

APPLICATION OF GILMORE-GOMORY ALGORITHM TO MULTI WIDTH
CUTTING STOCK PROBLEMS

TUĞÇE ÜSTÜNER

MAY 2015

APPLICATION OF GILMORE-GOMORY ALGORITHM TO MULTI WIDTH
CUTTING STOCK PROBLEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
IZMIR UNIVERSITY OF ECONOMICS

BY

TUĞÇE ÜSTÜNER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MAY 2015

Approval of the Graduate School of Natural and Applied Sciences



(Prof. Dr. Ismihan Bayramoğlu)

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Intelligent Production Systems** option of **Intelligent Engineering Systems**.



(Prof. Dr. M. Arslan Örnek)

Head of Department

We have read the thesis entitled **Application of Gilmore-Gomory Algorithm to Multi Width Cutting Stock Problems** prepared by **Tuğçe ÜSTÜNER** under supervision of **Prof. Dr. Arslan ÖRNEK**, and we hereby agree that it is fully adequate, in scope and quality, as a thesis for the degree of **Master of Science in Intelligent Production Systems** option of **Intelligent Engineering Systems**.



(Prof. Dr. M. Arslan ÖRNEK)

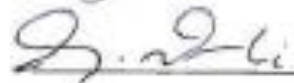
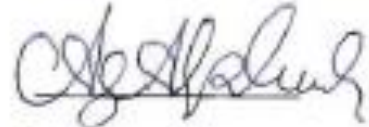
Supervisor

Examining Committee Members:
(Chairman, Supervisor and Members)

Assoc. Prof. Dr. Adil ALPKOÇAK
Computer Engineering Dept., DEÜ

Prof. Dr. M. Arslan ÖRNEK
Industrial Engineering Dept., IUE

Asst. Prof. Dr. Kamil Erkan KABAK
Industrial Engineering Dept., IUE



ABSTRACT**APPLICATION OF GILMORE-GOMORY ALGORITHM TO MULTI WIDTH
CUTTING STOCK PROBLEMS**

Üstüner, Tuğçe

M.Sc. in Intelligent Engineering Systems
Graduate School of Natural and Applied Sciences

Supervisor: Prof. Dr. Arslan ÖRNEK

May 2015, 103 pages

Cutting Stock Problem is one of the most important problem for the paper factories. Main objective of the factories, making best cutting plan combining demands which have different widths and quantities. After the cutting process, it is very important that choosing rolls which will go to cutting process considering using number of rolls. For this reason, combining of demands should be considered. The objective is choosing the right number of roll which will go to cutting process while combining demands and minimizing the trim loss which means that minimizing number of rolls. In the literature, there are a lot of researches and models are applied and improved. In this thesis, aim is minimizing using number of rolls also generating cutting pattern considering different roll widths. A mathematical modeling is developed and it is studied with using real data and using different roll widths from the paper factory. Also, improving and applying Gomory Algorithm the results in terms of performance measure are compared and discussed throughly.

Keywords: cutting stock problem; heuristic; roll; trim minimization.

ÖZ

ÇOK ENLİ KESME PROBLEMLERİNE GILMORE-GOMORY ALGORİTMASININ UYGULANMASI

Üstüner, Tuğçe

Akıllı Mühendislik Sistemleri Yüksek Lisans Programı
Fen Bilimleri Enstitüsü

Tez Danışmanı: Prof. Dr. Arslan ÖRNEK

Mayıs 2015, 103 sayfa

Kesim problemi kağıt fabrikalarının en önemli problemlerinden biridir. İşletmelerin temel amacı; farklı en ve adetlerde gelen siparişlerin kombinesi yapılarak en uygun şekilde kesim planlarını yapmaktır. Kesim işleminden sonra kalan fire göz önünde bulundurulduğunda kesim işlemine girecek olan bobin sayısı seçimi çok önemlidir. Bu sebepten dolayı siparişlerin kombinelerinin nasıl yapıldığı da dikkate alınmalıdır. Amaç oluşabilecek en az fireyle yani kullanılacak en az bobin sayısı ile doğru siparişleri kombin yaparak üretime girecek en doğru bobin sayısını seçmektir. Literatüre bakıldığında bu alanda birçok uygulama ve model geliştirilmiştir. Bu tezde kesilecek bobin sayısı minimizasyonu amaçlanarak farklı bobin enleri dikkate alınarak kesme eşleri oluşturulmuştur. Problemin matematiksel modeli oluşturulup kağıt fabrikasından alınan gerçek veri setleri ve farklı bobin enleri ile çalıştırılmıştır. Ayrıca Gomory Algoritması geliştirilip uygulanarak sonuçlar performans ölçütü üzerinden karşılaştırılmış ve ayrıntılı bir şekilde tartışılmıştır.

Anahtar Kelimeler: kesme problemi; sezgisel; bobin; fire minimizasyonu.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor Prof. Dr. Arslan Örnek for his guidance, contribution, academic and continuing support, and patience during my studies in the Izmir University of Economics for my master thesis.

I would like to extend my thanks to them who have helped me during the master program. Especially, my sisters İpek Süğüt and Hatice Doğramacı for their support and contribution in my MSc years in the Izmir University of Economics. Additionally, I would like to appreciate to them for their advice and comments while enrolling in Intelligent Production Systems Master Program.

A special thanks to my dear friends; Mehmet Serbes, Asena Kaya and my boss Metin Siloni.

Lastly, my dear family Serpil Üstüner, Ali Üstüner and Tuğkan Üstüner have provided all their support during these years. They are very important for me. I would like to thank and express my gratitude to them for their endless love, being with me and trusting me in all my life. I cannot achieve success without them.

CONTENTS

ABSTRACT	iii
ÖZ	v
ACKNOWLEDGEMENTS	vii
CONTENTS	viii
LIST OF TABLES.....	x
LIST OF FIGURES	xi
TERMS AND ABBREVIATIONS	xii
1 INTRODUCTION	1
1.1. SCOPE OF THE MASTER THESIS	2
2 LITERATURE REVIEW	4
3 CUTTING STOCK PROBLEM	13
4 RELATIONSHIP BETWEEN THE CSP AND GA.....	19
5 GILMORE GOMORY ALGORITHM.....	21
6 PROBLEM DEFINITION	24
7 DATA COLLECTION.....	28
8 MATHEMATICAL MODELLING.....	30
8.1. MATHEMATICAL FORMULATION.....	30
9 APPLICATION OF GILMORE GOMORY ALGORITHM.....	33
9.1. GILMORE-GOMORY ALGORITHM.....	34
9.2. IMPROVED GILMORE-GOMORY ALGORITHM FOR MULTIPLE STANDARD ROLLS.....	38
10 RESULTS.....	42
11 CONCLUSION.....	44
REFERENCES	45
APPENDICES.....	47
APPENDIX - A.....	47

APPENDIX - B.....51
APPENDIX - C.....73
APPENDIX - D.....83

LIST OF TABLES

Table 1 Literature Surveys	11
Table 2 Characteristics features while classfying problem	16
Table 3 Example table of trim loss calculation for roll 1020 mm and 1730 mm.....	26
Table 4 Roll widths	28
Table 5 Demands and their widths	28
Table 6 Diagonal matrix for 1020 mm.....	29
Table 7 Initial matrix for 1020 mm	35
Table 8 Iteration matrix.....	37
Table 9 Algorithm result.....	37
Table 10 Initial matrix for 1020 mm and 1730 mm	39
Table 11 Iteration matrix.....	41
Table 12 Algorithm result.....	41
Table 13 Solution table for mathematical model and Gilmore-Gomory Algorithm.....	43

LIST OF FIGURES

Figure 1 One Dimensional Cutting Stock Example	14
---	----

TERMS AND ABBREVIATIONS

CSP	: Cutting Stock Problem
CS	: Cutting Stock
1-D	: One Dimensional
2-D	: Two Dimensional
PSO	: Particle Swarm Optimization
LP	: Linear Programming
IP	: Integer Programming
DP	: Dynamic Programming
MIP	: Mixed Integer Programming
Non-LP	: Non Linear Programming
ILP	: Integer Linear Programming
SRI	: Stock Remove Insert
SA	: Simulated Annealing
EA	: Evolutionary Algorithm
GA	: Genetic Algorithm
PGA	: Pattern Generating Algorithm
KA	: Knapsack Algorithm
SHP	: Sequential Heuristic Procedure
CG	: Column Generation
GRASP	: Greedy Randomized Adaptive Search Procedure

CHAPTER 1

INTRODUCTION

The Cutting Stock Problem which is called in the literature 'CSP' is used in many different industries. Paper, glass, steel, wood and plastic popular examples are in these areas for CSP.

CSP is one of the classical problem example and first appliance in the field of operations research methods. The problem is finding the best solution way of cutting pattern items from using stock rolls such that trim loss and used number of rolls are minimized and total demand is satisfied. The aim of process is cutting large objects to convert them to smaller objects. The main purpose of CSP is minimizing trim loss which is number of rolls while cutting process in the thesis.

CSP is an integer linear programming problem, and solving this problem is not easy as known. In order to save profit annually in the factories, trim loss should be minimized deciding true stock roll which will go to cutting process. Because affects amount of the trim loss are very important for the industry profits.

Generally there are three different groups for solving methods. These different groups are; Algorithmic methods, Innovative methods and Metaheuristic methods.

Algorithmic methods contribute an optimal solution, despite of calculating complexity. But in big problems which have more and complex data, run time consuming occurs. That is why the complete algorithmic methods were infrequently used in the past.

In Innovative methods, generally answer is not optimum necessarily however give the response quickly. The innovative methods can be one of the ideal techniques, because their answers are close to the optimum responses. These methods usually are applied for special problems. Mentioned method is not applied as a general method because of depending private conditions excessively.

However various innovative solutions can be suggested in real world applications so as to solve the CSP but applying them is not helpful in similar problems due to the special characteristics of such problems .

In metaheuristic methods, the solution process is generally advised for lower levels, and contrary to traditional metaheuristic methods, they are not limited to local optimum conditions. The information about the CSP will be given in Cutting Stock Problem Chapter in detail.

1.1. Scope of the Master Thesis

In this study, we focus on minimizing the number of cutting rolls which means that trim loss, determining the cutting patterns which should be satisfied the demand.

In the factory, paper is bought from the out of Turkey and it is known that it is very expensive raw material in the world especially for the box and sheets factories. This paper comes to business such as rolls. According to demand these rolls should be cut. For this reason deciding cutting plan and generating cutting patterns are so important for the production cost of the factory. So, in order to decrease these cost, two mathematical models are improved. In the first model, all the cutting patterns are generated. In the second model, Gomory Model was rebuilt according to different roll widths and solved in order to find optimal solution. Because loss means that extra cost for the business. In order to decrease the extra cost and minimize the production cost, combination of the cutting patterns should be occurred and processed effectively.

Our scope is minimizing amount of rolls in cutting process. A mathematical model is developed and solved in Optimization Programming Language, CPLEX. Decreasing the cutting rolls optimally, Gomory Algorithm is improved and applied according to our problem using different roll widths and defined its steps in order to solve heuristically. The objective of the problem is minimizing the number of cutting rolls while the cutting process, satisfying demand, generating and assigning cutting pattern to the roll considering different roll width. The main inputs are demands, widths of rolls.

The remainder of this thesis is organized as follows. Chapter 2 introduces a review of the literature on related previous work about cutting stock, mathematical models and algorithms which were used to solve this kind of problem. Chapter 3 introduces what is the cutting stock problem, the main objectives, description of ideas, forms of CSP. In Chapter 4 CSP and GA relationship is discussed. Problem definition is explained in Chapter 6. Mathematical model and its solution are defined in Chapter 8.

In Chapter 5, Gomory Algorithm which is a metaheuristic method is developed for the CSP and lastly, conclusion is written in Chapter 11.

CHAPTER 2

LITERATURE REVIEW

There are many various exercises and works in literature on different aspects of Cutting Stock Problem. The relevant works were summarized in this chapter.

Cutting Stock Problem focuses on minimum wastage or scrap criterias with one or two dimensional stock according to customer demands. Cutting stock and packing problems appear in many industrial settings where larger pieces are cut into smaller pieces in order to produce demands that are wanted by other industries or customers, like paper, steel and fiber industries. A solution for the cutting problem consists in determining a group of patterns and their repetitiveness which means that how many times a cutting pattern will be cut in order to satisfy demand.

The cutting stock problem is called 'CSP' which affects industry profit has been studied seriously in production planning area but the results are not often used at real production sites. They are usually found using hand methods because real processing constraints have not been focused on. Making small changes in cutting pattern can create considerable benefits in different areas such as production cost which is the most important factor for factories. CSP contains different objectives and constraints, which directly depend on technological and organizational parameters of each company.

Cutting Stock Problems are separated categories in each. One of them is 'One-Dimensional Cutting Stock Problem' and the second one is 'Two-Dimensional Cutting Stock Problem'.

In 1939 studies have been started in this area by Russian economist 'Kantorovich'. A lot of formulation, mathematical models and solution procedures have been developed in this area. Thanks to Kantorovich, studies led to the continuous relaxation.

Gilmore and Gomory (1961), studied first and most important advance solution technique in CSP area. CSP is a which sort of problem is showed by them. According to Gilmore and Gomory CSP completes an order at minimum cost for specified lengths in order to be cut from given stock lengths. Pattern generation technique was improved using linear programming by them so as to solve the one dimensional trim loss minimization problem. When they converted it to integer programming problem, the large number of variables contain generally infeasible solutions. This same difficulty continued when only an approximate solution was being sought by linear programming. In order to overcome this difficulty The Column Generation Method was improved by Gilmore and Gomory.

In this method, setting of simple patterns which are form of the initial basis solution is the first step, then the solution is improved by removing a cutting pattern and generating a new pattern . The new cutting pattern is generated using the auxiliary problem which is easy to solve; knapsack problem normally is relate to the auxiliary problem, and there are different methods for solving this kind of problem. The new column is generated. Because this action is done in order to improve the solution.

After Gilmore and Gomory proposed the column generation method, many researchers used it for the Cutting Stock Problem.

In order to solve the roll wastage problem, this algorithm was used by Pierece(1964) in the paper industry. Because of occuring defaults in the problem, Hahn (1968) improved a dynamic programming algorithm.

Queiroz et al. (1971) presented other algorithms for the following three dimensional guillotine cutting problems which are unbounded knapsack, cutting stock and strip packing. They considered the case where the items have fixed orientation and the case where orthogonal rotations around all axes are allowed. For the unbounded 3D knapsack problem, they developed the repetition formula proposed by for the rectangular knapsack problem and presented a dynamic programming algorithm that uses reduced raster points.

Haessler and Sweeney (1991) focused on basic formulation topics and solution procedures for solving one and two dimensional cutting stock problems. They defined linear programming, sequential heuristic and hybrid solution procedures.

They also suggested an approach for solving large problems with limits on the number of times an ordered size may appear in a pattern for two-dimensional cutting stock problems with rectangular shapes.

Lin (1994) presented a study which considering the minimizing trim loss in a paper cutting process. In the study, operation begins with some parent rolls of specific widths are to be cut to meet orders. These rolls have specific widths. The objective of the study was to find out the way to achieve the required cutting operation and occurring the smallest amount of trim loss by exploiting linear programming.

This problem was solved by observing the 'pricing out' operation for the paper cutting problem which was equivalent to a knapsack problem.

In recent years, researchers have improved applying evolutionary approaches to these problems, including Genetic Algorithms and Evolutionary Programming.

Liang et. al developed EP algorithm for CSP with and without contiguity. The propose is realized using two new mutation operators. Experimental studies have been achieve to examine the efectiveness of the EP algorithm. They showed that EP can support a simple yet more efective alternative to GA's in solving cutting stock problems with and without contiguity. The solutions found by EP are significantly better than to those found by GAs.

Carvalho (1999) reviewed several linear programming formulations for one-dimensional cutting stock and bin packing problems. He analysed some relations between the corresponding LP relaxations, and their relative strengths, and refered how to derive branching schemes that can be used in the exact solution of these problems, using branch-and-price technique.

Özgüven and Çalışkan (2002) who are Turkish academicians showed a model which was written by Konrad in their study. In the problem, there were a lot of customer demands in which materials had fixed width which will cut according to different length. The aim was minimizing trim loss. Their

model is not enough in order to solve so, they proposed using a mixed integer model which answering the different requirements.

Cutting Stock Problems are complexity problems. Because it has large number of the cutting patterns that may be encountered. The large number of cutting patterns returns the solution generally infeasible, when the cutting stock problem was expressed as an IP problem. When using the linear programming formulation of the cutting stock problem is available of integer variables, then the effect of the number of cutting patterns will be decreased. An auxiliary problem proceeds from the formulation where the columns of the linear programming constraint matrix need to be determined.

Cerqueira and Yanasse (2006) reviewed some linear programming models for the CSP.

Also, Novianingsih et al. considered 2-D CSP where single rectangular stocks have to be cut into some smaller pieces so that the number of stocks needed to meet demands is minimum. Also, they focused on studying to the problem where the stocks have to be cut with guillotine cutting type. Problem is formulated as an IP and the relaxation problem was solved by column generation technique. New pattern generation was formulated based on method of stripe. In obtaining the integer solution, they rounded down the optimal solution of the relaxation problem and then they derived an extra MIP for satisfying demands.

Demircan and Soyuer (2007) improved a method using real data which is taken from factory in order to form stainless steel process. Factory supplies materials which have different length and types.

In order to solve the problem, they suggested two steps solution. In the first step, different cutting shapes were obtained for every piece and alternative raw materials length using heuristic method. In the second step, these cutting shapes which are taken heuristic method were applied to integer-linear programming. Using this method, trim loss was minimized, which raw material will be used and which length will be cut and how many will be cut were determined. Also, all of the customer demands were satisfied.

Glass and Oostrum described a new hierarchical 2D-guillotine Cutting Stock Problem. This method is contrast to general methods for CSP, because aim was not wastage. The packing stages of cake manufacturing

was the content. The company's first objective was to minimize total processing time at the subsequent in packing stage. This objective reduced to one of minimizing the number of parts produced. They applied a closed form optimization approach to these problems for certain cases.

In 2008 a new mathematical model was presented and applied by Golfeto et al. which was GRASP metaheuristic to solve ordered cutting stock problem. This heuristic appropriate to minimize the raw material used by industries that deal with reduced raw material stocks in which just in time method used for production process. In such cases, classic models for solving the cutting stock problem were useless. Results obtained from computational experiments for a set of random instances demonstrate that the proposed method can be applied to large industries that process cuts on their production lines and do not stock their products.

Cerqueira and Yanasse (2008) introduced us a heuristic method that produces a solution for the one-dimensional cutting stock problem with a reduced number of different patterns in the solution. Firstly, this method proposed separating the items in two disjointed groups, according to their demands. Patterns is generated with items of these groups and those with limited trim loss are accepted. Then, problem was solved with items whose demands were not satisfied and, when the solution obtained, they applied a pattern reducing procedure of the literature.

Arbib et al. (2010) addressed a one-dimensional CSP. They focused on not only minimizing trim loss, but also they needed that the set of cutting patterns construct the solution can be sequenced so that the number of stacks of parts obtained.

In order to solve problem, a new integer linear programming formulation was improved and used. Constraints of the formulation raise quadratically with the number of specific part types.

Sugi et al. (2010) worked with the 2-dimensional rectangular cutting stock problem in which the shape of a cut piece is rectangular, they assumed that a roll-shaped stock often used in actual processing and proposing a solution taking processing called 3-stage guillotine cutting into account.

Cathrine et al. (2011), prepared a study for the carpentry sector. The carpentry sector like any other industry was faced with a cutting stock

problem to minimize incurred wastage. The aim of this problem was to establish a mathematical model which will solve the CSP using column generation approach. The interview method was used to collect data relating to the cutting stock problem. The column generation approach of iterative computational routines was used because it developed successively better solutions until an optimal solution is obtained. The results revealed that the method was an appropriate method in solving business problems, that was, how many boards should be cut to meet demand with minimum incurred waste.

Abuhassan and Nasereddin (2011) touched on the application. They aimed that decreasing the losses for the problem of cutting in one dimension.

Nozarian et al. (2013) focused on trim-loss amount. They applied simulated annealing algorithm in order to reduce trim-loss considering production cost amount. New theory is improved and applied in order to solve the trim loss problem. Furthermore, a solution based on Imperialist Competitive Algorithm were presented that reduced the wastage as well as concentrating them on the minimum number of stocks.

An algorithmic solution approach was presented by Suliman (2014) to overcome the difficulty in solving non-linear integer formulation of the problem. The algorithm was based on the traditional approach where the lot sizing was determined for each period, and then the best cutting patterns were generated.

Sürsal explained one dimensional and three dimensional cutting stock problems. Decision model was developed in order to solving these problems.

On the other hand, heuristic application continued in CSP area. For instance, the other work was written by Levine and Ducatelle. They presented a pure ACO approach, as well as an ACO approach augmented with a simple but very effective local search algorithm. It was shown that the pure ACO approach can outperform some existing solution methods, whereas the hybrid approach can compete with the best known solution methods. The local search algorithm was also run with random restarts and shown to perform significantly worse than when combined with ACO.

Macedo et al. presented a detailed search for software packages with using two-dimensional cutting stock method.

Suliman wrote another study which was about simple pattern generating method. He was developed it for solving the auxiliary problem. It was based on an ad hoc solution method described in literature for the knapsack problem. A search tree was used to develop the pattern generation method.

Another heuristic approach was written by Shen and Yu. A heuristic strategy that was based on the results of analysis of the optimal cutting pattern of particles with successful search processes was described, which process a global optimization problem of the cutting-stock as a sequential optimization problem by multiple stages. During every sequential stage, the best cutting pattern for the current situation was researched and processed. This strategy was repeated until all the required stocks have been generated.

Another work is Branch and Price technique was written by Pal. Branch and price was established so as to solve for large data using integer programming problems. This method combined the standard branch-and-bound framework of solving integer programming problems with Column Generation. In each node of the branch-and-bound tree, the bound was calculated by solving the LP relaxation. The LP relaxation was solved using Column Generation. They discussed their project on improving the performance of branch and price based algorithms for solving the industrial one-dimensional cutting stock problem.

The other study which I talked in my literature review of this area lastly was written by Macedo and Alves. They described a model for the two-dimensional Cutting Stock Problem using two stages in order to solve using the guillotine constraint. It was a linear programming arc-flow model, formulated as a minimum flow problem, which was an extension of a model proposed by Carvalho for the one dimensional case. They researched how this model behaved using with commercial software, explicitly considering all its variables and constraints. They also implied a new family of cutting planes, and considered some extensions of the original problem.

In the thesis, we developed new mathematical model and solved for the Cutting Stock Problem. The difference from the other studies we considered different roll widths. Also the problem was integrated to a heuristic method which named is 'Gomory Algorithm'. These different roll

widths were not considered before using Gomory Algorithm. The details of the problem will be explained in the other chapters.

Table 1 Literature Surveys

			APPLICATION PART					D			AIM		
			LP	CG	LONG ROLLS PAPER	ALUMINUM ROLLING	GUILLOTINE	HEURISTIC	1	2		3	
1	ROBERT W. HAESSLER AND PAUL E. SWEENEY	CSP AND SOLUTION PROCEDURES(1991)	*						* SHP	*	*		
2	CEMAL ÖZGÜVEN AND FİLİZ ÇALIŞKAN	SİPARİŞE GÖRE ÜRETİM YAPAN İŞLETMELERDE KISITLI MALZEME ÇEŞİDİ DURUMUNDA FİREYİ MINİMİZE EDEN PARÇA KESİM PLANININ BELİRLENMESİ(2012)	MIP										MINIMIZING WASTE
3	S.M. A. SULIMAN	AN ALGORITHM FOR SOLVING LOT SIZING AND CSP WITHIN ALUMINUM FABRICATION INDUSTRY(2012)	Non-LP			*							MINIMIZING TRIM-LOSS
4	SHIRIN NOZARIAN,MAJID VAF AEI JAHAN,MEHRDAD JALALI	AN IMPERIALIST COMPETITIVE ALGORITHM FOR 1-D CSP(2013)							* SA	*			MINIMIZING TRIM-LOSS
5	K. NOVIANINGSIH,R. HADIANTI, S. UTTUNGGADEWA	COLUMN GENERATION TECHNIQUE FOR SOLVING 2-D CSP:METHOD OF STRIPE APPROACH (2007)	IP	*					*		*		
6	CLAUDIO ARBIB	CS WITH BOUNDED OPEN STACKS:A NEW ILP MODEL(2010)	IP							*			MINIMIZING TRIM-LOSS
7	MASAO SUGI,YUSUKE SHIOMI,TSUYOSHI OKUBO,KAZUYOSHI INOUE,JUN OTA	A SOLUTION FOR 2D RECTANGULAR CSP WITH 3-STAGE GUILLOTINE-CUTTING CONSTRAINT(2010)							*		*		
8	CELIA A. GLASS,JEROEN M. VAN OOSTRUM	BUN SPLITTING:A PRACTICAL CSP(2008)					*		*		*		FINDING OPTIMAL CUTTING STRATEGY
9	KAZUNGA CATHERINE,MUTAMBARA H.N. LILLIAN, MAPURISA JABULANI	A CARPENTRY CSP(2011)	IP	*					* COLUMN GENERATION, BRANCH AND BOUND ALGORITHM				MINIMIZING WASTE
10	GÖKAY SÜRSAL	THE CSP AND A SOLUTION ALGORITHM BY LP	*						*KA	*	*	*	MINIMIZING WASTE
11	RODRIGO RABELLO GOLFETO,ANTONIO CARLOS MORETTI,LUIZ LEDUINO DE SALLES NETO	A GRASP METAHEURISTIC FOR THE ORDERED CSP(2008)							* GRASP ALGORITHM	*			MINIMIZING RAW MATERIAL AND MINIMIZATION TRIM LOSS
12	KO-HSIN LIANG, XIN YAO,CHARLES NEWTON,DAVID HOFFMAN	A NEW EVOLUTIONARY APPROACH TO CSP WITH AND WITHOUT CONTIGUITY(1998-1999)							*EA	*			MINIMIZING TRIM LOSS AND MINIMIZATION NUMBER OF STOCKS WITH WASTAGE
13	THIAGO A. DE QUEIROZ, FLAVIO K. MIYAZAWA, YOSHIKO WAKABAYASHI, EDUARDO C. XAVIER	ALGORITHMS FOR 3D GUILLOTINE CUTTING PROBLEMS:UNBOUNDED KNAPSACK ,CS AND STRIP PACKING	DP	*					*			*	

14	P.C. GILMORE, R.E. GOMORY	A LP APPROACH TO THE CSP(1961)	IP															MINIMIZING COST
15	IZZEDDIN A.O. ABUHASSAN, HEBAH H.O. NASEREDDIN	CSP:SOLUTION BEHAVIORS(2011)		*						* SEQUENTIAL HEURISTIC APPROACH	*							REDUCING LOSSES TO THE PROBLEM OF CUTTING
16	GONÇALO RENILDO LIMA CERQUEIRA, HORACIO HIDEKI YANASSE	A PATTERN REDUCTION IN A 1-D CSP BY GROUPING ITEMS ACCORDING TO THEIR DEMANDS (2008-2009)	*							*	*							MINIMIZING AMOUNT OF TIME, IMPROVING PERFORMANCE
17	JOHN LEVINE AND FREDERICK DUCATELLE	ANT COLONY OPTIMISATION AND LOCAL SEARCH FOR BIN PACKING AND CSP	*							*								MINIMIZING NUMBER OF TOTAL STOCKS
18	J.M. VALERIO DE CARVALHO	LP MODELS FOR BIN PACKING AND CSP(2000-2001)	*							* BRANCH AND PRICE ALGORITHM	*							
19	FATMA DEMIRCAN, HALUK SOYUER	1-D CSP	*							*	*							
20	RITA MACEDO, ELSA SILVA, CLAUDIO ALVES, FILIPE PEREIRA E ALVELOS, J.M. VALERIO DE CARVALHO, CLAUDIO ARBIB, FABRIZIO MARINELLI, FERDINANDO PEZELLA, LUIGI DE GIOVANNI, LUCA GAMBELLA	2D CUTTING STOCK OPTIMIZATION SOFTWARE SURVEY											*					
21	SAAD M.A. SULIMAN	PATTERN GENERATING PROCEDURE FOR THE CSP(2001)					*			* PGA								MINIMIZING TOTAL WASTE
22	XIANJUN SHEN, YUANXIANG LI, JINCAI YANG, LI YU	A HEURISTIC PARTICLE SWARM OPTIMIZATION FOR CSP BASED ON CUTTING PATTERN								PSO	*							MINIMIZING TOTAL TRIM LOSS
23	SOUMITRA PAL	IMPROVING BRANCH AND PRICE ALGORITHMS FOR SOLVING 1-D CSP								*	*							
24	RITA MACEDO, CLAUDIO ALVES, J.M. VALERIO DE CARVALHO	ARC-FLOW MODEL FOR THE 2-D CSP	*										*					
25	GONÇALO RENILDO LIMA CERQUEIRA, HORACIO HIDEKI YANASSE	LP MODELS FOR THE 1-D CSP	*									*						
26	VINCENT CONITZER	CONSTRAINT AND COLUMN GENERATION AND THE CS	*	*	*					*								USING AMOUNT OF PAPER
27	REINALDO MORABITO AND VOLDIR GARCIA	THE CSP IN A HARDBOARD INDUSTRY(1997)	DP, IP							* LEXICOGRAPHICAL METHOD								MINIMIZING THE WASTE MATERIAL, DETERMINING BEST PATTERNS

CHAPTER 3

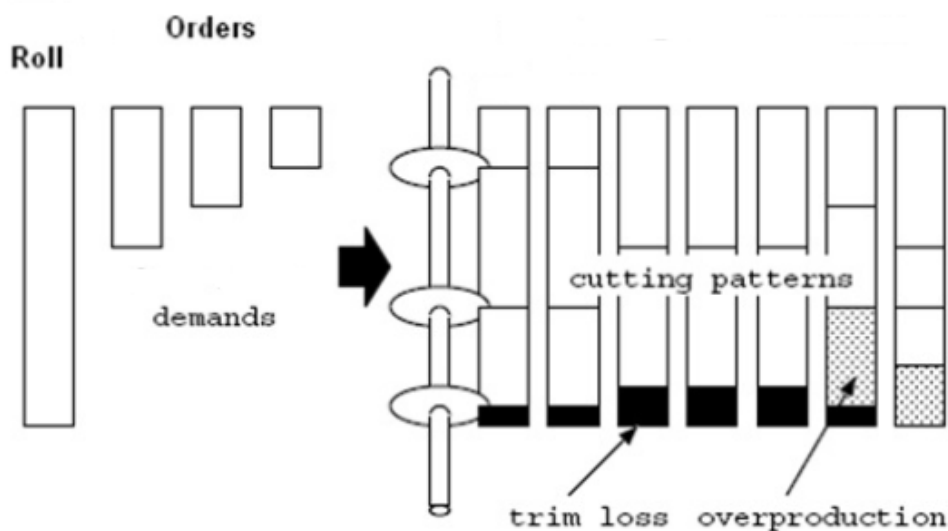
CUTTING STOCK PROBLEM

Cutting stock problems were studied before in operations research area. In these studies, there are different constraints, decision variables, indices, sets, may be objectives. In many real world applications of business and industry generally, it is known that optimization problems have given set of small objects which are called items or pieces, into a given set of larger ones which named are stock sheets. Also in these kind of problems generally have same aim which is the reducing waste or minimizing production cost using less raw material with true generating patterns. These problems, with all their extensions and variants, are well known to be NP-hard. This means that all algorithms currently known for finding optimal solutions require a number of computational steps that may grow exponentially with the problem size rather than according to a polynomial function.

After the some years, other CSP approaches have developed in order to solve the problems. These approaches are exact and heuristic methods. Heuristic methods have greater flexibility considering specific constraints in problem and offer a good trade-off between the quality of a solution and its computational effort. Generally, heuristic techniques need to be used for large CSPs. Some of the heuristic algorithms which were applied to CSPs in the recently years with success. These heuristics examples are genetic algorithm, simulated annealing and column generation. In these methods, requiring to provide good, but these methods do not provide the optimal solutions.

Linear or dynamic programming and branch-and-bound techniques are based from the exact algorithms. Because of being complexity and extensive nature of these problems, many different optimization formulations and solution approaches improved in the literature, according to their needs such as dimension, application field, constraints and requirements. Therefore, many researchers surveys were categorized their studies on this subject in the literature chapter of my thesis.

The Cutting Stock Problem is an important class for the combinatorial optimization areas. The general goals of Cutting Stock Problems are to minimize the trim loss or the production cost. Most CSP solution methods are established for specific objective functions. Generally, each of them has second objective. The second objective for the first group Cutting Stock Problem is to minimize the number of used stocks. This type of CSP with two objectives has been solved or worked by using some heuristic techniques. The second objective for the second class of Cutting Stock Problem is to minimize the number of partially finished items. This is known as the cutting pattern sequencing problem which named CSP with contiguity in the literature.



**Figure 1 One Dimensional Cutting Stock Example
(Abuhassan and Nasereddin, 2011)**

CSP has many different forms, these are One Dimensional (1-D) CSP such as sheets of wood, second one is Two Dimensional (2-D) CSP like cutting cloth or paper to cut rectangular and these are more complex according to One dimensional CSP.

In one dimensional CSP, a part or some parts of the raw material might not be used again, for instance when a piece is cut. It happens due to applying various patterns freely. Before, cutting problem has attracted the attention of many researchers all over the world. Choosing a cutting pattern and its sequences is put forward by cutting problems. Sometimes cutting problems are too complex, and it is not easy to find an optimized response for them. In such case, even the smallest improvement in cutting pattern, may lead to a considerable minimizing in raw material, which are used over. Most standard problems that are related to one dimensional cutting problem are known as NP-complete problems. However it is possible in many cases to model them by mathematical programming and find a solution via accurate or approximate methods.

On the basis of H. Dyckoff's typology, the one dimensional problem can be explained as 1/V/D/R whenever sufficient materials are available. "1" means for one dimensionality of the problem. "V" means that all required items should be produced by a selection of big consuming pieces; in other words, although some parts of stocks are only used, all orders will be produced. "D" means that there are several big consuming pieces in different sizes and "R" represents the number of items.

Practically, the cost of using a special cutting model and changing the cutting patterns are important ordinary factors as well as cutting wastage. The secondary aspects can be considered by an appropriate formulation and making the smallest improvement in cutting pattern. It leads to a big thriftiness in stocks, which are consumed quickly and repeatedly in huge mass. Ineffectiveness and excess of manual methods, which are applied by cutting contractors, reminds the necessity of cutting automation. Furthermore, the potential of suggested methods can be separated and compared easily, due to the impossibility of comparing the different solutions together. There are several algorithms and methods in order to calculate the one dimensional

cutting wastages, considering various factors such as constraints, demands, materials.

The large variety of applications reported in the literature by Dyckhoff (1990) to develop a classification scheme for cutting stock and packing problems.

If we want to give an example for CSP, examples were studied in generally a factory which uses long rolls of paper of a fixed width 'W'. Paper can be cut as it comes out. For instance, producing two rolls of width 'W/2'. The rolls can be cut only vertical direction. Certain orders for paper have been needed to fill. Each order 'i' has a width 'w_i' and a length 'l_i'. If they have same width, multiple rolls of paper can be stitched together.

Table 2 Classifying problems using four characteristics table as follows

CHARACTERISTICS FEATURES WHILE CLASSIFYING PROBLEM				
CHARACTERISTICS FEATURES	DIMENSIONALITY	KIND OF ASSIGNMENT	ASSORTMENT OF LARGE OBJECTS	ASSORTMENT OF SMALL ITEMS
1	Number of dimensions	All large objects and a selection of small items	One large object	Few items of different dimensions
2		A selection of large objects and all small items	Many identical large objects	Many items of many different dimensions
3			Different large objects	Many items of relatively few dimensions
4				Many identical items

For instance, if there is a single order of length 'l' and width 'W/2', one roll of length 'l/2' and width W can be produced, cut it into two rolls of length 'l/2' and width 'W/2', and can be stitched them together in order to obtain one roll of length 'l' and width 'W/2'.

In the other direction cannot be stitched. For instance, two rolls of width W/4 into a roll of width W/2 cannot be combined. if it did, then the paper would look ugly everywhere. The goal is satisfying all of the orders while

using the minimum amount of paper. The total paper which has width 'W' that can be produced to be as short as possible. Since they can be only cut vertically, as well the paper can be cut as it is coming out of the machine.

At any point in time, the paper can be cut into a certain combination of widths. It is called such a combination a pattern. For instance, if 'W = 11', one pattern is to cut the paper into widths 5,5, and 1. May be there is not orders of width 1 so that the produced roll of width 1 is simply waste. In this case, it is simply say that the pattern is {5, 5} and it is implicit that the remaining 1 is wasted. So, a number will never occur in a pattern unless having an order of that width. Suppose that $W = 11$, and having three orders:

- $w_1 = 5, l_1 = 20$;

- $w_2 = 4, l_2 = 10$;

- $w_3 = 2, l_3 = 9$;

optimal solution here is using the pattern {5, 5} for a length of '5', and the pattern {2, 4, 5} for a length of '10'.

A roll of width '5' of length $2 * 5 + 10 = 20$ ', a roll of width 4 of length 10, and a roll of width 2 of length 10. It satisfies all orders. The objective of '15'.

It means that the total length of width 'W' paper that can be produced. More generally, all different patterns can be represented, indexed by 'j'.

In general, there are many such patterns; some of them are clearly dominated by other patterns such as {4, 5} is dominated by {2, 4, 5}.

Although having dominated patterns, there are undominated patterns like '{5, 5}, {5, 4, 2}, {5, 2, 2, 2}, {4, 4, 2}, {4, 2, 2, 2}, {2, 2, 2, 2, 2}'.

So, writing all dominated and undominated list are unnecessary and time losing. Because of that situation the linear program was modeled generating all the patterns 'j'. In these kind of problems, generally 'x_j' is called the amount of pattern 'j' that can produce, and 'a_{ij}' is named the number of times that the width of order 'i' occurs in pattern 'j'. For instance, if pattern '1' is '{5, 5}', then 'a₁₁' = '2'.

On the other hand, in two-dimensional cutting stock problem where stock sheets have to be cut into a set of smaller pieces so that the demand is satisfied. Number of cutting patterns are needed to generate and determine so that the number of stock sheets should be used minimum. The problem is

a well known problem appears in many industries, such as at glass industries, aircraft industries, ship builder, steel industries, and leather industries. The problem also appears in land development, facilities layout, and electrical circuit layout.

In practically, there are two cutting types for Cutting Stock Problem. These are guillotine cutting type and non-guillotine cutting type. These cutting types are different in each. The guillotine cutting type is a cutting type where any cut must run from one edge of a stock sheet to the opposite edge in a straight line. Simultaneously, in non-guillotine cutting type, a cut does not have to run from end to end of the stock sheet.

Also the 2D-CSP can be further classified into several categories, depending on the problem's specific constraints. It can be regular, if the shapes of the items to be cut can be described by few parameters, or irregular, otherwise is irregular category. Cutting irregular shapes is also known as nesting. Regular category cuts can be two different types which are rectangular or non-rectangular, according to whether the items are rectangles or have a different shape, respectively. Rectangular cutting is called oriente. If a sheet can only be cut from side to side, then guillotine-type cutting patterns; observe that problems allowing non-guillotine patterns are generally much harder to solve. A staged pattern is a guillotine pattern cut into pieces in a limited number of phases.

The direction of the first stage cuts may be either horizontal or vertical and the cuts of the same stage are in the same direction. The cut directions of any two adjacent stages must be perpendicular to each other. If the maximum number of stages is not allowed to exceed n , the problem is called n -staged. The relationship between the CSP and one of the algorithm method which is named Gilmore-Gomory's Algorithm will be explained in Chapter 4.

CHAPTER 4

RELATIONSHIP BETWEEN THE CSP AND GILMORE AND GOMORY'S ALGORITHM

Cutting stock problems occurs when raw materials are such as paper, cardboard and textiles in manufacturing company by rolls of large widths. While production planning, in order to satisfy demand and minimize wastage these rolls have to be cut into subrolls of smaller widths. It is not always possible to cut the rolls without leftovers. These leftovers parts are called trim 'loss', 'wastage' or 'scrap'.

In this section we will discuss what is the Gilmore-Gomory's Algorithm, features of the algorithm and relationship between the Gomory Algorithm and Cutting Stock Problem.

From past to now, it has been known that minimizing scrap is one of the most important problem in many industries. Because raw material is very important cost for the factories. This problem has been looked into from a different point of view in the past. Many problems were researched and tried to solve and find optimal in industrial engineering area from past to now.

Different optimization methods have been occurred and this optimization methods were searched also applied largely to various problems. Linear programming and the other specific solution methods can be used effectively in small data problems. When the problem size are large and complex, heuristic methods studies had been started. As result of these studies, it has seen that exponential growth of the search space and time loosing was occurred in order to find optimal solution.

The solution procedure of Gilmore-Gomory Algorithm published in 1963. Gilmore and Gomory studied for the CSP considering minimizing the wastage cost. In the Linear Programming Approach of method, the LP relaxation of the problem is considered and solved then a rounding procedure is used to take an integer solution. But some difficulties were occurred. One of them is, occurring large number of cutting patterns in the LP relaxation approach. Because generating them can be hardly.

The Gilmore-Gomory cutting plane algorithm is occurred in order to find the solution for the continuous relaxation of problem and the other aim was ensuring from its optimal solution using one or more inequalities. The solution disrupted these inequalities itself. These inequalities are added to the problem. Problem is still reoptimize. Then, the method is applied again to the new solution, process applied as long as the optimal solution becomes integer.

CHAPTER 5

GILMORE-GOMORY'S ALGORITHM

The cutting standard-sized pieces of stock material problems are named cutting stock problem. These pieces can be occurred by paper rolls or sheet metal, leather. They are divided or cut pieces of specified sizes while minimizing material. After cutting process, leftover is named trim loss or wastage. This trim loss can be named number of used rolls wasted in the cutting process. In these kind of problems, what trim loss is should be determined and explained. It is one of the critical point of these kind of problems. Because trim loss and number of used materials cannot be same at every problems.

In mathematical area, an optimization problem is occurred by applications. The cutting stock problem is an NP-complete problem. It can be return to the knapsack problem because of computational complexity. The problem can be formulated as an integer linear programming problem. This formulation applies not only one-dimensional problems but also many variations can be done, it is possible. The objective of cutting stock problem can be minimizing trim loss or maximizing the total value of the produced items using each order with different value.

Generally, if the number of demands increase, the amount of possible cutting patterns increase exponentially as a function of m , in cutting stock problems. It cannot be practical to compute the all possible cutting patterns. In order to preventing wasting time while solving, an alternative approach is improved which is Column Generation method. The Column Generation

method sometimes can be much more effective than the original advent, especially problem size increases.

In 1960s, the column generation approach was introduced and applied to the cutting stock problem by Gilmore and Gomory. Gilmore and Gomory proved and showed that this approach is guaranteed to converge to the optimal solution, without using and generating all possible cutting patterns. This method solves the cutting-stock problem by starting using few patterns which are determined before. If it is needed, it generates additional cutting patterns and use them in order to find optimal solution .

The other special part is one-dimensional cutting stock problem for general cutting stock problem. Solving an auxiliary optimization problem such as one dimensional cutting stock introduced us the new patterns which are named the knapsack problem.

In the Knapsack Problem, dual variables are found and used from the linear program. This problem is one of the strong method to solve these kind of cutting stock problems. The other known methods are branch and bound algorithm and dynamic programming like knapsack problem.

Some fractions causes limitation for the Gilmore and Gomory method. Because handling integrality is a limitation for the problem. Sometimes rounding to the nearest integer can not be useful for the cutting stock problem, because sub-optimal solution can be occurred. It means that under or over-production for some orders. But modern algorithms overcame this limitation for including very large instances of the problem.

Sometimes, occurring same trim loss can be possible in the cutting stock problem . So, possibility of corruption can be occurred. Also geneating new patterns increase the effect of this degeneracy without affecting the trim loss. Gilmore Gomory Algorithms can be coded like CPLEX or another programming language. The solution procedures of Gomory Algorithm are;

Step 1. Firstly, find the Simplex Tableau.

Step 2. Strong Gomory Cutting Planes associated with each row that has a fractional right hand side are found.

Step 3. Add these cutting patterns to the Simplex tableau including primal feasibility.

Step 4. In order to find a solution for the new LP, use the Dual Simplex Algorithm.

Step 5. If the solution is optimal, stop, otherwise return to Step 2.

CHAPTER 6

PROBLEM DEFINITION

The Cutting Stock Problem is the problem of filling an order at minimum cost for specified numbers of lengths of material to be cut from given stock lengths of given cost (Gilmore et al.1961).

The CSP is an integer programming problem. However, since the integer programming problems are known to be non-deterministic polynomial-time hard, the Cutting Stock Problem is formulated as linear programming problem by relaxing the integer requirements. After the linear programming optimal has been found, a rounding-up-procedures used to get optimum in the integer programming. It arises from many applications in industry including paper, glass, shoe-leather cutting, furniture, machine-building.

The most important and first aim of every business is to optimize cost which means that to maximize profit or minimize the cost of operation while satisfying demands.

In my thesis, my problem occurs in a carton and corrugated factory in Torbali/Izmir. This study was made using the real data from a factory which works about the corrugated area. Raw material is paper which is bought from abroad such as England, France, Germany, Israel etc. and domestic. In the depot, there are rolls which have different widths.

The thesis focuses minimizing the number of cutting rolls using the pieces which were cut with different amount from the fixed width materials for the factory which produces many products according to demand.

In the factories, managers should provide the stock enough in order to utilizing the advantages of stock and escaping the disadvantages of stock. When the demand comes, the materials which are required are controlled in the depot.

Generally, in the stock there are rolls which have different widths and lengths. These rolls should be cut in order to satisfy demand. In this situation, the most important cost is the trim loss cost which occurs when the cutting process is made for one more than product. Trim loss means that using number of rolls in this thesis.

Minimizing trim loss and minimizing number of used rolls are not same. In order to show this difference trim loss tables were prepared. In these tables, there are demand width, cutting patterns, width of rolls, number of rolls and trim loss calculation. Demand width and rolls width were known. Cutting patterns were occurred by hand. These patterns include all possible combinations. Number of rolls were calculated by using mathematical model.

In the trim loss calculation, there are three parts. These are used area per pattern, total area per pattern and loss per pattern. Total area was calculated by number of rolls multiplying with roll width. Used area per pattern was calculated by demand width multiplying with number of rolls and the number of subrolls corresponding desired width in that pattern and lastly loss per pattern was calculated by subtracting used area per pattern from total area per pattern.

aij	19	20	21	22	23	24	25	26	27	28	29
500	0	0	0	2	3	0	0	0	1	0	0
450	1	0	0	0	0	0	0	0	0	0	0
645	0	0	0	0	0	0	0	0	0	0	0
430	0	0	4	1	0	2	0	2	1	0	0
370	0	0	0	0	0	0	0	0	0	0	0
495	1	2	0	0	0	0	0	0	0	0	0
850	0	0	0	0	0	1	2	0	0	1	0
750	0	0	0	0	0	0	0	1	1	1	2
725	0	0	0	0	0	0	0	0	0	0	0
720	0	0	0	0	0	0	0	0	0	0	0
WIDTH OF ROLL	1020	1020	1730	1730	1730	1730	1730	1730	1730	1730	1730

# OF ROLLS	0	0	8	0	0	0	27	0	0	0	32
-------------------	---	---	---	---	---	---	----	---	---	---	----

TRIM LOSS CALCULATION											
USED AREA PER PATTERN	0	0	13760	0	0	0	45900	0	0	0	48000
TOTAL AREA PER PATTERN	0	0	13840	0	0	0	46710	0	0	0	55360
LOSS PER PATTERN	0	0	80	0	0	0	810	0	0	0	7360

Table 3. Example table of trim loss calculation for roll 1020 mm and 1730 mm

First Model Solution	Second Model Solution
TRIM LOSS	TRIM LOSS
25025	22575
# of used rolls	# of used rolls
185	200

After the cutting process, if the last piece will be used for the other production of product is used, if it is not it will go to scrap. In the study, the aim is making the cutting plan optimally and calculating number of rolls which will satisfy demand.

In terms of the losing material which is the standard width, the need is known to be substantially increased track parts in case of interruption of the loss occurring due to excess wastage. From this reason, it is very important that giving the information about the material from the stock. There is no problem if the all kind of rolls are in the depot.

But the factory which produces the products according the demand buy the rolls. These rolls are so expensive. So, the scrap cost should be less in order to decrease the production cost. The requiring rolls which has different widths are in the factory that kind of products are too much.

It is known that the factory produces the products according to customer's demand. In the production, paper is combined with starch, caustic and borax. This combination is processed in corrugated machine which

length of the machine is 2 meters. Papers have different widths. These widths are; 1020 mm, 1120 mm, 1320 mm, 1430 mm, 1530 mm, 1630 mm, 1730 mm, 1830 mm, 1930 mm and maximum 2000 mm.

Production is processed according to customer request so, there are a lot of different demand with different widths. While these demand were planning, they classified according to their amount and width. This means that, every demand cannot be combined with in each. While combining, their combination of width should be go to the cutting roll and the trim loss should be minimum. If the scrap is minimum, it is known that combination of demand is near the optimum. In order to understand the combination is optimum, mathematical model and algorithm were improved and their solution were compared.

A mathematical model which gives the suitable cutting plan is improved satisfying these demands. Also the object of this model, seeing the using number of rolls for every different widths and choosing the most suitable cutting pattern which minimizes the number of used rolls. Created a mathematical model as well as encountered in details in order to see results from ignoring and larger data for heuristic algorithm was developed to solve.

CHAPTER 7

DATA COLLECTION

The data was collected from cartoon and corrugated factory which is in Torbalı/Izmir. The data collection include matrices for the different roll widths and demands. There are eleven different roll widths and ten different demand widths and amount.

Table 4. Roll Widths

NO	WIDTHS
1	1020
2	1120
3	1220
4	1320
5	1430
6	1530
7	1630
8	1730
9	1830
10	1930
11	2000

Table 5. Demands and their widths

NO	DEMANDS	WIDTHS
1	10	500
2	20	450
3	50	645
4	60	430
5	40	370
6	45	495
7	55	850
8	65	750
9	80	725
10	45	720

There are initial matrices for every different roll. These matrices are 10x10 and shows that how many rolls require in order to satisfy for every demand. X_1 and $X_2 \dots X_{10}$ show that number of subrolls corresponding desired width in that pattern.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
500	2	0	0	0	0	0	0	0	0	0
450	0	2	0	0	0	0	0	0	0	0
645	0	0	1	0	0	0	0	0	0	0
430	0	0	0	2	0	0	0	0	0	0
370	0	0	0	0	2	0	0	0	0	0
495	0	0	0	0	0	2	0	0	0	0
850	0	0	0	0	0	0	1	0	0	0
750	0	0	0	0	0	0	0	1	0	0
725	0	0	0	0	0	0	0	0	1	0
720	0	0	0	0	0	0	0	0	0	1

Table 6. Diagonal matrix for 1020 mm

CHAPTER 8

MATHEMATICAL MODELING

In mathematical modeling part, two models were analyzed. First model was used in order to compare Gilmore-Gomory Algorithm's solutions. In the model, all possible combinations of cutting patterns were created. That model can be used for standard roll and multi width rolls. The aim of this model is minimizing number of used rolls. In order to obtain optimal solution all combinations were entered by hand.

The second model is an extra model in order to describe that minimizing trim loss and minimizing number of used rolls are different. This model can be used for standard roll and multi width rolls too. Coefficient of objective function was used to explain using different roll width. The aim is minimizing trim loss at this time.

8.1. MATHEMATICAL FORMULATION

SETS&INDICES

I Set of desired widths

J Set of cutting patterns

i Index of set desired widths, $i \in I = \{1, 2, \dots, |I|\}$

j Index of set cutting patterns, $j \in J = \{1, 2, \dots, |J|\}$

PARAMETRES

a_{ij} The number of width i in cutting pattern j

b_j The demand for subrolls of width i

DECISION VARIABLES

x_j The number of rolls for which cutting pattern j is used

$$\text{Minimize } \sum_{j \in J} x_j \quad (1)$$

s.t.

$$\sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \quad (2)$$

$$x_j \geq 0 \text{ and integer } \forall j \in J \quad (3)$$

The objective function (1) denotes the total number of standard rolls which satisfy the demand, and has to be minimized.

Second constraint is written for the fact that the number of produced subrolls has to be greater than or equal to the number of demanded subrolls.

Constraints (3) show the domain of variables.

The above formulation can be extended to one-dimensional problem with multiple standard widths ($w_k, k = 1, 2, \dots, |K|$) with a fixed length L . For each standard width w_k , let n_k be the number of patterns, x_{jk} be the number of the j th pattern to be cut, and c_{jk} be the associated cost of cutting each j th pattern. Then the j th pattern can be represented a_{ijk} with l th component.

The IP model for multiple standard widths:

$$\text{Minimize } \sum_{k \in K} \sum_{j \in J} c_{jk} x_{jk} \quad (4)$$

s.t.

$$\sum_{k \in K} \sum_{j \in J} a_{ijk} x_{jk} \geq b_i \quad \forall i \in I \quad (5)$$

$$x_{jk} \geq 0 \text{ and integer } \quad \forall j \in J, \forall k \in K \quad (6)$$

The objective function (4) minimizes the total associated cost of cutting patterns.

Constraints (5) ensure the same condition like in the model of one standard roll. Sixth constraints show the domain of variables.

CHAPTER 9

APPLICATION OF GILMORE-GOMORY'S ALGORITHM

The cardboard factory manufacturers cuts cardboards. The cutting department buys the cardboard such as rolls with different widths. They have to be cut into subrolls with desired widths. The different lengths of rolls in this instance are 1020 mm, 1120 mm, 1220 mm, 1320 mm, 1430 mm, 1530 mm, 1630 mm, 1730 mm, 1830 mm, 1930 mm and 2000 mm. The customer order subrolls of cardboard have various widths. It means that every demand has a different width. The question is how to cut the standard rolls such that the demand of all customers is satisfied, and the amount of wasted cardboard which is called the trim loss is as small as possible. Our aim is to generate the best cutting patterns for each order package with cutting minimum amount of rolls.

Finding all cutting patterns can be possible for a small order package. But the number of cutting patterns can easily grow in some problem which includes large-scale orders. In this part, how to solve general model for a standard roll without using all cutting patterns will be showed, and hence without considering all decision variables x_1, x_2, \dots, x_n .

The matrix **A** will only exist actually; it is never constructed clearly. Only a submatrix of **A** is generated for this solution approach. There are two phases in each iteration step in Gilmore-Gomory algorithm,. A LO-model is solved in the first phase, and the second phase occurs of a knapsack model

to be solved, which either creates optimality, or leads to a cutting pattern that is added to the submatrix of \mathbf{A} .

The formulation of the algorithm is as follows:

9.1. Gilmore-Gomory Algorithm

Input: Model, with an order package, including the amount of demanded subrolls with the corresponding widths.

Output: An optimal solution of model.

Step 0: Initialization. Choose an initial full row rank matrix $\mathbf{A}^{(1)}$, of which the columns correspond to cutting patterns. For instance, take $\mathbf{A}^{(1)} = \mathbf{I}_m$, Go to Step 1.

Step 1: This step is for Simplex algorithm. Let $\mathbf{A}^{(k)}$, $k \geq 1$, be the current technology matrix (after k iterations of the Gilmore-Gomory algorithm), of which the columns correspond to cutting patterns; let $J(k)$ be the index set of the columns of $\mathbf{A}^{(k)}$. Solve the LO-model:

$$\begin{aligned} \min \quad & \sum_{j \in J(k)} x_j \\ \text{s.t.} \quad & \sum_{j \in J(k)} A_j^{(k)} x_j = b \quad (P_k) \\ & x_j \geq 0 \text{ for } j \in J(k), \end{aligned}$$

with $\mathbf{A}_j^{(k)}$ the j 'th column of $\mathbf{A}^{(k)}$. Let $y_1^{(k)}, \dots, y_m^{(k)}$ be the values of an optimal dual solution, corresponding to the current optimal basis matrix of (P_k) .

Go to Step 2.

Step 2: Column generation. Solve the knapsack model:

$$\begin{aligned} \max \quad & \sum_{i=1}^m y_i^{(k)} u_i \\ \text{s.t.} \quad & \sum_{i=1}^m w_i u_i \leq 1730 \quad (K_k) \\ & u_1, \dots, u_m \geq 0. \end{aligned}$$

Let $\mathbf{u}^{(k)} = [u_1^{(k)} \dots u_m^{(k)}]^T$ be an optimal solution of (K_k) , and let α_k be the optimal objective value of (K_k) . Go to Step 3.

Step 3: Optimality test and stopping rule. If $\alpha_k > 1$, then let $\mathbf{A}^{(k+1)} = [\mathbf{A}^{(k)} \mathbf{u}^{(k)}]$ and return to Step I. If $\alpha_k \leq 1$, then stop: the pair $\{\mathbf{u}^{(k)}, \alpha_k\}$ is an optimal solution of model.

In the thesis, firstly we arranged matrices for the rolls according to demand widths. It means that there are 11 matrices for the demands. These matrices are diagonal matrices. In the matrix, there are 10 different desired widths. These are 500,450,645,430,370,495,850,750,725,720 mm and their frequencies is called $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$. For instance; there are two '500 mm' in the 1020 mm roll. There are two '450 mm' in the 1020 mm roll. There is one '645 mm' in the 1020 mm roll.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
500	2	0	0	0	0	0	0	0	0	0
450	0	2	0	0	0	0	0	0	0	0
645	0	0	1	0	0	0	0	0	0	0
430	0	0	0	2	0	0	0	0	0	0
370	0	0	0	0	2	0	0	0	0	0
495	0	0	0	0	0	2	0	0	0	0
850	0	0	0	0	0	0	1	0	0	0
750	0	0	0	0	0	0	0	1	0	0
725	0	0	0	0	0	0	0	0	1	0
720	0	0	0	0	0	0	0	0	0	1

Table 7. Initial Matrix for 1020 mm

The details of the first iteration of the algorithm is as follows:

Step 1. Solve the LO-model:

Minimize $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$

s.t.

$$2x_1 \geq 10$$

$$2x_2 \geq 20$$

$$x_3 \geq 50$$

$$2x_4 \geq 60$$

$$2x_5 \geq 40$$

$$2x_6 \geq 45$$

$$x_7 \geq 55$$

$$x_8 \geq 65$$

$$x_9 \geq 80$$

$$x_{10} \geq 45 \quad x_1, \dots, x_{10} \geq 0.$$

An optimal solution is calculated using a computer package, CPLEX:

$x_1=5, x_2=10, x_3=50, x_4=30, x_5=20, x_6=22.5, x_7=55, x_8=65, x_9=80, x_{10}=45$, with optimal objective value $z=382.5$.

The knapsack problem to be solved in next step uses as objective coefficients the optimal dual values of the current constraints:

$y_1=y_2=y_4=y_5=y_6=0.5, y_3=y_7=y_8=y_9=y_{10}=1$. (These optimal dual values are reported in the output of the computer package.)

Step 2. Solve the knapsack problem

$$\text{Maximize } 0.5u_1+0.5u_2+0.1u_3+0.5u_4+0.5u_5+0.5u_6+u_7+u_8+u_9+u_{10}$$

s.t

$$500u_1+450u_2+645u_3+430u_4+370u_5+495u_6+850u_7+750u_8+725u_9+720u_{10} \leq 1020$$

$u_1, \dots, u_{10} \geq 0$, and integer.

The optimal solution (generated by a computer package) is:

$u_1=u_2=u_4=u_6=u_7=u_8=u_9=u_{10}=0, u_3=u_5=1$. The optimal objective value satisfies

$\alpha_1 = 1.5$. Since $\alpha_1 > 1$, optimality has not yet been reached.

Step 3. Construct matrix $\mathbf{A}^{(2)}$ from $\mathbf{A}^{(1)}$ by adding the column

$$[0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]^T.$$

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}
500	2	0	0	0	0	0	0	0	0	0	0
450	0	2	0	0	0	0	0	0	0	0	0
645	0	0	1	0	0	0	0	0	0	0	1
430	0	0	0	2	0	0	0	0	0	0	0
370	0	0	0	0	2	0	0	0	0	0	1
495	0	0	0	0	0	2	0	0	0	0	0
850	0	0	0	0	0	0	1	0	0	0	0
750	0	0	0	0	0	0	0	1	0	0	0
725	0	0	0	0	0	0	0	0	1	0	0
720	0	0	0	0	0	0	0	0	0	1	0

Table 8. Iteration matrix

In the next iteration, Step 1 and Step 2 are repeated for the matrix $\mathbf{A}^{(2)}$. The algorithm stops at iteration k if the optimal objective value α_k of model (K_k) is ≤ 1 . With our calculations, the Gilmore-Gomory algorithm performed 3 iterations. The resulting optimal solution is listed below:

Optimal Solution											
# of Rolls	5	10	10	22,5	55	65	80	45	40	30	
Order(mm)	Cutting Pattern										Optimal Dual Value
500	2	0	0	0	0	0	0	0	0	0	0,5
450	0	2	0	0	0	0	0	0	0	0	0,5
645	0	0	1	0	0	0	0	0	1	0	1
430	0	0	0	0	0	0	0	0	0	2	0,5
370	0	0	0	0	0	0	0	0	1	0	0
495	0	0	0	2	0	0	0	0	0	0	0,5
850	0	0	0	0	1	0	0	0	0	0	1
750	0	0	0	0	0	1	0	0	0	0	1
725	0	0	0	0	0	0	1	0	0	0	1
720	0	0	0	0	0	0	0	1	0	0	1
TRIM LOSS CALCULATION (mm)	100	1200	3750	675	9350	17550	23600	13500	200	4800	
TOTAL TRIM LOSS (mm)	74725										

Table 9. Algorithm Result

In previous explanations and details, as you can see, the algorithm was performed for a standard roll width, 1020 mm. Our main goal for this thesis is applying this algorithm to multiple standard roll widths. We made some modifications for the algorithm and the details are given below:

9.2. Improved Gilmore-Gomory Algorithm for Multiple Standard Rolls

Input: Model, with an order package, including the amount of demanded subrolls with the corresponding widths.

Output: An optimal solution of model.

Step 0: Initialization. Choose an initial full row rank matrix $\mathbf{A}^{(1)}$, of which the columns correspond to cutting patterns. For instance, take $\mathbf{A}^{(1)} = \mathbf{I}_m$. Go to Step 1.

Step 1: Simplex algorithm step. Let $\mathbf{A}^{(k)}$, $k \geq 1$, be the current technology matrix (after k iterations of the Gilmore-Gomory algorithm), of which the columns correspond to cutting patterns; let $J(k)$ be the index set of the columns of $\mathbf{A}^{(k)}$. Solve the LO-model:

$$\begin{aligned} \min \quad & \sum_{j \in J(k)} x_j \\ \text{s.t.} \quad & \\ & \sum_{j \in J(k)} A_j^{(k)} x_j = b \quad (P_k) \\ & x_j \geq 0 \quad \text{for } j \in J(k), \end{aligned}$$

with $\mathbf{A}_j^{(k)}$ the j 'th column of $\mathbf{A}^{(k)}$. Let $y_1^{(k)}, \dots, y_m^{(k)}$ be the values of an optimal dual solution, corresponding to the current optimal basis matrix of (P_k) .

Go to Step 2.

Step 2: Column generation. Solve the knapsack model:

$$\begin{aligned} \max \quad & \sum_{j=1}^n \sum_{i=1}^m y_i^{(k)} u_{ji} \\ \text{s.t.} \quad & \\ & \sum_{i=1}^m w_i u_{ji} \leq \text{width}_j \quad \forall j \in J \quad (K_k) \\ & u_{ji} \geq 0 \quad \forall i \in I \quad \forall j \in J \end{aligned}$$

The different point from the original algorithm is that in the optimal solution of (K_k) , we can have more than one \mathbf{u} vector and we take into account all of these \mathbf{u} vectors, and let α_k be the optimal objective value of (K_k) .

Go to Step 3.

Step 3: Optimality test and stopping rule. In the solution of Step 2, if there is/are different \mathbf{u} vector(s) from previous iterations, return to Step I. If there is no different \mathbf{u} vector for adding to \mathbf{A} vector, then stop: Optimality has been reached.

The given example will help the reader to understand the improved algorithm. In the example, the instance consists of two different standard rolls, 1020 mm and 1730 mm. Before starting to the improved algorithm, we first arranged the diagonal matrix for this instance which is listed below:

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
500	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
450	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
645	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
430	0	0	0	2	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
370	0	0	0	0	2	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
495	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0
850	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0
750	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0
725	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0
720	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2
ROLL WIDTH	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1730	1730	1730	1730	1730	1730	1730	1730	1730	1730

Table 10. Initial Matrix for 1020 mm and 1730 mm

The details of the first iteration is as follows:

Step 1. Solve the LO-model:

Minimize

$$X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8 + X_9 + X_{10} + X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{17} + X_{18} + X_{19} + X_{20}$$

s. t.

$$2X_1 + 3X_{11} \geq 10$$

$$2X_2 + 3X_{12} \geq 20$$

$$X_3 + 2X_{13} \geq 50$$

$$2X_4 + 4X_{14} \geq 60$$

$$2X_5 + 4X_{15} \geq 40$$

$$2X_6 + 3X_{16} \geq 45$$

$$X_7 + 2X_{17} \geq 55$$

$$X_8 + 2X_{18} \geq 65$$

$$x_9 + 2x_{19} \geq 80$$

$$x_{10} + 2x_{20} \geq 45$$

$$x_1, \dots, x_{20} \geq 0.$$

An optimal solution is calculated using a computer package, CPLEX:

$x_1 = x_2 = x_3 = x_4 = x_5 = x_6 = x_7 = x_8 = x_9 = 0$, $x_{10} = 45$, $x_{11} = 3.33$, $x_{12} = 6.66$, $x_{13} = 25$, $x_{16} = x_{14} = 15$, $x_{15} = 10$, $x_{17} = 27.5$, $x_{18} = 32.5$, $x_{19} = 40$, $x_{20} = 22.5$, with optimal objective value $z = 197.5$. The knapsack problem to be solved in next step uses as objective coefficients the optimal dual values of the current constraints: $y_1 = y_2 = y_6 = 0.33$, $y_4 = y_5 = 0.25$, $y_3 = y_7 = y_8 = y_9 = y_{10} = 0.5$. (These optimal dual values are reported in the output of the computer package.)

Step 2. Solve the knapsack problem

Maximize

$$0.33(u_{11} + u_{21}) + 0.33(u_{12} + u_{22}) + 0.5(u_{13} + u_{23}) + 0.25(u_{14} + u_{24}) + 0.25(u_{15} + u_{25}) + 0.33(u_{16} + u_{26}) + 0.5(u_{17} + u_{27}) + 0.5(u_{18} + u_{28}) + 0.5(u_{19} + u_{29}) + 0.5(u_{110} + u_{210})$$

s.t

$$500u_{11} + 450u_{12} + 645u_{13} + 430u_{14} + 370u_{15} + 495u_{16} + 850u_{17} + 750u_{18} + 725u_{19} + 720u_{110} \leq 1020$$

$$500u_{21} + 450u_{22} + 645u_{23} + 430u_{24} + 370u_{25} + 495u_{26} + 850u_{27} + 750u_{28} + 725u_{29} + 720u_{210} \leq 1730$$

$u_{11}, \dots, u_{210} \geq 0$, and integer.

The optimal solution (generated by a computer package) is:

$$u_{11} = u_{12} = u_{14} = u_{16} = u_{17} = u_{18} = u_{19} = u_{110} = 0, u_{13} = u_{15} = 1,$$

$u_{21} = u_{22} = u_{24} = u_{25} = u_{26} = u_{27} = u_{28} = u_{29} = u_{210} = 0$, $u_{23} = 1$, The optimal objective value satisfies $\alpha_1 = 2$. Since we have different u vectors to enter the matrix, optimality has not yet been reached.

Step 3. Construct matrix $\mathbf{A}^{(2)}$ from $\mathbf{A}^{(1)}$ by adding the columns

$$[0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]^T, [0 \ 0 \ 2 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]^T.$$

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
500	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
450	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
645	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	2
430	0	0	0	2	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
370	0	0	0	0	2	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	1
495	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
850	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
750	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0
725	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0
720	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0
ROLL WIDTH	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1730	1730	1730	1730	1730	1730	1730	1730	1730	1730	1020	1730

Table 11. Iteration Matrix

In the next iteration, Step 1 and Step 2 are repeated for the matrix $\mathbf{A}^{(2)}$. The algorithm stops at iteration k if there is no different \mathbf{u} vector from previous iterations. With our calculations, the Gilmore-Gomory algorithm performed 6 iterations. The resulting optimal solution is listed below:

Optimal Solution											
# of Rolls	40	32,5	27,5	25	22,5	15	8,75	6,66	5	2,083	
Order(mm)	Cutting Pattern										Optimal Dual Value
500	0	0	0	0	0	0	0	0	2	0	0,25
450	0	0	0	0	0	0	0	3	0	0	0,25
645	0	0	0	2	0	0	0	0	0	0	0,375
430	0	0	0	0	0	4	0	0	0	0	0,25
370	0	0	0	1	0	0	0	1	0	4	0,25
495	0	0	0	0	2	0	0	0	0	0	0,25
850	0	0	2	0	0	0	0	0	0	0	0,5
750	0	2	0	0	0	0	0	0	0	0	0,5
725	2	0	0	0	0	0	0	0	0	0	0,5
720	0	0	0	0	1	0	2	0	1	0	0,5
TRIM LOSS CALCULATION (mm)	11200	7475	825	1750	450	150	2538	66,6	50	521	
TOTAL TRIM LOSS (mm)	25024,85										

Table 12. Algorithm Result

The verification of this algorithm is constructed by solving the problem with the original model which includes all cutting patterns. The results show that this algorithm also gives optimal solution.

In addition, a lot of combinations of different standard roll widths (4-7-8-10 and 11 different standard roll combinations) are analyzed with this algorithm. In the appendix part, the reader can see the results.

When the reader sees the results, he/she can be sure that this approach gives optimality without considering all cutting patterns.

CHAPTER 10

RESULTS

In the thesis, mathematical was improved. This model was designed according to eleven different rolls and all possible cutting pattern combinations were occurred . Gilmore-Gomory's Algorithm firstly was applied for standard roll and then algorithm was improved and applied for eleven different roll widths. The aim is minimizing number of used rolls while cutting process in mathematical model and Gilmore-Gomory's Algorithm. So, results which were taken from mathematical model and Gilmore-Gomory's Algorithm were compared. It has seen that used cutting patterns and number of used rolls were same. In order to show these solutions comparing tables were prepared.

# of used rolls	Roll widths	Mathematical model solution	Algorithm solution
11	1020-1120-1220-1320-1430-1530-1630-1730-1830-1930-2000	149,667	149,667
10	1020-1120-1220-1430-1530-1630-1730-1830-1930-2000	149,667	149,667
8	1020-1220-1430-1530-1730-1830-1930-2000	149,667	149,667
7	1020-1220-1430-1530-1730-1830-2000	149,667	149,667
6	1020-1220-1430-1530-1730-2000	149,667	149,667
5	1020-1220-1430-1530-1730	185	185
4	1020-1220-1530-1730	185	185
3	1020-1220-1730	185	185
1	1730	185	185
1	1020	382,5	382,5

Table 13. Solution table for Mathematical model and Gilmore-Gomory Algorithm

CHAPTER 11

CONCLUSION

In conclusion, cutting stock problem was defined in detail. Surveys, mathematical models, and heuristic applications were explained in the literature surveys part. Mathematical model was developed and it was improved according to our problem generating all possible cutting patterns in order to find optimal solution and calculated trim loss which means that used number of rolls while cutting processing. Then, Gilmore-Gomory Algorithm was studied for only one roll width and this model was improved according to different roll widths which is the first application in the literature. Before, all of the studies in the literature were done for the one roll width. But in this thesis, some different roll widths were used and calculated how many rolls should be used while processing. Algorithm was studied with two, three, seven, eight, ten and eleven different roll widths. So, demands were satisfied and how many rolls which should be required for the production process were selected. These studies were coded in an optimization programming language, CPLEX. Solutions which were taken from mathematical model and improved Gilmore-Gomory Algorithm were compared and it has been seen that solutions were same.

REFERENCES

- [1] ACHHALZ H. and A. KONRAD (1998), "Operations Research Schont den Wald" R-News, Juli 1998.
- [2] KLAUS K.P. and M. STEVEN (1993), Produktionsplanung, Physical Verlag.
- [3] KUPSCH P.U. (1979), Lager in Kern, Handwörterbuch der Produktionswirtschaft, Stuttgart.
- [4] MÜLLER M. (1973), Operations-Research , 3. Auflage, Verlag Vahlen.
- [5] WALTER D. and K. KLEIBOHRM (1988), Operations Research, Carl Hanser Verlag München Wien.
- [6] K. Claire and E. Remila, "A near-optimal solution to a two-dimensional cutting stock problem", Math. Oper. Res. 25 no. 4 (2000), 645-656.
- [7] V. Chvatal, Linear programming, W.H. Freeman and Company, New York, 1983.
- [8] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem, part i", Journal of Operation Research 9 (1961).

-
- [9] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem, part ii", *Journal of Operation Research* 9 (1961).
- [10] M. Hifi, "Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems", *Journal of Combinatorial Optimization* 8 (2004).
- [11] F. Vanderbeck, "Computational Study of a Column Generation Algorithm for Bin Packing and Cutting Stock Problems", *Journal of Mathematical Programming* 86 no.33 (1999).

APPENDIX A

Initial Matrices for the different roll widths

A.1. Initial matrix for roll 2000 mm

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
500	4	0	0	0	0	0	0	0	0	0
450	0	4	0	0	0	0	0	0	0	0
645	0	0	3	0	0	0	0	0	0	0
430	0	0	0	4	0	0	0	0	0	0
370	0	0	0	0	5	0	0	0	0	0
495	0	0	0	0	0	4	0	0	0	0
850	0	0	0	0	0	0	2	0	0	0
750	0	0	0	0	0	0	0	2	0	0
725	0	0	0	0	0	0	0	0	2	0
720	0	0	0	0	0	0	0	0	0	2

$A_1 =$

5. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5
optimal value=)156,5						
dual v.1	0,2	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0
dual v.3	0,33	0	1	1	0	0
dual v.4	0,166	0	0	0	0	0
dual v.5	0,2	0	0	1	1	0
dual v.6	0,25	1	0	0	0	0
dual v.7	0,5	1	1	1	1	1
dual v.8	0,416	0	0	0	1	1
dual v.9	0,4	0	0	0	0	0
dual v.10	0,4	0	0	0	0	0
alpha=8,497						

6. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6
optimal value=)155,625							
dual v.1	0,833	0	0	0	0	0	0
dual v.2	0,833	0	0	0	0	0	0
dual v.3	0,33	0	0	1	0	0	0
dual v.4	0,166	0	0	0	1	0	0
dual v.5	0,833	0	1	0	0	0	0
dual v.6	0,25	0	0	0	0	1	0
dual v.7	0,5	1	0	0	0	0	1
dual v.8	0,416	0	0	0	0	0	0
dual v.9	0,458	0	0	0	0	0	0
dual v.10	0,458	0	1	1	2	2	1
alpha=8,662							

7. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)153,236								
dual v.1	0,2	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	0	0	0	0	0	0	0
dual v.4	0,244	0	1	1	4	1	2	2
dual v.5	0,2	0	0	1	0	0	1	0
dual v.6	0,25	1	2	0	0	0	0	0
dual v.7	0,422	0	0	0	0	0	0	0
dual v.8	0,377	0	0	0	0	0	0	0
dual v.9	0,4	1	0	1	0	2	1	1
dual v.10	0,375	0	0	0	0	0	0	0
alpha=8,367								

8. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5
optimal value=)152,292						
dual v.1	0,22	0	0	0	0	0
dual v.2	0,22	0	0	0	0	1
dual v.3	0,33	0	0	1	0	1
dual v.4	0,22	0	0	1	1	0
dual v.5	0,166	0	0	0	0	0
dual v.6	0,25	2	2	0	0	0
dual v.7	0,444	0	0	0	1	0
dual v.8	0,388	0	0	0	0	0
dual v.9	0,388	0	1	1	0	0
dual v.10	0,375	0	0	0	1	0
alpha=7,895						

9. Iteration		dec. V1	dec. V2	dec. V3
optimal value=)151,354				
dual v.1	0,208	0	0	0
dual v.2	0,208	0	0	0
dual v.3	0,333	1	0	0
dual v.4	0,208	0	0	0
dual v.5	0,1875	0	0	1
dual v.6	0,25	0	1	1
dual v.7	0,416	0	0	0
dual v.8	0,395	0	0	0
dual v.9	0,395	1	2	1
dual v.10	0,375	0	0	0
alpha=8,007				

10. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7	dec. V8
optimal value=)151,131									
dual v.1	0,25	0	0	0	0	0	0	0	0
dual v.2	0,25	1	3	1	3	4	1	3	2
dual v.3	0,33	0	0	1	0	0	0	1	0
dual v.4	0,214	0	0	1	0	0	0	0	0
dual v.5	0,196	2	0	0	1	0	4	0	0
dual v.6	0,25	0	0	0	0	0	0	0	0
dual v.7	0,41	0	0	0	0	0	0	0	0
dual v.8	0,392	0	0	0	0	0	0	0	0
dual v.9	0,375	0	0	0	0	0	0	0	1
dual v.10	0,375	0	0	0	0	0	0	0	0
alpha=8,244									

11. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)150,532					
dual v.1	0,25	0	0	1	1
dual v.2	0,222	0	0	0	0
dual v.3	0,333	0	0	0	0
dual v.4	0,212	0	0	0	0
dual v.5	0,194	2	3	0	1
dual v.6	0,25	2	0	0	0
dual v.7	0,412	0	0	0	0
dual v.8	0,393	0	0	2	1
dual v.9	0,375	0	0	0	0
dual v.10	0,375	0	1	0	0
alpha=7,967					

12. Iteration		dec. V1	dec. V2
optimal value=)150,162			
dual v.1	0,212	0	0
dual v.2	0,222	0	0
dual v.3	0,333	0	0
dual v.4	0,212	0	0
dual v.5	0,194	0	1
dual v.6	0,25	1	1
dual v.7	0,412	0	0
dual v.8	0,393	2	1
dual v.9	0,375	0	0
dual v.10	0,375	0	0
alpha=7,967			

2. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6
optimal value=)	175,29						
dual v.1	0,25	0	0	0	0	0	1
dual v.2	0,25	0	0	0	0	1	0
dual v.3	0,33	0	1	0	0	0	0
dual v.4	0,25	1	0	0	0	0	0
dual v.5	0,2	0	0	0	1	0	0
dual v.6	0,25	0	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0	0
dual v.8	0,5	0	0	0	0	0	0
dual v.9	0,5	1	1	2	2	2	2
dual v.10	0,375	0	0	0	0	0	0
alpha=7,81							

3. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5
optimal value=)	166,91					
dual v.1	0,2	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0
dual v.3	0,33	0	1	0	1	0
dual v.4	0,25	1	0	0	1	1
dual v.5	0,2	0	0	0	0	0
dual v.6	0,25	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0
dual v.8	0,5	1	1	2	1	2
dual v.9	0,4	0	0	0	0	0
dual v.10	0,4	0	0	0	0	0
alpha=7,69						

4. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6
optimal value=)	158,79						
dual v.1	0,2	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0
dual v.3	0,33	0	0	0	0	1	0
dual v.4	0,25	0	3	1	0	1	1
dual v.5	0,2	1	0	1	0	0	0
dual v.6	0,25	0	0	0	0	0	0
dual v.7	0,5	1	0	0	2	1	1
dual v.8	0,375	0	0	0	0	0	0
dual v.9	0,4	0	0	0	0	0	0
dual v.10	0,4	0	0	1	0	0	1
alpha=7,06							

5. Iteration		dec. V1	dec. V2
optimal value=)154,66			
dual v.1	0,2	0	0
dual v.2	0,2	0	0
dual v.3	0,33	0	0
dual v.4	0,09	0	0
dual v.5	0,2	1	0
dual v.6	0,25	0	1
dual v.7	0,5	0	0
dual v.8	0,45	2	2
dual v.9	0,4	0	0
dual v.10	0,4	0	0
alpha=7,16			

6. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)151,208					
dual v.1	0,2	0	0	0	0
dual v.2	0,2	0	0	0	0
dual v.3	0,33	0	0	0	0
dual v.4	0,25	1	4	1	2
dual v.5	0,2	1	0	0	1
dual v.6	0,25	0	0	0	0
dual v.7	0,416	0	0	0	0
dual v.8	0,375	0	0	0	0
dual v.9	0,4	1	0	2	1
dual v.10	0,333	0	0	0	0
alpha=6,93					

7. Iteration		dec. V1	dec. V2
optimal value=)150,95			
dual v.1	0,2	0	0
dual v.2	0,2	0	0
dual v.3	0,33	0	0
dual v.4	0,2	0	0
dual v.5	0,2	0	1
dual v.6	0,25	0	0
dual v.7	0,466	0	1
dual v.8	0,375	0	0
dual v.9	0,4	2	1
dual v.10	0,33	0	0
alpha=6,724			

8. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)150,29					
dual v.1	0,2	0	0	0	0
dual v.2	0,2	0	0	0	0
dual v.3	0,33	0	0	0	0
dual v.4	0,2	0	0	0	0
dual v.5	0,2	0	0	0	0
dual v.6	0,25	1	2	1	1
dual v.7	0,4	0	0	0	0
dual v.8	0,375	0	0	0	0
dual v.9	0,4	0	0	0	0
dual v.10	0,4	1	1	2	2
alpha=6,66					

9. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6
optimal value=)150,25							
dual v.1	0,233	0	0	0	0	0	0
dual v.2	0,333	0	0	0	0	1	0
dual v.3	0,333	0	0	0	0	0	0
dual v.4	0,2083	0	0	0	0	0	0
dual v.5	0,2	0	4	2	3	4	4
dual v.6	0,25	1	0	2	0	0	1
dual v.7	0,416	0	0	0	0	0	0
dual v.8	0,375	0	0	0	0	0	0
dual v.9	0,383	1	0	0	0	0	0
dual v.10	0,375	0	0	0	1	0	0
alpha=6,57							

10. Iteration		dec. V1	dec. V2
optimal value=)150			
dual v.1	0,2083	0	0
dual v.2	0,2083	0	0
dual v.3	0,333	0	0
dual v.4	0,2083	0	0
dual v.5	0,1875	0	0
dual v.6	0,25	2	1
dual v.7	0,416	0	0
dual v.8	0,375	0	0
dual v.9	0,395	1	2
dual v.10	0,375	0	0
alpha=6,593			

11. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)149,94								
dual v.1	0,25	0	0	0	0	0	0	0
dual v.2	0,25	1	3	1	3	4	2	3
dual v.3	0,333	0	0	1	0	0	1	1
dual v.4	0,21875	0	0	1	0	0	0	0
dual v.5	0,1875	0	0	0	1	0	1	0
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,4375	0	0	0	0	0	0	0
dual v.8	0,375	1	0	0	0	0	0	0
dual v.9	0,375	0	0	0	0	0	0	0
dual v.10	0,34375	0	0	0	0	0	0	0
alpha=6,736								

12. Iteration		dec. V1	dec. V2	dec. V3
optimal value=)149,79				
dual v.1	0,22	0	0	0
dual v.2	0,22	1	0	2
dual v.3	0,33	0	2	0
dual v.4	0,2083	0	1	0
dual v.5	0,194	0	0	1
dual v.6	0,222	0	0	0
dual v.7	0,416	0	0	0
dual v.8	0,38	0	0	0
dual v.9	0,38	1	0	1
dual v.10	0,374	0	0	0
alpha=8,59				

12. Iteration
optimal value=)149,667
alpha=6,532

B.3.1020mm-1220mm-1430mm-1530mm-1730mm-1830mm-2000mm

1. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)180,917								
dual v.1	0,25	0	1	0	0	0	0	1
dual v.2	0,25	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,25	0	0	0	0	0	0	0
dual v.5	0,2	1	0	0	0	0	1	0
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0	0	0
dual v.8	0,5	0	0	1	0	0	0	0
dual v.9	0,5	0	0	0	0	0	0	0
dual v.10	0,5	0	1	0	2	2	2	2
alpha=6,56								

2. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)175,91								
dual v.1	0,2	0	0	0	0	0	0	0
dual v.2	0,25	0	1	0	0	0	0	1
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,25	0	0	0	0	0	0	0
dual v.5	0,2	1	0	0	0	0	1	0
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0	0	0
dual v.8	0,5	0	1	0	0	0	0	0
dual v.9	0,5	0	0	1	2	2	2	2
dual v.10	0,4	0	0	0	0	0	0	0
alpha=6,56								

3. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)166,91								
dual v.1	0,2	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	1	0
dual v.4	0,25	0	1	0	0	0	1	1
dual v.5	0,2	1	0	0	0	0	0	0
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0	0	0
dual v.8	0,5	0	1	1	2	2	1	2
dual v.9	0,4	0	0	0	0	0	0	0
dual v.10	0,4	0	0	0	0	0	0	0
alpha=6,44								

4. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)158,79								
dual v.1	0,2	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	1	0	0	0	0	0	0
dual v.4	0,25	0	0	3	1	0	0	1
dual v.5	0,2	1	1	0	1	0	0	0
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,5	0	1	0	0	2	2	1
dual v.8	0,375	0	0	0	0	0	0	0
dual v.9	0,4	0	0	0	0	0	0	0
dual v.10	0,4	0	0	0	1	0	0	1
alpha=5,98								

5. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)154,72								
dual v.1	0,25	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,125	0	0	0	0	0	0	0
dual v.5	0,2	1	1	0	0	0	0	1
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,5	0	1	0	0	0	2	1
dual v.8	0,4375	0	0	1	2	2	0	1
dual v.9	0,4	0	0	0	0	0	0	0
dual v.10	0,375	0	0	0	0	0	0	0
alpha=6,01								

6. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)	152,20							
dual v.1	0,25	0	0	0	0	0	0	1
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,216	0	0	0	1	0	0	0
dual v.5	0,2	1	0	2	1	0	1	0
dual v.6	0,25	0	1	0	0	2	0	0
dual v.7	0,408	0	0	0	0	0	0	0
dual v.8	0,391	0	0	0	0	0	0	0
dual v.9	0,4	0	1	0	1	1	2	2
dual v.10	0,375	0	0	0	0	0	0	0
alpha=5,676								

7. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)	151,91							
dual v.1	0,2	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,2	0	0	0	0	0	0	0
dual v.5	0,2	1	0	2	0	0	1	0
dual v.6	0,25	0	1	0	0	0	0	1
dual v.7	0,4	0	0	0	0	0	0	0
dual v.8	0,4	0	0	0	0	0	0	0
dual v.9	0,4	0	0	0	2	0	0	0
dual v.10	0,4	0	1	0	0	1	2	2
alpha=5,66								

8. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)	151,70							
dual v.1	0,2	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,216	0	0	0	1	0	0	0
dual v.5	0,2	1	0	2	1	0	1	0
dual v.6	0,25	0	1	0	0	2	0	1
dual v.7	0,408	0	0	0	0	0	0	0
dual v.8	0,391	0	0	0	0	0	0	0
dual v.9	0,4	0	1	0	1	1	2	2
dual v.10	0,375	0	0	0	0	0	0	0
alpha=7,895								

9. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)151,208								
dual v.1	0,25	0	0	0	0	0	0	0
dual v.2	0,25	0	1	3	0	3	4	3
dual v.3	0,33	1	0	0	0	0	0	1
dual v.4	0,216	0	0	0	0	0	0	0
dual v.5	0,2	1	2	0	4	1	0	0
dual v.6	0,25	0	0	0	0	0	0	0
dual v.7	0,408	0	0	0	0	0	0	0
dual v.8	0,391	0	0	0	0	0	0	0
dual v.9	0,375	0	0	0	0	0	0	0
dual v.10	0,375	0	0	0	0	0	0	0
alpha=5,76								

10. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)150,652								
dual v.1	0,25	0	1	0	0	0	0	1
dual v.2	0,25	0	0	0	0	0	0	0
dual v.3	0,33	1	0	1	0	0	0	0
dual v.4	0,21	0	0	0	0	0	0	0
dual v.5	0,2	1	0	2	4	2	3	4
dual v.6	0,25	0	0	0	0	2	0	0
dual v.7	0,408	0	0	0	0	0	0	0
dual v.8	0,391	0	0	0	0	0	0	0
dual v.9	0,375	0	0	0	0	0	0	0
dual v.10	0,375	0	1	0	0	0	1	0

11. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)149,76								
dual v.1	0,22	0	0	0	0	0	0	0
dual v.2	0,222	0	0	0	0	0	0	0
dual v.3	0,333	1	0	0	0	0	0	0
dual v.4	0,24	0	1	3	1	4	0	2
dual v.5	0,194	1	0	0	1	0	0	3
dual v.6	0,22	0	2	0	0	0	1	0
dual v.7	0,425	0	0	0	0	0	0	0
dual v.8	0,379	0	0	0	0	0	0	0
dual v.9	0,388	0	0	0	1	0	2	0
dual v.10	0,333	0	0	0	0	0	0	0
alpha=5,69								

12. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7
optimal value=)149,66								
dual v.1	0,2	0	1	0	0	0	0	1
dual v.2	0,2	0	0	0	0	0	0	0
dual v.3	0,333	1	0	1	0	2	0	0
dual v.4	0,2	0	0	0	0	1	0	0
dual v.5	0,2	1	0	2	0	0	1	0
dual v.6	0,2	0	0	0	0	0	0	0
dual v.7	0,39	0	0	0	0	0	0	0
dual v.8	0,4	0	0	0	0	0	0	0
dual v.9	0,4	0	0	0	0	0	0	0
dual v.10	0,4	0	1	0	2	0	2	2
alpha=5,52								

13. Iteration	
optimal value=)149,667	

B.4.1020mm-1220mm-1530mm-1730mm

1. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)197,5					
dual v.1	0,33	0	1	0	0
dual v.2	0,33	0	0	0	0
dual v.3	0,5	1	1	1	2
dual v.4	0,25	0	0	1	0
dual v.5	0,25	1	0	0	1
dual v.6	0,33	0	0	1	0
dual v.7	0,5	0	0	0	0
dual v.8	0,5	0	0	0	0
dual v.9	0,5	0	0	0	0
dual v.10	0,5	0	0	0	0
alpha=3,91					

2. Iteration					
optimal value=)191,25		dec. V1	dec. V2	dec. V3	dec. V4
dual v.1	0,33	0	1	0	0
dual v.2	0,33	2	0	0	3
dual v.3	0,375	0	0	0	0
dual v.4	0,25	0	0	0	0
dual v.5	0,25	0	0	0	1
dual v.6	0,33	0	0	0	0
dual v.7	0,5	0	0	0	0
dual v.8	0,5	0	0	0	0
dual v.9	0,5	0	0	0	0
dual v.10	0,5	0	1	2	0
alpha=3,73					

3. Iteration					
optimal value=)189,58		dec. V1	dec. V2	dec. V3	dec. V4
dual v.1	0,33	1	1	0	0
dual v.2	0,25	0	0	0	0
dual v.3	0,375	0	0	0	0
dual v.4	0,25	0	0	0	0
dual v.5	0,25	0	0	0	0
dual v.6	0,33	1	0	0	2
dual v.7	0,5	0	0	0	0
dual v.8	0,5	0	0	0	0
dual v.9	0,5	0	0	0	0
dual v.10	0,5	0	1	2	1
alpha=3,65					

4. Iteration					
optimal value=)185,83		dec. V1	dec. V2	dec. V3	dec. V4
dual v.1	0,33	2	1	0	2
dual v.2	0,25	0	0	0	0
dual v.3	0,375	0	0	0	0
dual v.4	0,25	0	0	0	0
dual v.5	0,25	0	0	0	0
dual v.6	0,25	0	0	0	0
dual v.7	0,5	0	0	0	0
dual v.8	0,5	0	0	0	0
dual v.9	0,5	0	0	0	0
dual v.10	0,5	0	1	2	1
alpha=3,65					

5. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)185					
dual v.1	0,25	0	1	0	0
dual v.2	0,25	0	0	0	0
dual v.3	0,375	1	0	0	0
dual v.4	0,25	0	0	0	0
dual v.5	0,25	1	0	0	0
dual v.6	0,25	0	0	0	0
dual v.7	0,5	0	0	0	0
dual v.8	0,5	0	0	0	0
dual v.9	0,5	0	0	0	0
dual v.10	0,5	0	1	2	2
alpha=3,375					

6. Iteration	
optimal value=)185	

B.5.1020mm-1120mm-1220mm-1320mm-1430mm- 1530mm-1630mm-1730mm-1830mm-1930mm- 2000mm

1. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7	dec. V8
optimal value=)180,917									
dual v.1	0,25	0	0	1	0	0	0	0	1
dual v.2	0,25	0	0	0	0	0	0	1	0
dual v.3	0,33	1	0	0	1	0	0	0	0
dual v.4	0,25	0	0	0	0	0	0	0	0
dual v.5	0,2	1	1	0	0	0	1	0	0
dual v.6	0,25	0	0	0	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0	0	0	0
dual v.8	0,5	0	1	0	0	0	0	0	0
dual v.9	0,5	0	0	0	0	0	0	0	0
dual v.10	0,5	0	0	1	1	2	2	2	2
alpha=10,26									

2. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6
optimal value=)175,29							
dual v.1	0,25	0	0	0	0	0	1
dual v.2	0,25	0	0	0	0	1	0
dual v.3	0,33	0	0	1	0	0	0
dual v.4	0,25	0	1	0	0	0	0
dual v.5	0,2	1	0	0	1	0	0
dual v.6	0,25	0	0	0	0	0	0
dual v.7	0,5	0	0	0	0	0	0
dual v.8	0,5	0	0	0	0	0	0
dual v.9	0,5	1	1	1	2	2	2
dual v.10	0,375	0	0	0	0	0	0
alpha=10,26							

3. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)166,91					
dual v.1	0,2	0	0	0	0
dual v.2	0,2	0	0	0	0
dual v.3	0,33	1	0	1	0
dual v.4	0,25	0	0	1	1
dual v.5	0,2	0	0	0	0
dual v.6	0,25	0	0	0	0
dual v.7	0,5	0	0	0	0
dual v.8	0,5	1	2	1	2
dual v.9	0,4	0	0	0	0
dual v.10	0,4	0	0	0	0

alpha=10,14

4. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5	dec. V6	dec. V7	dec. V8
optimal value=)158,79									
dual v.1	0,2	0	0	0	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0	0	0	0
dual v.3	0,33	0	0	0	0	0	0	1	0
dual v.4	0,25	0	0	3	1	0	0	1	1
dual v.5	0,2	3	1	0	1	0	0	0	0
dual v.6	0,25	0	0	0	0	0	0	0	0
dual v.7	0,5	0	1	0	0	1	2	1	1
dual v.8	0,375	0	0	0	0	0	0	0	0
dual v.9	0,4	0	0	0	0	1	0	0	0
dual v.10	0,4	0	0	0	1	0	0	0	1

alpha=9,31

5. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)154,66					
dual v.1	0,2	0	0	0	0
dual v.2	0,2	1	0	0	0
dual v.3	0,33	0	0	0	0
dual v.4	0,099	0	0	0	0
dual v.5	0,2	0	0	1	0
dual v.6	0,25	0	0	0	1
dual v.7	0,5	1	1	0	0
dual v.8	0,45	0	1	2	2
dual v.9	0,4	0	0	0	0
dual v.10	0,4	0	0	0	0

alpha=9,46

6. Iteration		dec. V1	dec. V2	dec. V3	dec. V4	dec. V5
optimal value=)154,66						
dual v.1	0,2	0	0	0	0	0
dual v.2	0,2	0	0	0	0	0
dual v.3	0,33	0	0	0	0	0
dual v.4	0,25	1	2	4	1	2
dual v.5	0,2	1	0	0	0	1
dual v.6	0,25	0	0	0	0	0
dual v.7	0,416	0	0	0	0	0
dual v.8	0,375	0	0	0	0	0
dual v.9	0,4	1	1	0	2	1
dual v.10	0,333	0	0	0	0	0
alpha=9,18						

7. Iteration		dec. V1	dec. V2
optimal value=)150,95			
dual v.1	0,2	0	0
dual v.2	0,2	0	0
dual v.3	0,33	0	0
dual v.4	0,2	0	0
dual v.5	0,2	0	1
dual v.6	0,25	0	0
dual v.7	0,466	0	1
dual v.8	0,375	0	0
dual v.9	0,4	2	1
dual v.10	0,333	0	0
alpha=8,862			

8. Iteration		dec. V1	dec. V2	dec. V3	dec. V4
optimal value=)150,29					
dual v.1	0,2	0	0	0	0
dual v.2	0,2	0	0	0	0
dual v.3	0,33	0	2	0	0
dual v.4	0,2	0	0	0	0
dual v.5	0,2	0	0	1	0
dual v.6	0,25	1	0	1	1
dual v.7	0,4	0	0	0	0
dual v.8	0,375	0	0	0	0
dual v.9	0,4	0	0	1	0
dual v.10	0,4	1	0	0	2
alpha=8,862					

12. Iteration		dec. V1	dec. V2	dec. V3
optimal value=)149,79				
dual v.1	0,2	0	0	0
dual v.2	0,2	1	0	2
dual v.3	0,333	0	2	0
dual v.4	0,2	0	1	0
dual v.5	0,2	0	0	1
dual v.6	0,2	0	0	0
dual v.7	0,4	0	0	0
dual v.8	0,4	0	0	0
dual v.9	0,4	1	0	1
dual v.10	0,4	0	0	0
alpha=8,59				

12. Iteration
optimal value=)149,667
alpha=8,59

APPENDIX C

Parts of trim loss calculation table

C.1. Image parts of trim loss calculation table for width 1020 mm and 1730 mm

aij	21	25	29	43	59	61	62	63	64	67
500	0	0	0	0	0	1	2	0	0	0
450	0	0	0	0	3	0	0	0	0	0
645	0	0	0	0	0	0	0	0	0	0
430	4	0	0	0	0	1	0	0	0	0
370	0	0	0	4	1	0	0	0	0	2
495	0	0	0	0	0	0	0	0	0	0
850	0	2	0	0	0	0	0	1	0	0
750	0	0	2	0	0	0	0	0	1	0
725	0	0	0	0	0	1	1	1	1	1
720	0	0	0	0	0	0	0	0	0	0
WIDTH OF ROLL	1730	1730	1730	1730	1730	1730	1730	1730	1730	1730

# OF ROLLS	8	27	32	8	6	3	3	1	1	1

TRIM LOSS CALCULATION										
USED AREA PER PATTERN	13760	45900	48000	11840	10320	4965	5175	1575	1475	1465
TOTAL AREA PER PATTERN	13840	46710	55360	13840	10380	5190	5190	1730	1730	1730
LOSS PER PATTERN	80	810	7360	2000	60	225	15	155	255	265

C.2. Image parts of trim loss calculation table for width 1020 mm,1220 mm ,1730 mm

aij	54	61	75	76	97	105	109
500	0	0	1	0	0	0	0
450	0	0	0	0	0	0	0
645	0	0	0	0	0	0	0
430	4	0	0	0	0	0	0
370	0	0	3	4	0	0	0
495	0	0	0	0	0	0	0
850	0	1	0	0	0	0	1
750	0	1	0	0	1	0	0
725	0	0	0	0	1	2	0
720	0	0	0	0	0	0	1
WIDTH OF ROLL	1730	1730	1730	1730	1730	1730	1730

# OF ROLLS	3	33	10	2	12	34	22
-------------------	---	----	----	---	----	----	----

TRIM LOSS CALCULATION							
USED AREA PER PATTERN	5160	52800	16100	2960	17700	49300	34540
TOTAL AREA PER PATTERN	5190	57090	17300	3460	20760	58820	38060
LOSS PER PATTERN	30	4290	1200	500	3060	9520	3520

C.3. Image parts of trim loss calculation table for width 1020 mm,1220 mm,1530 mm and 1730 mm

aij	62	99	140	154	168	171	174	176
500	0	0	0	1	0	0	2	0
450	0	0	0	0	2	3	0	0
645	0	0	0	0	0	0	0	0
430	0	0	0	0	0	0	0	0
370	0	0	0	3	0	1	0	0
495	0	0	0	0	0	0	0	0
850	0	0	1	0	0	0	0	0
750	2	0	1	0	1	0	0	1
725	0	0	0	0	0	0	1	1
720	0	2	0	0	0	0	0	0
WIDTH OF ROLL	1530	1530	1730	1730	1730	1730	1730	1730

# OF ROLLS	4	1	55	8	1	6	1	1
-------------------	---	---	----	---	---	---	---	---

TRIM LOSS CALCULATION								
USED AREA PER PATTERN	6000	1440	88000	12880	1650	10320	1725	1475
TOTAL AREA PER PATTERN	6120	1530	95150	13840	1730	10380	1730	1730
LOSS PER PATTERN	120	90	7150	960	80	60	5	255

C.4. Image parts of trim loss calculation table for width 1020 mm,1220 mm, 1430 mm, 1530 mm and 1730 mm

aij	119	133	154	157	162	175	190	194
500	0	0	0	0	0	0	0	0
450	0	0	0	0	0	0	0	0
645	0	0	0	0	0	0	0	0
430	0	0	0	0	0	0	4	0
370	0	4	0	2	0	0	0	0
495	0	0	0	0	0	0	0	0
850	0	0	0	0	0	0	0	2
750	2	0	1	0	0	0	0	0
725	0	0	1	1	2	1	0	0
720	0	0	0	0	0	1	0	0
WIDTH OF ROLL	1530	1530	1530	1530	1530	1530	1730	1730

# OF ROLLS	32	8	1	1	23	1	8	27
-------------------	----	---	---	---	----	---	---	----

TRIM LOSS CALCULATION								
USED AREA PER PATTERN	48000	11840	1475	1465	33350	1445	13760	45900
TOTAL AREA PER PATTERN	48960	12240	1530	1530	35190	1530	13840	46710
LOSS PER PATTERN	960	400	55	65	1840	85	80	810

C.5. Image parts of trim loss calculation table for width 1020 mm,1220 mm,1430 mm, 1530 mm, 1730 mm and 2000 mm

aij	271	343	378	434	439	440	441	450	486	488	490
500	0	0	0	0	1	0	0	0	1	0	0
450	0	0	0	0	0	0	2	0	0	0	0
645	2	0	3	0	0	0	0	0	0	0	0
430	0	0	0	1	0	1	0	0	0	0	2
370	1	0	0	0	0	0	1	1	4	3	3
495	0	1	0	0	0	0	0	0	0	0	0
850	0	0	0	1	0	0	0	1	0	0	0
750	0	0	0	0	0	0	0	0	0	1	0
725	0	2	0	0	2	2	1	1	0	0	0
720	0	0	0	1	0	0	0	0	0	0	0
WIDTH OF ROLL	1730	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000

# OF ROLLS	1	13	16	45	8	9	10	10	2	1	3
-------------------	---	----	----	----	---	---	----	----	---	---	---

TRIM LOSS CALCULATION											
USED AREA PER PATTERN	1660	25285	30960	90000	15600	16920	19950	19450	3960	1860	5910
TOTAL AREA PER PATTERN	1730	26000	32000	90000	16000	18000	20000	20000	4000	2000	6000
LOSS PER PATTERN	70	715	1040	0	400	1080	50	550	40	140	90

C.6. Image parts of trim loss calculation table for width 1020 mm,1220 mm,1430 mm, 1530 mm, 1730 mm , 1830 mm and 2000 mm

ajj	270	488	520	576	581	582	583	592
500	0	0	0	0	1	0	0	0
450	0	0	0	0	0	0	2	0
645	2	0	3	0	0	0	0	0
430	1	0	0	1	0	1	0	0
370	0	0	0	0	0	0	1	1
495	0	1	0	0	0	0	0	0
850	0	0	0	1	0	0	0	1
750	0	1	0	0	0	0	0	0
725	0	1	0	0	2	2	1	1
720	0	0	0	1	0	0	0	0
WIDTH OF ROLL	1730	2000	2000	2000	2000	2000	2000	2000
# OF ROLLS	1	45	16	45	6	2	10	9
TRIM LOSS CALCULATION								
USED AREA PER PATTERN	1720	88650	30960	90000	11700	3760	19950	17505
TOTAL AREA PER PATTERN	1730	90000	32000	90000	12000	4000	20000	18000
LOSS PER PATTERN	10	1350	1040	0	300	240	50	495

C.7. Image parts of trim loss calculation table for width 1020 mm,1220 mm,1430 mm, 1530 mm, 1730 mm , 1830 mm, 1930 mm and 2000 mm

aij	270	664	690	699	755	760	761	762
500	0	0	0	0	0	1	0	0
450	0	0	0	0	0	0	0	2
645	2	0	0	3	0	0	0	0
430	1	0	0	0	1	0	1	0
370	0	0	0	0	0	0	0	1
495	0	1	1	0	0	0	0	0
850	0	0	0	0	1	0	0	0
750	0	0	2	0	0	0	0	0
725	0	2	0	0	0	2	2	1
720	0	0	0	0	1	0	0	0
WIDTH OF ROLL	1730	2000	2000	2000	2000	2000	2000	2000

# OF ROLLS	1	13	32	16	45	7	10	10
-------------------	---	----	----	----	----	---	----	----

TRIM LOSS CALCULATION								
USED AREA PER PATTERN	1720	25285	63840	30960	90000	13650	18800	19950
TOTAL AREA PER PATTERN	1730	26000	64000	32000	90000	14000	20000	20000
LOSS PER PATTERN	10	715	160	1040	0	350	1200	50

C.8. Image parts of trim loss calculation table for width 1020 mm,1220 mm,1430 mm, 1530 mm, 1630 mm, 1730 mm , 1830 mm, 1930 mm and 2000 mm

aij	438	688	753	779	788	844	849	850	851
500	0	0	0	0	0	0	1	0	0
450	0	0	0	0	0	0	0	0	2
645	2	0	0	0	3	0	0	0	0
430	0	0	0	0	0	1	0	1	0
370	1	3	0	0	0	0	0	0	1
495	0	0	1	1	0	0	0	0	0
850	0	0	0	0	0	1	0	0	0
750	0	1	0	2	0	0	0	0	0
725	0	0	2	0	0	0	2	2	1
720	0	0	0	0	0	1	0	0	0
WIDTH OF ROLL	1830	1930	2000	2000	2000	2000	2000	2000	2000

# OF ROLLS	1	1	13	32	16	45	10	8	10
-------------------	---	---	----	----	----	----	----	---	----

TRIM LOSS CALCULATION									
USED AREA PER PATTERN	1660	1860	25285	63840	30960	90000	19500	15040	19950
TOTAL AREA PER PATTERN	1830	1930	26000	64000	32000	90000	20000	16000	20000
LOSS PER PATTERN	170	70	715	160	1040	0	500	960	50

C.9. Image parts of trim loss calculation table for width 1020 mm,1120 mm,1220 mm,1430 mm,1530 mm,1630 mm, 1730 mm,1830 mm,1930 mm and 2000 mm

ajj	426	725	775	801	810	866	871	873
500	0	0	0	0	0	0	1	0
450	0	0	0	0	0	0	0	2
645	2	0	0	0	3	0	0	0
430	0	1	0	0	0	1	0	0
370	0	0	0	0	0	0	0	1
495	1	0	1	1	0	0	0	0
850	0	0	0	0	0	1	0	0
750	0	2	0	2	0	0	0	0
725	0	0	2	0	0	0	2	1
720	0	0	0	0	0	1	0	0
WIDTH OF ROLL	1830	1930	2000	2000	2000	2000	2000	2000
# OF ROLLS	1	11	23	21	16	45	7	10
TRIM LOSS CALCULATION								
USED AREA PER PATTERN	1785	21230	44735	41895	30960	90000	13650	19950
TOTAL AREA PER PATTERN	1830	21230	46000	42000	32000	90000	14000	20000
LOSS PER PATTERN	45	0	1265	105	1040	0	350	50

C.10. Image parts of trim loss calculation table for width 1020, 1120, 1220, 1320, 1430, 1530, 1630, 1730, 1830, 1930 and 2000mm

aij	705	816	842	851	907	912	914
500	0	0	0	0	0	1	0
450	1	0	0	0	0	0	2
645	0	0	0	3	0	0	0
430	0	0	0	0	1	0	0
370	0	0	0	0	0	0	1
495	0	1	1	0	0	0	0
850	0	0	0	0	1	0	0
750	0	0	2	0	0	0	0
725	2	2	0	0	0	2	1
720	0	0	0	0	1	0	0
WIDTH OF ROLL	1930	2000	2000	2000	2000	2000	2000
# OF ROLLS	11	12,5	32,5	16,6667	45	9,25	4,5
TRIM LOSS CALCULATION							
USED AREA PER PATTERN	20900	24313	64838	32250	90000	18038	8978
TOTAL AREA PER PATTERN	21230	25000	65000	33333,3	90000	18500	9000
LOSS PER PATTERN	330	687,5	162,5	1083,33	0	462,5	22,5

APPENDIX D

D.1. Some images of CPLEX for the eleven different widths

Code

```

range pattern=1..110;

dvar float+ x[pattern];

minimize sum(i in pattern)x[i];

subject to
ct1:2*x[1]+2*x[11]+2*x[21]+2*x[31]+2*x[41]+3*x[51]+3*x[61]+3*x[71]+3*x[81]+3*x[91]+4*x[101]>=10;
ct2:2*x[2]+2*x[12]+2*x[22]+2*x[32]+3*x[42]+3*x[52]+3*x[62]+3*x[72]+4*x[82]+4*x[92]+4*x[102]>=20;
ct3:x[3]+x[13]+x[23]+2*x[33]+2*x[43]+2*x[53]+2*x[63]+2*x[73]+2*x[83]+2*x[93]+3*x[103]>=50;
ct4:2*x[4]+2*x[14]+2*x[24]+3*x[34]+3*x[44]+3*x[54]+3*x[64]+4*x[74]+4*x[84]+4*x[94]+4*x[104]>=60;
ct5:2*x[5]+3*x[15]+3*x[25]+3*x[35]+3*x[45]+4*x[55]+4*x[65]+4*x[75]+4*x[85]+5*x[95]+5*x[105]>=40;
ct6:2*x[6]+2*x[16]+2*x[26]+2*x[36]+2*x[46]+3*x[56]+3*x[66]+3*x[76]+3*x[86]+3*x[96]+4*x[106]>=45;
ct7:x[7]+x[17]+x[27]+x[37]+x[47]+x[57]+x[67]+2*x[77]+2*x[87]+2*x[97]+2*x[107]>=55;
ct8:x[8]+x[18]+x[28]+x[38]+x[48]+2*x[58]+2*x[68]+2*x[78]+2*x[88]+2*x[98]+2*x[108]>=65;
ct9:x[9]+x[19]+x[29]+x[39]+x[49]+x[59]+2*x[69]+2*x[79]+2*x[89]+2*x[99]+2*x[109]>=80;
ct10:x[10]+x[20]+x[30]+x[40]+x[50]+2*x[60]+2*x[70]+2*x[80]+2*x[90]+2*x[100]+2*x[110]>=45;
}

execute post{
var duality1=0;
var duality2=0;
var duality3;
var duality4;
var duality5;
var duality6;
var duality7;
var duality8;
var duality9;
}

```

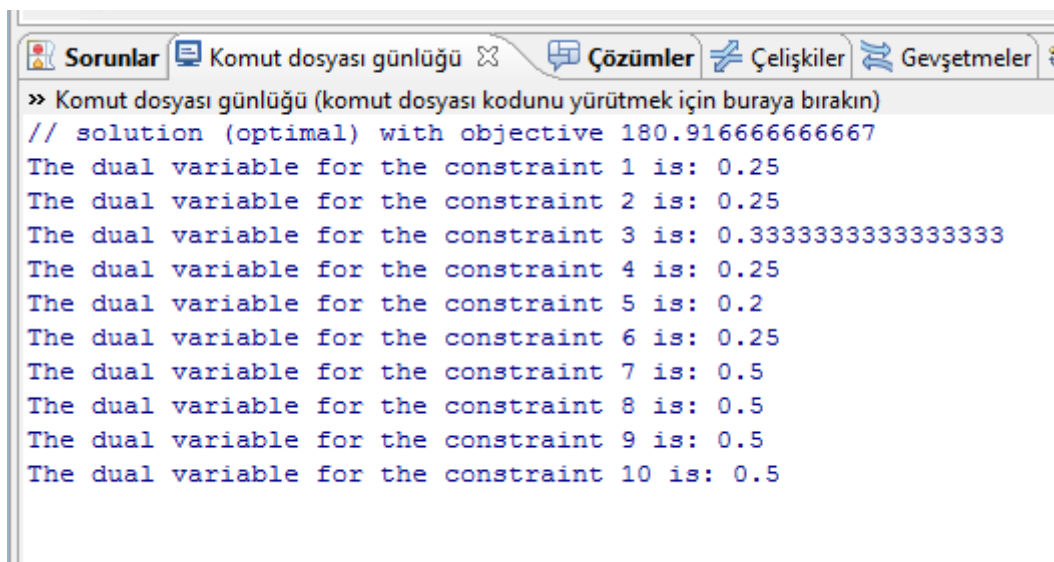
Code

```
var duality10;

duality1=ct1.dual;
duality2=ct2.dual;
duality3=ct3.dual;
duality4=ct4.dual;
duality5=ct5.dual;
duality6=ct6.dual;
duality7=ct7.dual;
duality8=ct8.dual;
duality9=ct9.dual;
duality10=ct10.dual;

writeln("The dual variable for the constraint 1 is: ",duality1);
writeln("The dual variable for the constraint 2 is: ",duality2);
writeln("The dual variable for the constraint 3 is: ",duality3);
writeln("The dual variable for the constraint 4 is: ",duality4);
writeln("The dual variable for the constraint 5 is: ",duality5);
writeln("The dual variable for the constraint 6 is: ",duality6);
writeln("The dual variable for the constraint 7 is: ",duality7);
writeln("The dual variable for the constraint 8 is: ",duality8);
writeln("The dual variable for the constraint 9 is: ",duality9);
writeln("The dual variable for the constraint 10 is: ",duality10);
}
```

Code



The screenshot shows a software interface with a menu bar containing 'Sorunlar', 'Komut dosyası günlüğü', 'Çözümler', 'Çelişkiler', and 'Gevşetmeler'. The 'Komut dosyası günlüğü' window is active, displaying the following output:

```
>> Komut dosyası günlüğü (komut dosyası kodunu yürütmek için buraya bırakın)
// solution (optimal) with objective 180.916666666667
The dual variable for the constraint 1 is: 0.25
The dual variable for the constraint 2 is: 0.25
The dual variable for the constraint 3 is: 0.333333333333333333
The dual variable for the constraint 4 is: 0.25
The dual variable for the constraint 5 is: 0.2
The dual variable for the constraint 6 is: 0.25
The dual variable for the constraint 7 is: 0.5
The dual variable for the constraint 8 is: 0.5
The dual variable for the constraint 9 is: 0.5
The dual variable for the constraint 10 is: 0.5
```

Code

```

range vars=1..10;
range width=1..11;
float cof[vars]=[0.25,0.25,0.33,0.25,0.2,0.25,0.5,0.5,0.5,0.5];
dvar int+ u[width,vars];

maximize sum(i in vars,j in width)cof[i]*u[j,i];
subject to
500*u[1,1]+450*u[1,2]+645*u[1,3]+430*u[1,4]+370*u[1,5]+495*u[1,6]+850*u[1,7]+750*u[1,8]+725*u[1,9]+
500*u[2,1]+450*u[2,2]+645*u[2,3]+430*u[2,4]+370*u[2,5]+495*u[2,6]+850*u[2,7]+750*u[2,8]+725*u[2,9]+
500*u[3,1]+450*u[3,2]+645*u[3,3]+430*u[3,4]+370*u[3,5]+495*u[3,6]+850*u[3,7]+750*u[3,8]+725*u[3,9]+
500*u[4,1]+450*u[4,2]+645*u[4,3]+430*u[4,4]+370*u[4,5]+495*u[4,6]+850*u[4,7]+750*u[4,8]+725*u[4,9]+
500*u[5,1]+450*u[5,2]+645*u[5,3]+430*u[5,4]+370*u[5,5]+495*u[5,6]+850*u[5,7]+750*u[5,8]+725*u[5,9]+
500*u[6,1]+450*u[6,2]+645*u[6,3]+430*u[6,4]+370*u[6,5]+495*u[6,6]+850*u[6,7]+750*u[6,8]+725*u[6,9]+
500*u[7,1]+450*u[7,2]+645*u[7,3]+430*u[7,4]+370*u[7,5]+495*u[7,6]+850*u[7,7]+750*u[7,8]+725*u[7,9]+
500*u[8,1]+450*u[8,2]+645*u[8,3]+430*u[8,4]+370*u[8,5]+495*u[8,6]+850*u[8,7]+750*u[8,8]+725*u[8,9]+
500*u[9,1]+450*u[9,2]+645*u[9,3]+430*u[9,4]+370*u[9,5]+495*u[9,6]+850*u[9,7]+750*u[9,8]+725*u[9,9]+
500*u[10,1]+450*u[10,2]+645*u[10,3]+430*u[10,4]+370*u[10,5]+495*u[10,6]+850*u[10,7]+750*u[10,8]+725
500*u[11,1]+450*u[11,2]+645*u[11,3]+430*u[11,4]+370*u[11,5]+495*u[11,6]+850*u[11,7]+750*u[11,8]+725
}

1,3]+430*u[1,4]+370*u[1,5]+495*u[1,6]+850*u[1,7]+750*u[1,8]+725*u[1,9]+720*u[1,10]<=1020;
2,3]+430*u[2,4]+370*u[2,5]+495*u[2,6]+850*u[2,7]+750*u[2,8]+725*u[2,9]+720*u[2,10]<=1120;
3,3]+430*u[3,4]+370*u[3,5]+495*u[3,6]+850*u[3,7]+750*u[3,8]+725*u[3,9]+720*u[3,10]<=1220;
4,3]+430*u[4,4]+370*u[4,5]+495*u[4,6]+850*u[4,7]+750*u[4,8]+725*u[4,9]+720*u[4,10]<=1320;
5,3]+430*u[5,4]+370*u[5,5]+495*u[5,6]+850*u[5,7]+750*u[5,8]+725*u[5,9]+720*u[5,10]<=1430;
6,3]+430*u[6,4]+370*u[6,5]+495*u[6,6]+850*u[6,7]+750*u[6,8]+725*u[6,9]+720*u[6,10]<=1530;
7,3]+430*u[7,4]+370*u[7,5]+495*u[7,6]+850*u[7,7]+750*u[7,8]+725*u[7,9]+720*u[7,10]<=1630;
8,3]+430*u[8,4]+370*u[8,5]+495*u[8,6]+850*u[8,7]+750*u[8,8]+725*u[8,9]+720*u[8,10]<=1730;
9,3]+430*u[9,4]+370*u[9,5]+495*u[9,6]+850*u[9,7]+750*u[9,8]+725*u[9,9]+720*u[9,10]<=1830;
i[10,3]+430*u[10,4]+370*u[10,5]+495*u[10,6]+850*u[10,7]+750*u[10,8]+725*u[10,9]+720*u[10,10]<=1930;
i[11,3]+430*u[11,4]+370*u[11,5]+495*u[11,6]+850*u[11,7]+750*u[11,8]+725*u[11,9]+720*u[11,10]<=2000;

```

Code

```

'/ solution (optimal) with objective 10.26
'/ Quality Incumbent solution:
'/ MILP objective                                1.0260000000e+001
'/ MILP solution norm |x| (Total, Max)          2.50000e+001 2.00000e+000
'/ MILP solution error (Ax=b) (Total, Max)      0.00000e+000 0.00000e+000
'/ MILP x bound error (Total, Max)              0.00000e+000 0.00000e+000
'/ MILP x integrality error (Total, Max)        0.00000e+000 0.00000e+000
'/ MILP slack bound error (Total, Max)          0.00000e+000 0.00000e+000
'/

i = [[0
      0 1 0 1 0 0 0 0 0]
      [0 0 0 0 1 0 0 1 0 0]
      [1 0 0 0 0 0 0 0 0 1]
      [0 0 0 1 0 0 0 1 0 0]
      [0 0 1 0 0 0 0 0 0 1]
      [0 0 0 0 0 0 0 0 0 2]
      ...

```

width (büyükük11)	vars (büyükük10)	Değer
1	4	0
1	5	1
1	6	0
1	7	0
1	8	0
1	9	0
1	10	0
2	1	0
2	2	0
2	3	0
2	4	0
2	5	1
2	6	0
2	7	0
2	8	1
2	9	0
2	10	0
3	1	1
3	2	0
3	3	0
3	4	0

width (büyükük11)	vars (büyükük10)	Değer
4	1	0
4	2	0
4	3	0
4	4	1
4	5	0
4	6	0
4	7	0
4	8	1
4	9	0
4	10	0
5	1	0
5	2	0
5	3	1
5	4	0
5	5	0
5	6	0
5	7	0
5	8	0
5	9	0
5	10	1
6	1	0
6	2	0