in the Semi-Levi [37]. Although their WCPRs are equal, first-fit can solve instances more rapidly. As a result, we decided to make use of first-fit decreasing algorithm so that approximate solutions are obtained rapidly without sacrificing solution quality significantly.

The algorithm begins to sort items according to their lengths in descending order. Then, locate items whose sizes are longer than half of the capacity into the stocks to eliminate redundant search since items whose sizes are larger than half of the capacity cannot be in the same pattern. The number of stocks opened is equal to the number of orders for each large item. Remaining items are tried to locate open stocks in order. If no open stocks are available to locate the next item, then the new stock will be opened. Two possible situations can be encountered during this assignment process if the length of the item is suitable to assign to the existing stocks. First, the order amount of the item on hand is more than the number of suitable existing stocks. In this case, the algorithm assigns orders of that item as much as possible to the existing stocks, and then it opens new stocks in number which equals to unassigned orders. Second, the order amount of the item is less than or equal to the number of suitable existing stocks. In the second case, orders are assigned to the existing stocks with an exact amount of order. Since the orders of items are assigned as bulk, not one by one, there can be inefficient assignments. Additional improvement algorithm is implemented to deal with inefficiency. It checks whether there is an obvious inefficient assignment or not in the solution. To illustrate, suppose that eight stocks are separately cut into only one size of length, 3 meters. However, these items can be cut together in two stocks by aggregating their demands as 3-3-3-3 and 3-3-3-3. Pseudocode of first-fit decreasing algorithm and example of improvement algorithm are mentioned in Appendix A.

4.2.4 Parent Selection

The parents are selected after calculating the fitness values of the chromosomes. Since our aim is to minimize costs in the problem, lower fitness function values are better. The first $Percent_{best}$ of the chromosomes that have minimum fitness values are directly transferred to the mating pool. According to the preliminary results, transferring a small proportion of best solution directly to the mating pool provides algorithm converges better solutions compared to not implementing. The level of $Percent_{best}$ is fine-tuned at Chapter 5. We applied a parent selection procedure. For each chromosome in the population, we evaluated the following function.

In equation 4.9, $select_i$ value will be used to determine the probability of selection the chromosome i, where f_{max} and f_{min} value refer to the maximum and minimum fitness values observed at the population, respectively and f_i is the fitness value of the chromosome i.

$$select_i = \frac{f_{max} - f_i}{f_{max} - f_{min}} \tag{4.9}$$

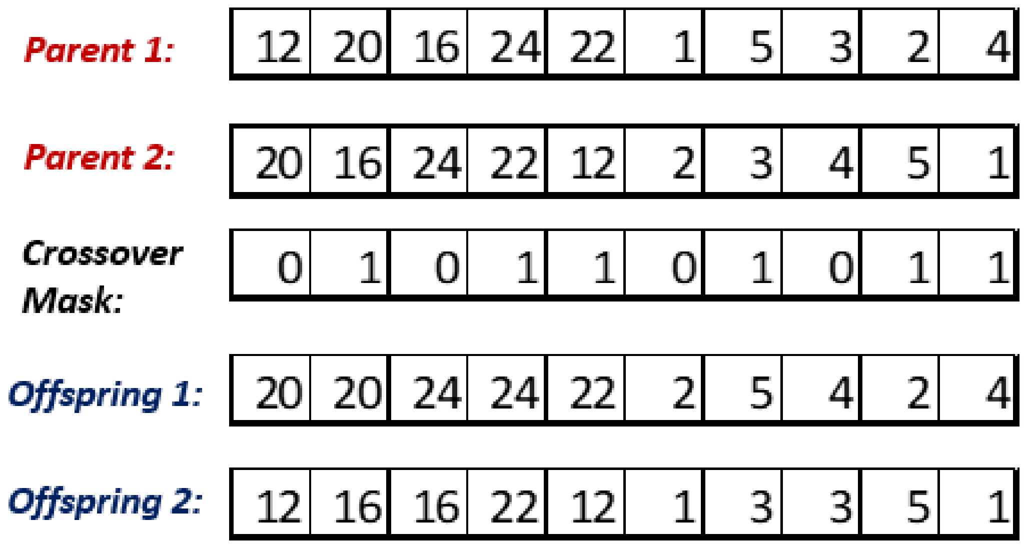
When the $select_i$ values are summed over the chromosomes, the total of these values may exceed one, so we need to normalize these values to calculate the selection probabilities. Assume that $prob_i$ refers to the probability of selection of the chromosome i, its value is calculated using the equation (4.10).

$$prob_i = \frac{select_i}{\sum\limits_{i} select_i} \tag{4.10}$$

After determining selection probabilities for the chromosomes, the discrete cumulative probability distribution is formed. The random numbers are generated to determine which chromosomes will be sent to the mating pool.

4.2.5 Crossover Operators

The mating pool is firstly shuffled to avoid the mating of elite solutions. Then, we matched the parents consecutively from the mating pool. According to the crossover probability P_c, we decided on whether to apply crossover or not. If the decision does not have a crossover, the parents directly become as offsprings. For the rest of them, we created $2n$ size vector (the size is the same with the size of the chromosome) that includes randomly selected 0-1 entries. It is the binary crossover mask to select the genes from the parents. If the entry is 0, it means that the related cells of the offspring 1 and 2 should come from second and first parent, respectively. If it is 1, it should be vice versa. To illustrate that, we provide a small example in Figure 4.5.



Parent 1:	12	20	16	24	22	1	5	3	2	4
Parent 2:	20	16	24	22	12	2	3	4	5	1
Crossover Mask:	0	1	0	1	1	0	1	0	1	1
Offspring 1:	20	20	24	24	22	2	5	4	2	4
Offspring 2:	12	16	16	22	12	1	3	3	5	1

Figure 4.5: Example of a crossover operation

To illustrate, crossover mask begins with 0. Therefore offspring-1 gets the value of 20 which is the first element of parent-2 and offspring-2 gets the value of 12 which is the first element of parent-1. Since the second element of crossover mask is 1, parent-1 sends its second element to offspring-1 and parent-2 sends its second element to offspring-2.

4.2.6 Mutation Operators

Mutation probability is fixed to some value in the algorithm first. Then, offsprings, that will be exposed to the mutation, will be selected according to the probability of mutation, P_m. Genes of the selected chromosomes for the mutation are changed. Number of genes to be mutated is calculated via $Percent_{mutation}$ which denotes the percentage of genes to be mutated in a chromosome. While changing a gene, the feasibility is preserved by changing the cells from the feasible set. Figure 4.6 shows the example of the mutation operation in a chromosome.

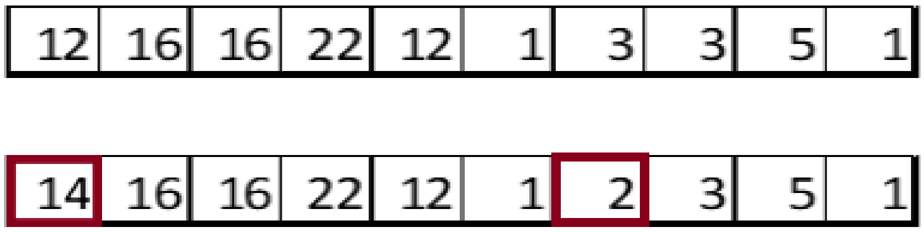


Figure 4.6: Mutation example

In this small example, the diameter size of item-1 is altered from Ø12 to Ø14 and cutting time of item-2 is mutated from 3 to 2. In order to determine levels of P_m and $Percent_{mutation}$, we tested different levels of these parameters in the experiments at Chapter 5.

4.2.7 Forming Population for Next Generation

After the crossover and mutation operations, we directly take the offsprings to the next generation without comparing the fitness values of the offsprings and parents. Consequently, parents are killed even if they have better fitness value so that the algorithm may explore different solutions, and premature convergence can be prevented.

4.2.8 Termination Criteria

We define two different termination criteria in the genetic algorithm. First one is the time limit, if the algorithm reaches the stated time limit, it stops. The second one is the convergence, we look at the last 100 iterations and take averages of the first 10 and last 10 solutions, separately. If the improvements between the two sets are smaller than 0.001%, the algorithm stops. It is enough to satisfy one of the termination criteria for ending the algorithm. At the end of the iterations, the CSP subproblems are solved using CPLEX, rather than calling the first-fit algorithm, in such a way that each chromosome is solved in a parallel manner, and the best solution found is saved.

4.2.9 Pseudocode of the Genetic Algorithm

Algorithm 3 A Genetic Algorithm for 1.5MPCSP

```
    Step 1: Initialize Parameters
        Initialize the parameter values: $P_c, P_m$, $Percent_{mutation}$, $Percent_{best}$,
    $Percent_{elite}$, population size
    Step 2: Generate Initial population
        Generate elite solution with base heuristic algorithm.
        (1- $Percent_{elite}$) of the population is created from the feasible set,
    $Percent_{elite}$ of the population is formed with elite solutions.
    Step 3: Fitness Value Calculations
        Calculate the fitness value for each individual by using first-fit algorithm
    Step 4: Parent Selection
        Take the $Percent_{best}$ of the best individuals to the mating pool directly
        Calculate the probability of selection for each individual
        Determine the rest of the parents according to the probability of
        selection
    Step 5: Crossover Operations
        Shuffle parents
        Consecutively match the parents
        According to probability of crossover $p_c$, determine the parents that will
    enter crossover operations
        Generate offsprings
    Step 5: Mutation Operations
        Determine the offsprings for mutation
        Change the genes of the mutated offsprings
    Step 6: Forming Population for the Next Generation
        Transfer the offsprings directly to the next generation
    Step 7: Termination
        if solution time= time limit OR convergence criterion is satisfied
            Stop, and call CPLEX to solve 1-D CSPs
            Report the best solution found
        else return to Step $3$
```

CHAPTER 5

COMPUTATIONAL RESULTS

In this chapter, we present the experimental results of solution approaches. Different sizes and types of problem instances are used in computational experiments. Instances are derived from the real construction projects developed for the industry. The performances of Kantorovich, arc-flow, and reflect formulations are tested for the 1.5-dimensional single-period cutting stock problem. The performances of reflect formulation and genetic algorithm are compared for the 1.5-dimensional multi-period cutting stock problem. Parameters of the genetic algorithm are also fine-tuned for improving the solution quality. Moreover, the contributions of 1.5-D CSP and 1.5-D MPCSP compared to the 1-D CSP are examined and discussed in the last section.

5.1 Data Collection & Instance Generation

In order to analyze our solution approaches realistically, we benefitted from the real construction projects taken from the industry. We use five different projects from different application areas such as hospitals, apartments, business centers, and public buildings. Each project is different in terms of sizes and properties. Four main properties of projects that make projects different are the number of items, the number of diameter types, the minimum item length, and the number of different lengths. Table 5.1 shows the properties of these projects. Intuitively, one expects that an increase in the number of items and the number of diameter types make difficult to solve problems. On the other hand, the number of different lengths and the minimum length used in projects also affect the problem complexity since these parameters have an influence upon network size. To clarify, if the number of different length in an instance increases, the network size grows accordingly. Therefore, it affects solution time and quality significantly. On the other hand, the minimum length of the instance has similar effects on network size. To illustrate, assume that there are two identical instances, except their minimum length of items. The minimum lengths of items observed in those instances are 50 cm and 100 cm, respectively. The instance, whose minimum length is 50 cm, has a larger network size compared to the instance, whose minimum length is 100 cm. It is because of that decrease in the lengths of items in an instance causes the network to possess more vertices and arcs.

Table 5.1: Properties of instances

Project Name	**# of Item**	**# of Different Dia. Size**	**Min. Item Length**	**# of Different Length**
Project-1	1835	6	20 cm	268
Project-2	1940	12	40 cm	254
Project-3	1540	7	62 cm	126
Project-4	11313	8	60 cm	457
Project-5	2900	5	43 cm	218

To compare solution approaches, we derived instances from five real projects in different application areas. Instances are generated with 100, 250, 500, and 1000 items by drawing randomly from each project separately. Moreover, instances with 2000 items are derived from both project-4 and project-5. Finally, instances with 5000 and 10000 items are created using only project-4. Five instances are generated for every size for all projects. Table 5.2 shows the number of instances created from every project and for different sizes. All projects are designed by a structural designer with 0.01 meters precision. To illustrate, item lengths increase in a way that 3.4 m, 3.41 m, 3.42 m, etc. Since stock length is 12 meters, the maximum amount of different length types can reach 1200 theoretically with this precision.

Instances are generated for single-period and multi-period problems separately. We assume that there are 10 periods, and inventory holding rate is 0.01 for multi-period problems. Since information about the demanded periods for items is not available for the data we collect, we generate period information randomly for each instance.

Table 5.2: Number of instances used in the experiments

	Number of Items							
Project Name	**100**	**250**	**500**	**1000**	**2000**	**5000**	**10000**	**Total**
Project-1	5	5	5	5	0	0	0	20
Project-2	5	5	5	5	0	0	0	20
Project-3	5	5	5	5	0	0	0	20
Project-4	5	5	5	5	5	5	5	35
Project-5	5	5	5	5	5	0	0	25

5.2 Computational Results for the Single-Period Cutting Stock Problem with Diameter Conversion in the Construction Industry

We tested exact solution approaches, which are the arc-flow and reflect formulations, on 120 instances for the single-period cutting stock problem with diameter conversion in the construction industry. Implementation is built using C++ and CPLEX 12.8.0 Concert Technology with Intel (R) Core (TM) i7-4790 CPU @3.10 GHz and 16GB RAM. To solve each instance, two hours of time limit is given to CPLEX. Note that the Kantorovich formulation is excluded from the experiment because it cannot solve instances larger than 50 by using Scip-GCG solver that applies B&P algorithm according to preliminary results.

In five different projects, the performances of arc-flow and reflect formulations are compared in the case of single-period CSP with diameter conversion in terms of solution quality. To evaluate the solution quality, average gap values of CPLEX are compared. In Table 5.3, the computational results are provided. The number of items is varied while comparing these two formulations. In some instances, CPLEX could not find any integer solution within the time limit. Therefore, the number of instances solved are recorded for every five instances which are generated for every project and every size of the number of items.

According to the computational results presented at Table 5.3, average gap values are lower in the reflect formulation in all projects and all sizes for the number of items compared to the arc-flow formulation. When the number of items increases, the average gap values also rise in both formulations as expected. Furthermore, if no upper bound value is obtained within the time limit, we classified those instances "not solved". A few instances with larger item size cannot be solved by both formulations. On the other hand, the reflect formulation outperforms the arc-flow formulation in terms of average gap values and the number of instances solved in each case. To illustrate this, the reflect formulation can solve all instances with 0.09% average gap value in Project-5 where the number of items is 2000 whereas the arc-flow formulation cannot solve any of these instances within the time limit. Moreover, the total numbers of instances solved are 113 and 72 for the reflect and arc-flow formulations, respectively. Finally, we suggest that the decision maker of the problem can use the reflect formulation in order to obtain better solutions for 1.5-D CSP instead of Kantorovich and arc-flow formulations. As a result, we did not make use of the arc-flow formulation in solving 1.5-D MPCSP.

5.3 Computational Results for the Multi-Period Cutting Stock Problem with Diameter Conversion in the Construction Industry

The reflect formulation and genetic algorithm approaches are tested experimentally on 120 instances for the multi-period cutting stock problem with diameter conversion in the construction industry. Implementation is built using C++ and CPLEX 12.8.0 Concert Technology with Intel (R) Core (TM) i7-4790 CPU @3.10 GHz and 16GB RAM. To solve each instance, two hours of time limit is given to CPLEX.

We also propose to round-up input length data for the reflect formulation, since it becomes weak when the stock length capacity increases due to its pseudo-polynomial structure. Lengths of items are generally designed with 0.05 m precision such as 3.4 m, 3.45 m, 3.5 m, etc. However, they can be rarely designed with 0.01 m precision such as 3.41 m, 3.42 m, 3.43 m, etc. In order to reduce network size, we propose to round-up lengths of items, whose magnitudes are assigned with 0.01 m precision, to the nearest potential length. To illustrate, suppose that lengths of items are 3.41 m, 3.42 m, 3.45 m, and 3.46 m. Their rounded final lengths will be 3.45 m, 3.45 m, 3.45 m, and 3.5 m. Consequently, rounding items' lengths provides to reduce network size in terms of both numbers of arcs and nodes. Although rounding procedure sacrifices

Table 5.3: Comparison of reflect and arc-flow formulations for 1.5-D CSP

		Reflect		**Arc-flow**	
Project Name	**# Items**	**# Solved**	**Avg. Gap (%)**	**# Solved**	**Avg. Gap (%)**
Project-1	100	5	0.10	5	0.47
Project-1	250	5	0.13	2	0.46
Project-1	500	5	0.13	0	NA
Project-1	1000	5	0.13	0	NA
Project-2	100	5	0.02	5	0.16
Project-2	250	5	0.02	4	0.78
Project-2	500	5	0.09	2	4.56
Project-2	1000	5	0.20	3	45.14
Project-3	100	5	0.01	5	0.01
Project-3	250	5	0.01	5	0.03
Project-3	500	5	0.03	5	0.10
Project-3	1000	5	0.06	5	0.22
Project-4	100	5	0.07	5	0.17
Project-4	250	5	0.05	5	0.85
Project-4	500	5	0.03	5	0.21
Project-4	1000	5	0.12	2	0.70
Project-4	2000	5	0.82	0	NA
Project-4	5000	3	38.43	0	NA
Project-4	10000	0	NA	0	NA
Project-5	100	5	0.10	5	0.33
Project-5	250	5	0.11	5	0.36
Project-5	500	5	0.11	3	0.56
Project-5	1000	5	0.09	1	2.22
Project-5	2000	5	0.09	0	NA
Total Instances Solved		**113**		**72**	

solution quality in advance, it paves the way for the reflect formulation becomes more powerful and capable to reach better solution quality compared to the original form. As a result, we included both original and rounded forms of the reflect formulation in the experiments.

In Table 5.4, the summary of the computational results are presented for the multi-period 1.5-D cutting stock problems. The performances of the reflect formulation, its rounded version, and genetic algorithm are compared. Before conducting these experiments, the parameters of the genetic algorithm are finetuned as it will be explained in Section 5.4. The experiments with the genetic algorithm are conducted with the selected algorithm parameters. The average gap values of the genetic algorithm and rounded version are calculated using the best lower bounds provided by CPLEX while solving the reflect formulation.

In these experiments, the same five projects are used and five instances are created for each project and item size. In total, 120 instances are used to compare solution approaches. Instances with 100 items are not examined by rounded version and genetic algorithm since they can be easily solved by the reflect formulation satisfactorily. For the reflect formulation and its rounded version, the average gap values and the number of instances solved are recorded. Since the genetic algorithm can find a solution for every case, the average gap values and average run times are used to make a reasonable comparison.

As expected, when the number of items increases, both reflect formulation and its rounded version have difficulties in solving instances. On the other hand, genetic algorithm spends much more time to satisfy termination criteria in the case where we have a large number of items. In the comparison of reflect formulation and its rounded version, the reflect-rounded can solve a larger number of instances. To illustrate that, rounded version solves 83 instances out of 95 instances. However, the reflect formulation enables to solve 30 of them from the same set of instances.

The genetic algorithm can reach small gap values in a shorter amount of time. When the number of items is small (e.g. when the number of items is equal to 250), rounded version finds lower gap values compared to genetic algorithm. On the contrary, the genetic algorithm outperforms both versions of the reflect formulation at instances

Table 5.4: Comparison of the reflect formulation and genetic algorithm for 1.5-D MPCSP

		Reflect		**Reflect Rounded**		**Genetic Alg.**	
Project Name	**# of Items**	**# Solved**	**Avg. Gap (%)**	**# Solved**	**Avg. Gap (%)**	**Avg. Run Time (s)**	**Avg. Gap (%)**
Project-1	100	5	1.13	-	-	-	-
Project-1	250	1	25.92	5	5.18	168.1	5.53
Project-1	500	0	NA	5	19.20	421.7	5.39
Project-1	1000	0	NA	5	55.82	1291.9	5.35
Project-2	100	5	0.17	-	-	-	-
Project-2	250	3	20.47	5	1.73	154.2	2.64
Project-2	500	0	NA	5	11.46	483.8	3.81
Project-2	1000	1	66.33	2	67.94	992.7	3.73
Project-3	100	5	0.01	-	-	-	-
Project-3	250	5	0.05	5	0.04	418.2	3.68
Project-3	500	5	0.94	5	0.25	735.1	3.78
Project-3	1000	2	2.22	5	11.60	1635.1	3.31
Project-4	100	5	0.07	-	-	-	-
Project-4	250	2	1.01	5	1.12	143.1	7.86
Project-4	500	2	49.61	5	8.29	559.6	5.29
Project-4	1000	0	NA	5	41.85	1456.8	5.68
Project-4	2000	4	66.00	5	63.25	3601.6	4.81
Project-4	5000	1	63.55	4	58.36	5826.9	4.62
Project-4	10000	0	NA	4	49.71	6668.2	4.87
Project-5	100	5	0.65	-	-	-	-
Project-5	250	4	13.70	5	0.91	199.9	4.40
Project-5	500	0	NA	5	25.04	386.1	4.05
Project-5	1000	0	NA	1	75.24	1141.0	2.90
Project-5	2000	0	NA	2	73.39	2966.3	2.50

with a larger number of items. For example, when the number of items is 1000, the average gap values for the reflect-rounded and genetic algorithm are 55.82% and 5.35% for Project-1, respectively. To conclude, the reflect-rounded can be used in a small number of items whereas the genetic algorithm is a powerful tool to solve large size 1.5D-MPCSP problems to reach reasonable gap values in shorter computation times.

As can be seen from the computational results, each project shows different attitudes in terms of solution quality and the number of instances solved. It is because of that each project has different properties in terms of four main characteristic which are number of items, the number of different diameter size, the minimum length of items, and the number of different length sizes in the instances as seen in Table 5.1. In order to show how these four properties affect solution quality, we analyze computational results. Firstly, it is obvious that when the number of items increases solving instances is getting harder. To illustrate, the number of instances solved is decreasing and the average gap value increases if the number of items increases for all projects. Secondly, it can be observed that an increase in the number of different diameter sizes has no major effect on solving instances. For example, projects 2 and 5 have similar properties except their number of different diameter sizes as 12 and 5, respectively. However, the total numbers of instances solved are equal for both projects and average gap values do not differ significantly. Thirdly, it can be deduced that decrease in the size of minimum item length will result in difficulties in solving instances since it causes network size to grow considerably. To illustrate, projects 1 and 5 have similar properties except their minimum item length's sizes as 20 cm and 43 cm, respectively. While the total numbers of instances solved are six and nine, the average gap values of Project-1 are larger than Project-5. Finally, the number of different item length sizes has an effect on solution quality as well. Project-3 and Project-4 have similar properties except their number of different item length sizes which are 126 and 457, respectively as seen at Table 5.1. If the average gap values and the total numbers of instances are compared, it can be easily seen that the instances of Project-4 are more difficult to solve than instances of Project-3. Therefore, we can conclude that an increase in the number of different item length sizes has a negative impact on solution quality.

5.4 Parameter Selection for Genetic Algorithm

The performance of a genetic algorithm is directly dependent on the parameter values. There are six parameters: crossover probability, mutation probability, $Percent_{elite}$ percentage of elite solution at the initial population, population size, $Percent_{mutation}$ indicates the percentage of genes in a chromosome to be mutated, and the $Percent_{best}$ that is the percentage of transferring the best parents to the mating pool directly.

To determine the best parameter values, we tested different parameter settlements on three instances for every project and for different sizes. In order to reduce the time spent on experiments, we restricted data set with three instances from every project and their item sizes up to 2000, instead of using five instances as in Sections 5.2 and 5.3. Since complete enumeration for parameters and their levels will lead to trying a large number of instances, we also restricted the number of parameters and their levels to be enumerated. Ultimately, the full-factorial experiment design is conducted on four parameters' levels. Table 5.5 shows the details of the experiment. Therefore, 36 different settlements, where the population size and the $Percent_{mutation}$ are fixed to 200 and $\%10$ respectively, are tested on 72 instances. Table 5.6 shows the average gap values for every settlement.

Table 5.5: Genetic algorithm parameters' levels to be enumarated ($\%$)

Mutation Prob.	6.0	12.0	18.0
$\boldsymbol{Percent_{best}}$	15.0	30.0	
Crossover Prob.	75.0	85.0	95.0
$\boldsymbol{Percent_{elite}}$	15.0	30.0	

After the full-factorial experiment on four parameters' level, we fixed their values ($Percent_{best} = 12\%$, mutation probability = 15%, crossover probability = 95%, $Percent_{elite}$ = 15%), where average of the gap values are minimum. After selecting mutation probability as 15%, another decision arises for the mutation technique in the genes of the chromosome.

Table 5.6: Average gap values in different parameter settlement

Settlement	**Mutation ($\%$)**	$Percent_{best}(\%)$	**Crossover ($\%$)**	$Percent_{elite}(\%)$	**Gap ($\%$)**
1	6	15	75	15	4.58
2	6	15	75	30	4.60
3	6	15	85	15	4.52
4	6	15	85	30	4.55
5	6	15	95	15	4.53
6	6	15	95	30	4.55
7	6	30	75	15	4.84
8	6	30	75	30	4.87
9	6	30	85	15	4.80
10	6	30	85	30	4.78
11	6	30	95	15	4.69
12	6	30	95	30	4.69
13	12	15	75	15	4.58
14	12	15	75	30	4.53
15	12	15	85	15	4.50
16	12	15	85	30	4.54
17	**12**	**15**	**95**	**15**	**4.46**
18	12	15	95	30	4.52
19	12	30	75	15	4.65
20	12	30	75	30	4.71
21	12	30	85	15	4.64
22	12	30	85	30	4.59
23	12	30	95	15	4.62
24	12	30	95	30	4.60
25	18	15	75	15	4.64
26	18	15	75	30	4.61
27	18	15	85	15	4.59
28	18	15	85	30	4.63
29	18	15	95	15	4.71
30	18	15	95	30	4.70
31	18	30	75	15	4.63
32	18	30	75	30	4.64
33	18	30	85	15	4.61
34	18	30	85	30	4.58
35	18	30	95	15	4.54
36	18	30	95	30	4.52

In this study, we decided to vary the percentage of the genes of the chromosome to be mutated. To select the $Percent_{mutation}$ value, the computational experiments are conducted. Three different levels of $Percent_{mutation}$ (5%, 10%, and 20%) are tested on the same 72 instances to set its value. Table 5.7 shows that average gap values of $Percent_{mutation}$ levels.

Table 5.7: Average gap values in different mutation percentages

$\boldsymbol{Percent_{mutation}}$(%)	**Avg. Gap(%)**
5	4.47
10	**4.46**
20	4.62

According to the results of the experiments, it is concluded that the average gap values do not change significantly when the $Percent_{mutation}$ value is varied. The minimum average gap value is obtained when $Percent_{mutation}$ value is 10%. Therefore, we selected it as 10% in subsequent experimentation.

Finally, experiments are carried out for setting an appropriate value for the population size. The genetic algorithm is tested with four different population sizes as 200, 500, 1000, and 2000. As it can be seen from Table 5.8, increasing population size will lead to increase solution time and let the algorithm explore better solutions. However, the algorithm can exploit good solutions at small population sizes. According to our computational experiments, the best average gap value, which is 4.38%, can be reached when population size is 2000.

Table 5.8: Average gap values and run times in different population sizes

Population Size	**Avg. Run Time (s)**	**Avg. Gap (%)**
200	453	4.46
500	504	4.54
1000	576	4.51
2000	**1047**	**4.38**

In order to see the effects of population sizes, graphs 5.1, 5.2, 5.3, and 5.4 show the

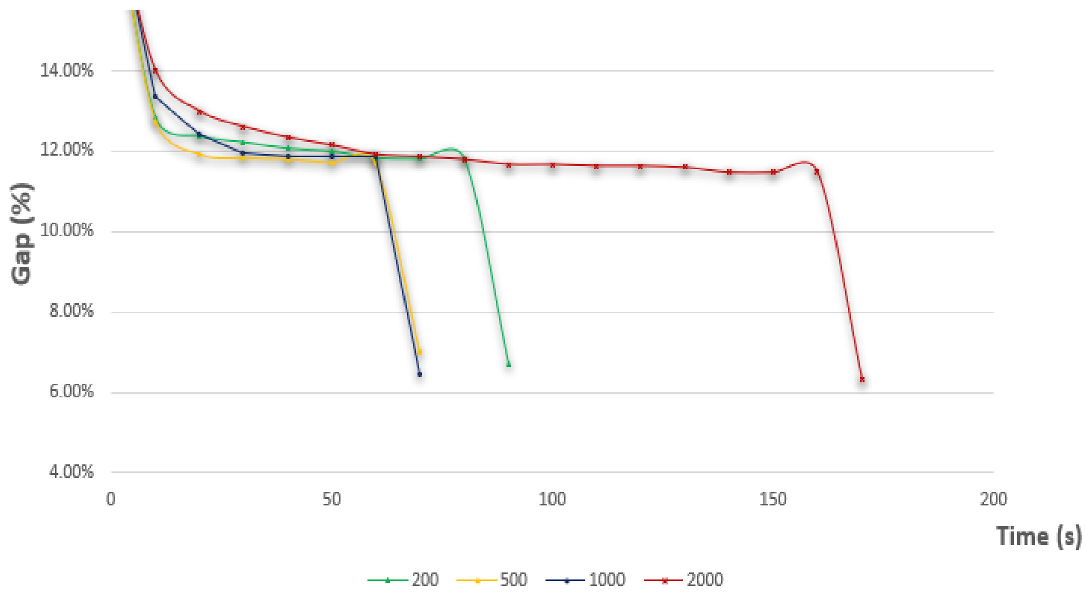


Figure 5.1: Gap values over time when the number of item is 250 for four different population sizes

gap values over time for different item sizes as 250, 500, 1000, and 2000. In each graph, four different colored lines (green, yellow, blue, and red) can be observed which refer to the behavior of the average gap values over time in four different population sizes (200, 500, 1000, and 2000), respectively.

At first glance, it draws the attention that there are sharp decreases in the gap values at the last iteration of the algorithm for every population size. When the algorithm remains at close gap values during a large number of iterations, CPLEX is called for solving chromosomes of the last population instead of using the first-fit algorithm. As a result, it finds optimal solutions for individual 1-D cutting stock problems by using original reflect formulation so that the algorithm provides better gap values.

It can be easily observed that the algorithm explore better solutions when the population size is at its largest level. Although there are no major differences among solution qualities, the largest level of the population size provides to obtain minimum average gap values for all item sizes. On the other hand, solution time increases because the algorithm converges later and solves more chromosomes for every iteration. In the lowest value of the population size, the genetic algorithm exploits solutions that decrease the gap value sharply. On the other hand, the algorithm converges early and

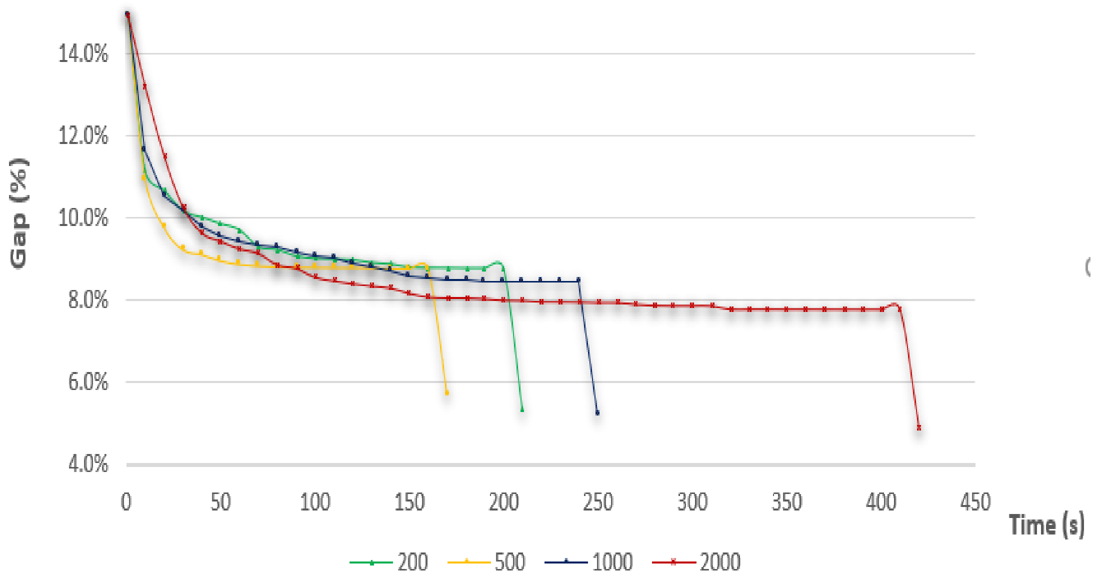


Figure 5.2: Gap values over time when the number of item is 500 for four different population sizes

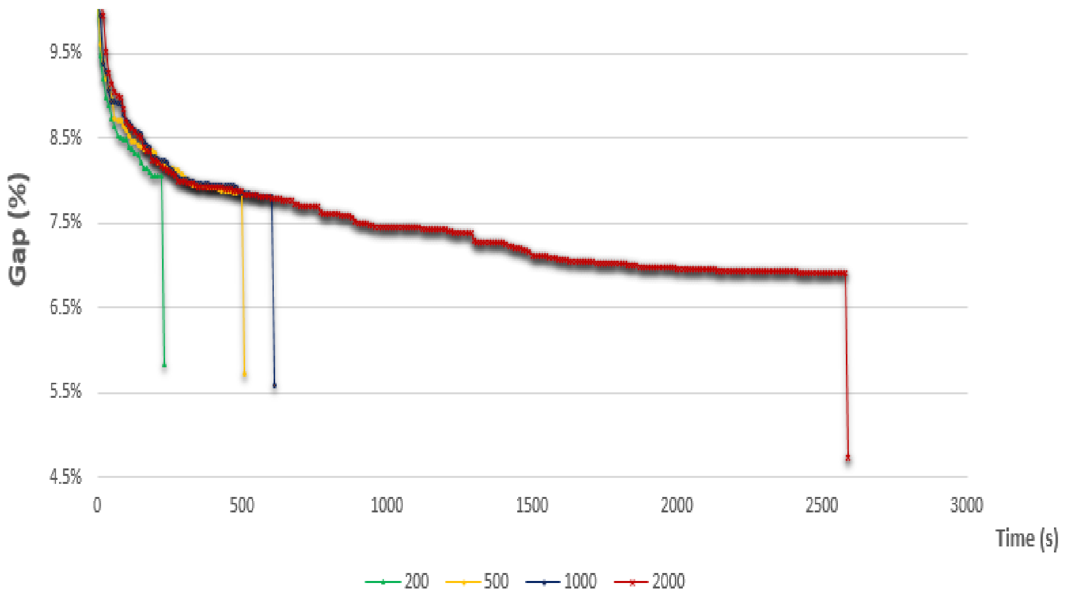


Figure 5.3: Gap values over time when the number of item is 1000 for four different population sizes

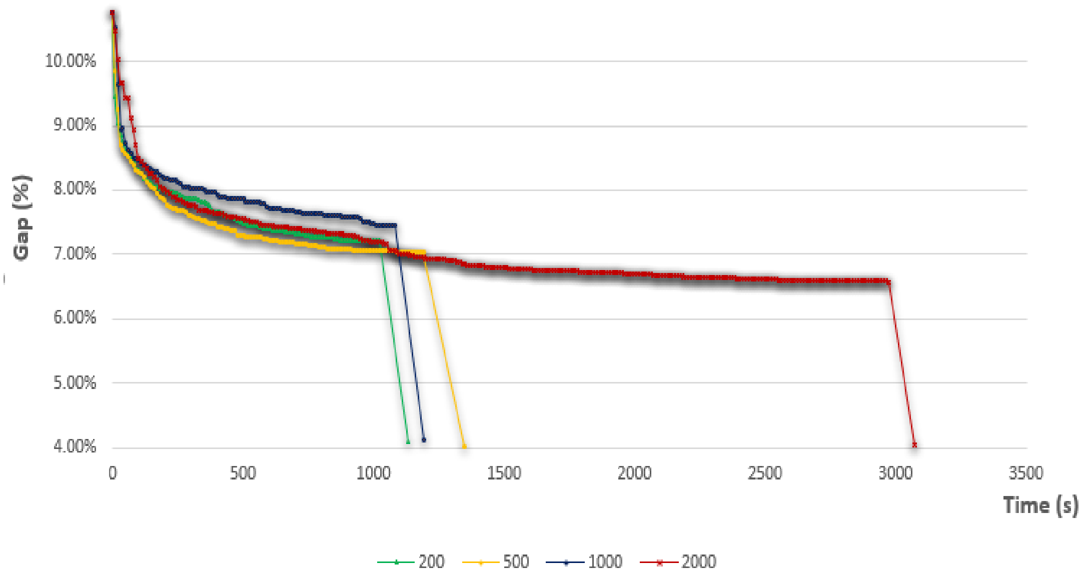


Figure 5.4: Gap values over time when the number of item is 2000 for four different population sizes

cannot explore better solutions.

5.5 Contributions of 1.5-Dimensional and Multi-Period Cutting Stock Problem

In the last section of computational results, we made comparisons among 1-D CSP, 1.5-D CSP, and 1.5-D MPCSP to demonstrate the contribution of 1.5-dimensional and multi-period structures of CSP. We prepare an experimental set using original data of five real projects. Properties of these projects are conserved as shown in Table 5.1. For the single-period problem, we tried to answer the following questions: What if diameters are not converted and how much gain do we get by converting diameter sizes? We solved five real projects' instances without considering conversion. Since each item will be cut from its original diameter, every diameter size forms 1-dimensional cutting stock problems. As a result, we solve 6, 12, 7, 8, and 5 (which are the number of different diameter sizes of projects) different and independent 1-D cutting stock problems for projects 1-5, respectively. Then, these real projects' instances are solved considering diameter conversion. Experiments are conducted within five hours of time limit. Table 5.9 shows the computational results. Since solving Project-

4 cannot be solved satisfactorily, we solve Project-4 by rounding their length sizes as mentioned in Section 5.3. According to the computational results, taking diameter conversion into account when preparing a cutting plan of rebars offers significant advantages. For instance, the decision maker can save 6 % from using rebar if Project-3 is solved considering diameter conversion.

Table 5.9: 1.5-D CSP % gain comparison to 1-D CSP

	Single-Period		
	% Gap		**Gain Compared to 1-D CSP (%)**
Project Name	**Reflect**	**Reflect-Rounded**	
Project-1	0.1	-	5.6
Project-2	0.1	-	5.9
Project-3	0.1	-	6.0
Project-4	5.5	0.5	4.3
Project-5	0.1	-	0.9

In order to evaluate the contribution of diameter conversion and multi-period structure of the problem together, we prepare another experimental set from the same five real projects. Properties of these projects are conserved as shown in Table 5.1. We tried to answer the following questions: What if diameters are not converted and holding inventory is not considered? How much gain do we get by converting diameter sizes and holding inventory? Therefore, we aimed to solve two different settlements composed of five real projects as if they are 1-D CSP. The first settlement is designed as all items are cut from its original diameter at its demanded period while the second settlement is designed as all items are cut from its original diameter in the first period. As a result, 60, 120, 70, 80, and 50 different and independent 1-D CSPs are solved for projects 1-5, respectively, since there are 10 time periods on those projects for the first settlement. Furthermore, 6, 12, 7, 8, and 5 different and independent 1-D CPSs are solved for projects 1-5, respectively since all items are cut at the beginning for the second settlement. Note that, in the first experiment, items are cut at their demanded periods and no inventory is carried to the next period. In this settlement, the decision maker misses the advantage of generating efficient patterns by holding inventory. On the other hand, in the second experiment items are cut in the first period and demands of later periods are supplied from the inventory. Although the decision maker can benefit from generating efficient cutting patterns by cutting all items at the first period, inventory holding cost incurred without compromising contribution of generating efficient patterns. Finally, instances are solved with the genetic algorithm, reflect formulation, and its rounded version considering diameter conversion and holding inventory within five hours of time limit. Table 5.10 shows the computational results. It can easily be observed that instances cannot be solved successfully using the reflect formulation and its rounded version. Therefore, we benefitted from the results of the genetic algorithm to make a comparison. As a result, holding inventory and converting diameters provides savings up to 7.3% from using rebars compared to the case where all items are cut at their demanded periods. Besides, saving can reach to 6.8% compared to the case where all items are cut at the first period. Furthermore, it can be concluded that these two strategies do not outperform each other since savings vary according to the projects as can be seen from the results. To illustrate, while the strategy of cutting all items in their demanded periods gives better results for projects 3-5, the strategy of cutting all items in the first period gives better results for projects 1-2. To conclude, converting diameter sizes and holding inventory provide cost reduction from using rebar, but the trade-off between generating efficient cutting patterns and holding inventory should be exploited in order to obtain better results.

Table 5.10: 1.5-D MPCSP % gain comparison to 1-D CSP

Multi-Period					
	% Gap			**Saving Compared to 1-D CSP if all items are cut at their demanded periods (%)**	**Saving Compared to 1-D CSP if all items are cut at the first period (%)**
Projects	**Reflect**	**Reflect-Rounded**	**Genetic**		
Project-1	Not Solved	Not Solved	4.1	5.6	5.3
Project-2	Not Solved	Not Solved	2.7	7.3	6.6
Project-3	Not Solved	12.5	3.1	5.6	6.8
Project-4	Not Solved	Not Solved	4.4	0.5	4.0
Project-5	Not Solved	Not Solved	1.6	0.3	3.6

CHAPTER 6

CONCLUSION

The construction industry has abundant managerial aspects that could be improved by using Operations Research. In many countries, the share of the construction industry constitutes a significant part of gross domestic product. Therefore, effective planning of its activities provides valuable reductions in the expenditures.

Reinforcement steel bar is one of the major cost components in the construction industry. The 1-dimensional cutting stock problem has been widely studied in the literature. It enables to generate efficient cutting patterns when it is used for rebars. However, convertibility of diameters of rebars provides the decision maker to create better cutting patterns in terms of trim loss. In this thesis, we define a new problem that arises in the construction industry considering both decisions of selecting appropriate diameter sizes and generating cutting patterns. Since there is a diameter dimension in addition to the length dimension, it differs from the 1-D CSP. However, choosing the appropriate size of this dimension is also part of the decision process. Therefore, the problem also differs from the 2-D CSP. It can be defined as open dimensional or the 1.5-dimensional cutting stock problem when its similar problems in the literature are considered. Furthermore, we also take into account the time dimension since the need for rebars occurs at different times for large and multiple building projects. Therefore, the trade-off between holding inventory and trim loss should be exploited in order to reduce the costs of construction projects. We defined this problem as the 1.5-dimensional multi-period cutting stock problem.

To solve the 1.5-D CSP, we considered three different formulations namely, Kantorovich, arc-flow, and reflect. Although LP relaxation of the Kantorovich formulation is weak, it facilitates to understand the problem structure. Since arc-flow and reflect formulations are network-based, efficient graphs should be generated in order to obtain better solution qualities. We modified these arc-flow and reflect formulations according to the 1.5-dimensional structure. Since the reflect outperforms the other formulations, we benefitted from it to model 1.5-D MPCSP. However, it is not capable of obtaining satisfactory results for moderate and large sizes of instances. Furthermore, we also design a genetic algorithm to reach reasonable gap values for moderate and large sizes of instances.

In order to answer our research questions, we made computational experiments using real data of the construction projects. Scrutiny of the computational results shows that properties of the projects such as the number of items, minimum item length, etc. have significant effects on solution quality. In addition, the parameters of genetic algorithm also affect solution quality as well. As an alternative solution approach, we propose rounded-up length sizes of instances in order to reduce network sizes so that the reflect formulation becomes more powerful to attack the 1.5-D CSP and 1.5-D MPCSP. It can be concluded that the reflect formulation is powerful of solving a large majority of instances for the 1.5-D CSP. However, it is not capable of obtaining satisfactory results on solving the moderate and large size instances for the 1.5-D MPCSP. Thus, we propose the rounded version of the reflect formulation for the small and moderate size instances and a genetic algorithm that reaches reasonable gap values for larger instances. Lastly, we conducted an experiment to figure out the contribution of diameter conversion and holding inventory. The results show that decision maker can obtain cost savings up to 6% and 7.3% for 1.5-D CSP and 1.5-D MPCSP, respectively.

As future research direction, transportation issue can be combined with these problems. Construction companies may conduct more than one projects simultaneously or consecutively. The need for rebars may occur at different locations and different times. Therefore, trade-off among logistic costs, inventory holding costs, and trim loss can be exploited to reduce costs of construction companies.

Multiple stock length situation can be considered as another future research direction. Although rebars are usually sold in 12 meters in the market, some special contracts can be made for different lengths of rebars. Consequently, better cutting patterns can be obtained using several stock length. However, it causes companies to pay more price for non-standard products. As a result, there is a trade-off between generating good patterns and cost of multiple stock length. The decision maker may accept to pay more for non-standard products in order to reduce cost of trim loss.

Labor cost can be also combined with the problems considered in thesis. Although converting diameter sizes is beneficial for the industry, increasing the number of stocks used in cutting processes may lead to using more labor. Therefore, more comprehensive study can be conducted by considering conversion of diamater sizes, generation of cutting patterns and labor at the same time.

REFERENCES

[1] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European journal of operational research*, vol. 183, no. 3, pp. 1109–1130, 2007.

[2] Y. T. Han and S. Y. Chang, "A subset sum approach to coil selection for slitting.," *International Journal of Industrial Engineering*, vol. 22, no. 3, 2015.

[3] X. Song, C. Chu, Y. Nie, and J. A. Bennell, "An iterative sequential heuristic procedure to a real-life 1.5-dimensional cutting stock problem," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1870–1889, 2006.

[4] J. V. De Carvalho, "Exact solution of bin-packing problems using column generation and branch-and-bound," *Annals of Operations Research*, vol. 86, pp. 629–659, 1999.

[5] M. Delorme and M. Iori, "Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems," in *Technical Report*, DEI "Guglielmo Marconi" Alma Mater Studiorum Università di Bologna, Italy, 2017.

[6] T. C. Association, "Turkish contracting in the international market." https://www.tmb.org.tr/doc/file/YDMHmarch2017.pdf, 2017. Accessed:March 2019.

[7] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Management science*, vol. 6, no. 4, pp. 366–422, 1960.

[8] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

[9] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations research*, vol. 8, no. 1, pp. 101–111, 1960.

[10] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.

[11] R. W. Haessler, “A procedure for solving the 1.5-dimensional coil slitting problem,” *AIIE Transactions*, vol. 10, no. 1, pp. 70–75, 1978.

[12] R. N. Gasimov, A. Sipahioglu, and T. Saraç, “A multi-objective programming approach to 1.5-dimensional assortment problem,” *European Journal of Operational Research*, vol. 179, no. 1, pp. 64–79, 2007.

[13] Z. S. M. Nadoushani, A. W. Hammad, J. Xiao, and A. Akbarnezhad, “Minimizing cutting wastes of reinforcing steel bars through optimizing lap splicing within reinforced concrete elements,” *Construction and Building Materials*, vol. 185, pp. 600–608, 2018.

[14] C. Zheng and M. Lu, “Optimized reinforcement detailing design for sustainable construction: Slab case study,” *Procedia Engineering*, vol. 145, pp. 1478–1485, 2016.

[15] V. Benjaoran, N. Sooksil, and M. Metham, “Effect of demand variations on steel bars cutting loss,” *International Journal of Construction Management*, vol. 19, no. 2, pp. 137–148, 2019.

[16] K. C. Poldi and S. A. de Araujo, “Mathematical models and a heuristic method for the multiperiod one-dimensional cutting stock problem,” *Annals of Operations Research*, vol. 238, no. 1-2, pp. 497–520, 2016.

[17] H. Reinertsen and T. W. Vossen, “The one-dimensional cutting stock problem with due dates,” *European Journal of Operational Research*, vol. 201, no. 3, pp. 701–711, 2010.

[18] G. M. Melega, S. A. de Araujo, and R. Jans, “Classification and literature review of integrated lot-sizing and cutting stock problems,” *European Journal of Operational Research*, vol. 271, no. 1, pp. 1–19, 2018.

[19] C. Le Hesran, A.-L. Ladier, V. Botta-Genoulaz, and V. Laforest, “Operations scheduling for waste minimization: A review,” *Journal of Cleaner Production*, 2018.

[20] M. Delorme, M. Iori, and S. Martello, “Bin packing and cutting stock problems: Mathematical models and exact algorithms,” *European Journal of Operational Research*, vol. 255, no. 1, pp. 1–20, 2016.

[21] M. Rao, “On the cutting stock problem,” *Journal of the Computer Society of India 7*, pp. 35–39, 1976.

[22] H. Dyckhoff, “A new linear programming approach to the cutting stock problem,” *Operations Research*, vol. 29, no. 6, pp. 1092–1104, 1981.

[23] H. Cambazard and B. O’Sullivan, “Propagating the bin packing constraint using linear programming,” in *International Conference on Principles and Practice of Constraint Programming*, pp. 129–136, Springer, 2010.

[24] F. Brandao and J. P. Pedroso, “Bin packing and related problems: general arc-flow formulation with graph compression,” *Computers & Operations Research*, vol. 69, pp. 56–67, 2016.

[25] O. Marcotte, “An instance of the cutting stock problem for which the rounding property does not hold,” *Operations Research Letters*, vol. 4, no. 5, pp. 239–243, 1986.

[26] G. Scheithauer and J. Terno, “The modified integer round-up property of the one-dimensional cutting stock problem,” *European Journal of Operational Research*, vol. 84, no. 3, pp. 562–571, 1995.

[27] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser, “Solving binary cutting stock problems by column generation and branch-and-bound,” *Computational optimization and applications*, vol. 3, no. 2, pp. 111–130, 1994.

[28] D. Ryan and E. Foster, “An integer programming approach to scheduling,” *Computer Scheduling of Public Transport*, pp. 269–280, 1981.

[29] G. Scheithauer and J. Terno, “A branch&bound algorithm for solving one-dimensional cutting stock problems exactly,” *Applicationes Mathematicae*, vol. 23, no. 2, pp. 151–167, 1995.

[30] P. H. Vance, “Branch-and-price algorithms for the one-dimensional cutting stock problem,” *Computational optimization and applications*, vol. 9, no. 3, pp. 211–228, 1998.

[31] G. Belov and G. Scheithauer, "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting," *European journal of operational research*, vol. 171, no. 1, pp. 85–106, 2006.

[32] S. Eilon and N. Christofides, "The loading problem," *Management Science*, vol. 17, no. 5, pp. 259–268, 1971.

[33] S. Martello, "Knapsack problems: algorithms and computer implementations," *Wiley-Interscience series in discrete mathematics and optimiza tion*, 1990.

[34] A. Scholl, R. Klein, and C. Jürgens, "Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers & Operations Research*, vol. 24, no. 7, pp. 627–645, 1997.

[35] P. Shaw, "A constraint for bin packing," in *International conference on principles and practice of constraint programming*, pp. 648–662, Springer, 2004.

[36] P. Schaus, J.-C. Régin, R. Van Schaeren, W. Dullaert, and B. Raa, "Cardinality reasoning for bin-packing constraint: application to a tank allocation problem," in *Principles and Practice of Constraint Programming*, pp. 815–822, Springer, 2012.

[37] D. Simchi-Levi, "New worst-case results for the bin-packing problem," *Naval Research Logistics (NRL)*, vol. 41, no. 4, pp. 579–585, 1994.

[38] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete applied mathematics*, vol. 28, no. 1, pp. 59–70, 1990.

[39] M. Quiroz-Castellanos, L. Cruz-Reyes, J. Torres-Jimenez, C. Gómez, H. J. F. Huacuja, and A. C. Alvim, "A grouping genetic algorithm with controlled gene transmission for the bin packing problem," *Computers & Operations Research*, vol. 55, pp. 52–64, 2015.

APPENDIX A

Table A.1: Example of a small instance for first-fit algorithm

	Length (cm)	**Demand**
Item 1	900	8
Item 2	625	5
Item 3	575	10
Item 4	400	2
Item 5	375	7
Item 6	300	6
Item 7	225	17
Item 8	125	3

Example of a small instance composing of eight items is used to show that how first-fit decreasing algorithm works. Table A.1 shows the lengths and demands of items in the example instance. The instance is solved with first-fit decreasing algorithm. Obvious inefficiencies are eliminated with an improvement part of the algorithm. The reason why first-fit decreasing produces inefficient patterns is that orders of the items are assigned to the patterns as bulk, not one by one. Therefore, using additional improvement is helpful to achieve better utilization levels of the patterns. Table A.2 and Table A.3 show how beneficial to add improvement part to the algorithm. The total number of stocks used is reduced from 28 to 23 adding improvement part to the algorithm while the optimal solution of this instance is 22. Pseudo-code of the first-fit decreasing algorithm is provided in Algorithm 4.

Table A.2: Example of first-fit algorithm without improvement

Without Improvement					
	# of usage	**item 1**	**item 2**	**item 3**	**Total Length (cm)**
Pattern 1	6	900	300		1200
Pattern 2	5	625	575		1200
Pattern 3	2	575	400	225	1200
Pattern 4	3	575	375	225	1175
Pattern 5	3	375	225	125	725
Pattern 6	2	900	225		1125
Pattern 7	6	225			225
Pattern 8	1	375	225		600
Total Pattern	28				

Table A.3: Example of first-fit algorithm with improvement

With Improvement						
	# of usage	**item 1**	**item 2**	**item 3**	**item 4**	**Total Length (cm)**
Pattern 1	6	900	300			1200
Pattern 2	5	625	575			1200
Pattern 3	2	575	400	225		1200
Pattern 4	3	575	375	225		1175
Pattern 5	3	375	225	125	225	950
Pattern 6	2	900	225			1125
Pattern 7	1	225	225			450
Pattern 8	1	375	225	225		825
Total Pattern	23					

Algorithm 4 Pseudocode of First-Fit Decreasing Algorithm

```
Step 1: Initialize patterns
    Sort items according to their lengths in descending order
    Open patterns for each item whose length is larger than capacity/2 in amount
of their order
Step 2: Assign Orders to the patterns
    For remaining items
        For each open pattern
            If item length < unused capacity of the pattern
                If order of the item > amount of convenient patterns
                    Assign orders as much as possible
                    Update order amount of the item
                    Go to next open pattern
                Else if order of the item = < amount of convenient patterns
                    Assign all orders
            Else Go to next pattern
        If no suitable pattern exists to assign orders
            Open new patterns in amount of remaining orders of the item
    If no orders of the item are left to assign, Go to Next item
Step 3: Improve Pattern Utilization
    Sort Patterns according to their total length in ascending order
    Check whether orders of items can be located to other patterns
    If so, assign orders of the items as much as possible to the other patterns
    Check whether orders of items can be aggregated in their own patterns
    If so, aggregate orders of the items as much as possible in the patterns
```