TWO DIMENSIONAL CUTTING STOCK PROBLEM WITH MULTIPLE STOCK SIZES

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

BY

UMUTCAN AYASANDIR

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

SEPTEMBER 2019

Approval of the thesis:

TWO DIMENSIONAL CUTTING STOCK PROBLEM WITH MULTIPLE STOCK SIZES

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

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

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

Prof. Dr. Meral Azizoğlu
Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Assist. Prof. Dr. Özgen Karaer
Industrial Engineering, METU _____

Prof. Dr. Meral Azizoğlu
Industrial Engineering, METU _____

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

Assist. Prof. Dr. Melih Çelik
Industrial Engineering, METU _____

Assist. Prof. Dr. Özlem Karsu
Industrial Engineering, Bilkent University _____

Date: 06.09.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Umutcan Ayasandır

Signature:

ABSTRACT

TWO DIMENSIONAL CUTTING STOCK PROBLEM WITH MULTIPLE STOCK SIZES

Ayasandır, Umutcan
Master of Science, Industrial Engineering
Supervisor: Prof. Dr. Meral Azizoğlu

September 2019, 94 pages

In this study, we consider a two dimensional cutting stock problem with multiple stock sizes and two stage guillotine cuts. Our objective is to maximize the difference between total revenue over all items and total cost over all used panels.

We propose two mathematical models and discuss their relative performances. We enhance the performances of the models by incorporating the properties of optimal solution that we derive.

The results of our computational study have revealed the satisfactory performance of one of our models with optimal properties for medium sized problem instances. We develop decomposition-based heuristics that produce high quality solutions in reasonable time.

Keywords: Cutting Stock Problem, Mathematical Models, Decomposition Methods

ÖZ

İKİ BOYUTLU STOK KESME PROBLEMİ

Ayasandır, Umutcan
Yüksek Lisans, Endüstri Mühendisliği
Tez Danışmanı: Prof. Dr. Meral Azizoğlu

Eylül 2019, 94 sayfa

Bu çalışmada, iki boyutlu çoklu stok büyüklüklerinin iki aşamalı giyotin kesim problemi ele alınmıştır. Amacımız, kesilen ürünlerden elde edilen toplam gelir ile kullanılan panellerdeki toplam maliyet arasındaki farkı ençoklamak olarak belirlenmiştir.

Problemin çözümü için iki matematiksel model önerilmiş ve performansları değerlendirilmiştir. Her iki modelin performansı da elde edilen en iyi çözüm özelliklerini kullanarak arttırılmıştır. Literatürden alınan örnek problemler üzerinde yapılan sayısal çalışmalar, modellerimizden birinin orta büyüklükte problem boyutlarında tatmin edici sonuçlar verdiğini ortaya koymuştur. Ayrıştırma tabanlı sezgisel algoritmalar geliştirilerek makul sürede yüksek kaliteli çözümler elde edilmiştir.

Anahtar Kelimeler: Stok Kesme Problemi, Matematiksel Modeller, Ayrıştırma Algoritmaları

To my family

ACKNOWLEDGEMENTS

Above all, I would like to thank my supervisor Prof. Meral Azizoglu. Her kind, helpful and professional attitude makes all things easier. She motivated me whenever I need help and support. The deep knowledge she has on her area of expertise makes her approach the problems in confident and decisive manner, that I impressed a lot. I learned from her in every phase and aspect of the study, so I feel very lucky to have the privilege to work with her.

I would like to thank Prof. Murat Köksalan for his contributions and support throughout the study and providing a pleasant working environment. I am grateful to Erdoğan Akbulak who brought an industrial problem to academy. He gave me an opportunity to study on a real-world problem by establishing, funding and supporting the research project. I would also like to express my sincere appreciation to Levent Parıldar and all Silkar Marble Incorporation employees who put effort to this project.

I cannot express my gratitude enough to my family. My dear father Burhan Ayasandır is always there to support me. I never felt alone and helpless because he always tries to do the best for me. My dear mother Nezihe Ayasandır makes all things beautiful with her existence, she is my friend, teacher and supporter who I love so much. I am very lucky to have my thoughtful and caring sister Ecem Arığ. My life would be difficult without her sincere love, support, and deep friendship. I would like to express my deepest feelings of love to the one who brought beauty to my life. I am thankful and blessed to have my beloved wife Feyza Ayasandır. I feel treasured to meet her, to love her and to get married with her.

I would like to thank my friend Ali Kaan Sungur to help me with his knowledge on computer sciences. I am grateful to Nail Karabay for being a very good roommate and friend. I appreciate Mustafa Muhammet Öztürk who put effort in motivating me to study on my thesis. Ersin Telemeci was my companion in the journey of this thesis and I am grateful for his very good friendship and support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Cutting stock problems obtain a set of small items from a set of large items with defined geometric dimensions of the items. The items may have one, two, three or more than three dimensions. In this study, we consider two-dimensional cutting stock problem that cuts a set of rectangular items from a set of rectangular stocks that are available in multiple dimensions.

Two basic types of cutting stock problems are studied according to the assignment type that defines the objective. The types are referred to as output maximization or input minimization. Output maximization problems assign a set of items with specified demands to a set of large items that may not be sufficient to cover all demand. So the problem is to obtain maximum number (or weighted number) of small items from the available large items. Input minimization problems, on the other hand, decides on the minimum number (or weighted) of large items so as to cover all demand. Minimization of panel costs is an input minimizing, maximization of total revenue on the other hand is an output maximizing concern.

In the literature the studies are either input minimization or output maximization type. Cutting stock problems and bin-packing problems are famous examples of input minimization type. Knapsack problems are different from both, because the total weight of the items placed on large objects is maximized. In this study, we consider the net revenue, i.e. profit, problem that maximizes the difference between the total revenue brought by all cut items and the total cost incurred by all used panels. Hence our problem has both input maximization and output minimization concerns.

One can use number of small items to categorize the literature in cutting problems. In bin packing problems, all item types are packed only once while in cutting stock

problems several items of each type are cut. In other words, bin-packing problem is a special case of cutting stock problem where demand of each item type is one. In this study, number of small items of each type is arbitrary, but not limited to one.

The small items may have rigid orientation such that their widths and their lengths should fit those of the large objects. Some small items may be rotated so that they may be placed with 90 degree change. Also, there may be some restrictions in cutting process coming from industrial applications, such as guillotine cutting constraints. In this study, we basically consider rigid, i.e., non rotatable items in guillotine cutting environment and discuss the extensions to the rotatable items case.

Two-dimensional cutting stock problem is a strongly NP-hard problem (Macedo et. al., 2008) and has many practical application areas. The application areas reported in the literature include but are not limited to the paper industry, wood industry and glass industry.

Our motivating example is a marble industry where stock large objects are available in limited number, different dimensions and quality that affect their costs. They do have many customers asking many different rectangular items for different prices and some of their final products may be composed of small cut items in the company. Each item has a demand above which production is forbidden and each produced, i.e. cut item, brings a defined revenue which might also convey information about the prestige of the customer in company's view. The revenues of the small items depend on the customer orders, delivery times and dimensions of the items. For some specified time period they want to maximize total profit without exceeding panel availabilities and customer demand. Due to the fact that the dimensions of the items cut depend on the special request of the customer, it is reasonable to limit number of items cut by the customer demand.

The journey of this study originates from this practical problem. We develop two mathematical models, one of which is extended from the literature.

We enhance the efficiency of the models by recognizing the properties of optimal solution.

The results of our computational study have revealed that our mathematical model can handle only medium sized problems with not so high demand quantities. To handle larger sized problem instances we develop several heuristic procedures and observe their satisfactory performance in terms of both speed and closeness to the optimal solutions.

The rest of the thesis is organized as follows. In Chapter 2, we give the review of the studies in the literature. In Chapter 3, we define our problem and discuss the alternative solution approaches. We give the results of our numerical study for each of the solution approaches in Chapter 4. Chapter 5 discusses the details of our application to the marble company. Chapter 6 concludes the study by pointing out main findings and discussing the possible research directions.

CHAPTER 2

LITERATURE REVIEW

2.1. Literature on Cutting and Packing Problems

Cutting stock problem (CSP) is first introduced by Gilmore and Gomory (1961). The authors first dealt with one-dimensional cutting stock problem and then extended the problem for two and more dimensions (Gilmore and Gomory, 1965). Since then, different variants and extensions of the CSP are handled in the literature. To classify the researches and publications properly, Dyckhoff (1990) introduced a typology for cutting and packing problems.

Cutting problems basically refer to the problems where small items are obtained by cutting large objects, like in CSP. Packing problems refer to the problems where small items are assigned and placed large objects, like bin packing (BPP) and knapsack problems. The difference mainly in the naming, coming from the aim of the problem but the solution approaches and the variants of the problems are similar.

Dyckhoff (1990) categorized the problem with respect to dimensionality, kind of assignment, assortment of large objects and assortment of small items. The number of publications related with the problem has increased after Dyckhoff's study, that can be attributed the interest it raised. Wascher et. al. (2007) proposed an improved typology that uses Dyckhoff's typology as a baseline to introduce their ideas and modifications. They also proposed new names for the criteria and problem types. Wascher et. al. (2007) divided the problem types into three as basic, intermediate and refined problem types. Kind of assignment and assortment of small items criteria define the basic problem types while the assortment of large objects criterion defines the intermediate problem types. Finally, refined problem types are defined with respect to the number of dimensions.

In this section, we concentrate on a narrower area of the literature, two dimensional cutting and packing problems. One may refer to the review papers for the classification of one-dimensional problems. Coffman et. al. (2013) classified the literature on approximation algorithms for bin packing problem. In terms of exact methods, Valerio and Carvalho (2002) compared the LP model formulations for one-dimensional BPP and CSP. Martinovic et. al. (2018) reviewed and compared the available modelling approaches in the literature for one-dimensional cutting stock problem. Also, Delorme et. al.'s (2016) paper gives the review of important mathematical formulations and exact solution methods in the last fifty years.

2.2. Literature on Two Dimensional Cutting and Packing Problem

In this section, we first review the related studies on the exact methods and then the non-exact ones.

2.2.1. Exact Methods

Fekete et. al. (2007) dealt with the orthogonal packing problem. In the problem, there are rectangular small items and one large rectangular object. The aim is to place small items into the large object. They used graph theoretical characterizations of assignments of items to the objects. By using these characterizations, the feasible assignments are evaluated as a group, not one by one, that shares same characteristics. A successful branch and bound algorithm is implemented thanks to the good bounds obtained.

Clautiaux et. al. (2008) proposed a new constraint based scheduling model to the same problem. They used the model in branch and bound scheme together with the new techniques to improve the solution approach. The approach generally outperforms the methods in the literature but for some instances Fekete et. al.'s (2007) method performs better.

Alvares-Valdez et. al. (2008) studied "strip packing problem", where the width of large object is fixed but the length of each strip is infinite. The objective is to minimize

the length of the large object while assigning the small items to this object. The authors proposed branch and bound algorithm for the problem. In branching scheme, they used bounds obtained from the LP relaxation of integer formulation model. Also, they reduced the branching tree by exploiting some dominance relations and bounds obtained by geometric considerations. The solution approach is effective in general, but there are some difficulties to solve the problem instances with small items.

Boschetti and Montaletti (2010) followed similar approach and proposed new lower and upper bounds to use in branch and bound algorithms. The proposed upper bounds are found by constructive heuristics and the lower bounds are found by relaxations of different mathematical formulations of the problem. The authors also proposed some reduction techniques. Their solutions are better in most of the instances.

Martello and Vigo (1998) proposed branch and bound algorithm to solve two-dimensional bin packing problem (2DBPP). Unlike the strip packing and orthogonal packing problems, there are generally infinite number of large objects in bin packing problems and aim is to minimize the number of bins used. For the branch and bound algorithm, they compute bounds by using the dimensions of the items, and one-dimensional version of the problem. At the outer branching tree, the item is assigned to a large object (bin) and a feasible assignment is tried to be found. At inner branching tree, the existence of feasible packing is questioned. The study was the first attempt to solve the bin packing problem by an exact method. The authors were able to find a solution to the problems with -up to- 120 items, by this method.

Lodi et. al. (2004) proposed a formulation with polynomial number of variables and constraints for bin packing problem. They proposed the model for the problems where the items have to be packed by levels. This restriction is motivated by industry capabilities, such as guillotine cutting. According to the authors, by using the LP relaxation objective of the proposed model in standard branch and bound package, 414 of 500 instances are solved to optimality within 5 minutes. Also, the model itself can be solved by a commercial MIP solver.

Pisinger and Sigurd (2007) used Dantzig-Wolfe decomposition based column generation method to solve the 2DBPP. The subproblem in the decomposition scheme is responsible for the assignment of items to the single large object. The solution approach is effective because the infeasibilities in subproblem returns back to the master problem as valid inequalities. Lower bounds computed by delayed column generation are superior or the same as the existing bounds in the literature. Optimal solutions can be obtained for the problems with –up to- 100 items. Guillotine cutting constraints can easily be accommodated in subproblem, so they obtained solutions for 2DBPP with guillotine constraints.

Macedo et. al. (2010) formulated the two dimensional problem as a minimum flow problem extending its one dimensional version. In the formulation, each cutting pattern corresponds to a path and positions in the large objects correspond to vertices. Model is compared the other models in the literature in terms of linear relaxations and proved to be stronger. The authors solved the model by commercial MIP solver in their solution approach. As the model has a pseudo-polynomial number of constraints and variables, the solutions are obtained in reasonable times.

Silva et. al. (2010) proposed an integer-programming model that uses possible residual large objects as parameters. The model performs well for the case when the dimensions of the items are not much smaller than the dimensions of the panels. Small items create more residual panels so the size of the model is increased. However, the model size is not affected by the demand of the items. Therefore the model is capable to solve many real world instances in reasonable times. Only in 29 of 672 instances of 4 real world problem, the model could not reach the optimal solution in two hours.

Pisinger and Sigurd (2005) studied 2DBPP with variable bin sizes (2DVSBPP) and proposed the first exact algorithm. They used Dantzig-Wolfe decomposition for the integer-linear formulation of the problem to find lower bounds for their branch and price algorithm. However, they showed that the algorithm could handle only small sized instances.

Furini and Malaguti (2013) studied the cutting stock problem with multiple stock size. They extended the Lodi et. al.'s (2004) model to multiple stock size case and duplicated the items with respect to their demands. They used some modelling techniques in order to reduce symmetries coming with the duplication. They also extend Silva et al.'s (2010) model to the multiple stock size case. Finally, they included different sized large objects in Gilmore and Gomory's exponential size model and solved the model by using branch and price techniques. Comparison of those three models suggests that the superiority of one model over any other is instant dependent. Models proposed are all suitable for the industries where guillotine-cutting restrictions are applied.

Guillotine cutting restrictions are available in some other studies as well. Dolatabadi et. al. (2012) studied two dimensional guillotine knapsack problem where only one type of large object is available. They implemented recursive algorithm that constructs maximum guillotine assignment to a large object. Bekrar et. al. (2010) included guillotine constraints for strip packing problem. Amossen and Pisinger (2010) studied multi-dimensional guillotine bin packing problem. They proposed a constructive algorithm for guillotine and non-guillotine cutting based on constraint programming. Fleszar (2016) studied stage unrestricted guillotine cutting problem for one large stock and multiple items. The item rotations are also considered in the study.

2.2.2. Non-Exact Methods

There are plenty of heuristic approaches for cutting and packing problems. We only survey the studies on the two dimensional cutting and packing problems with multi sized large objects, stocks or bins. In terms of non-exact solution approaches to those, Riehme et. al. (1996) studied the problem for extremely varying demands between items. In such case, they proposed to decompose the problem into two by first solving the problem for high demand items and then solving the residual problem. Their "stripe approach" is found advantageous compared to the other known methods in those years.

Alvares-Valdez et. al. (2002) developed and compared the heuristic methods to solve two-dimensional cutting stock problems. They analyzed column generation based heuristic algorithm, where the subproblem is solved by dynamic programming, constructive algorithm, tabu search algorithm or GRASP algorithm. They concluded that the dynamic programming gives better results than the local search heuristics.

Cintra et. al. (2008) studied the problem both for different bin sizes and single bin size. The proposed solution approach is based on column generation and columns are generated by a dynamic programming based algorithm. Optimal or quasi optimal solutions are obtained for the test instances in reasonable times.

Furini et. al. (2012) studied two-dimensional two-staged guillotine cutting problem. They proposed column generation based heuristic algorithm. The subproblem of the column generation algorithm is solved by dynamic programming based heuristic algorithm. They did computational experiments for both item rotations are allowed and not allowed cases. It is asserted that the proposed algorithm is superior than the algorithms available in the literature.

Alvarez Valdes et. al. (2013) developed GRASP/Path Relinking algorithm for the problem. The study is one of the first papers that implements path relinking to cutting and packing problems. Computational experiments revealed the algorithm produces promising results. The algorithm could easily be modified for the rotated counterparts of items.

Wei et. al. (2013) proposed a goal driven heuristic approach and suggested multiple binary search on objective value. For a given interval, the algorithm packs the items by tabu search algorithm. Then some post improvement procedure is implemented on the solution found. The performance of goal driven approach outperforms all existing heuristic algorithms.

Hong et. al. (2014) studied two-dimensional variable sized bin packing problem with guillotine constraint. Rotations of items can optionally be included in the problem.

The authors proposed hybrid meta heuristic algorithm. Their algorithm outperforms the existing algorithms.

The most closely related study to ours is that of Furini and Malaguti (2013). They consider two-dimensional two stage cutting stock problem with guillotine cuts like ours. The differences are due to objective function used and the approaches used to find implementable solutions.

CHAPTER 3

PROBLEM DEFINITION & SOLUTION APPROACHES

3.1. Problem Statement

Two dimensional cutting stock problem (2DCSP) assigns a set of small rectangular items to a set of larger rectangular stocks, so called panels. In this study, we consider a 2DCSP with the following assumptions:

Panels are cut parallel to the sides of the stock, and cross the stock from one side to another, i.e. guillotine cuts are used. Two stage cuts are used in such a way that the rectangular bars obtained through the guillotine cuts are further cut in parallel to obtain the exact shape.

The following figure depicts the difference between two stage guillotine cuts and non guillotine cuts.

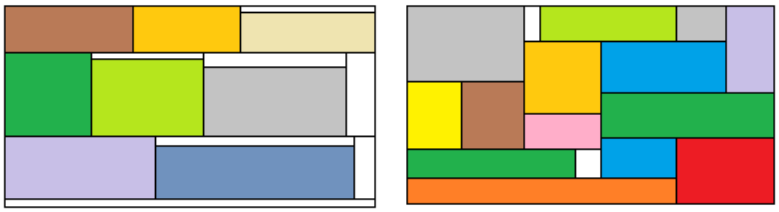


Figure 3.1 2-stage guillotine cuts and non guillotine cut (Non guillotine cuts on the right)

There are m panel types. Panel type h is characterized by its cost C_h and available number num_h.

There are n types of items. Item type j is characterized by its revenue r_j and requirement d_j.

The dimensions of panel h are specified by the length L_h and width W_h. Those of item j are characterized as l_j and w_j.

All parameters are known with certainty and are not subject to any change, i.e., the system is deterministic and static.

Our aim is to assign the items to the panels so as to maximize the total profit. We define the profit as the difference between the total revenue via all items and total cost incurred by used panels.

The problem is constrained version of the two-dimensional rectangular multiple stock size cutting stock problem according to the typology by Wascher et. al (2007) and is strongly NP-hard (see Furini and Malaguti, 2013).

3.2. Solution Approaches

We studied the problem from a marble industry in Turkey where cutting decisions are made based on the customer orders. The input of the cutting process, panels, is also provided according to a plan, based on those customer orders. Panels are obtained from three dimensional huge stone blocks and the process takes considerable time. The delivery times of the customer orders on the other hand, are not that long. Hence, one should deal with the orders particularly using available resources of the company. In our study, we assume that the number of panels of each type is limited and the demand figures are upper bounds on the amount of items cut. Limiting the number of cuts by demand is reasonable assumption for the industry because there is no guarantee that the items with same dimensions would be ordered in the near future. Also, by cutting the items above their demand, the decision maker loses the opportunity to use the available resources for the future demand of different items.

In the company, the available number of panels may not be sufficient to satisfy all demand. Accordingly, a promising solution would favor more profitable items and

more economical panel types without exceeding the item demands and panel availabilities.

Furini and Malaguti (2013) studied a very similar problem for the minimum total panel cost objective. They assume that the availability of panel types is unlimited, hence all demand can be satisfied with unlimited panel purchases. The demand is given as a lower bound and higher values of production above demand would be justified.

We first modified Furini and Malaguti's (2013) model to handle maximum profit objective, demand lower limits and availability upper limits. We refer the resulting model as Model 1.

3.2.1. Model 1

Furini and Malaguti's (2013) model is an extension of a formulation proposed by Lodi et. al. (2004) for two-dimensional bin packing problem. The two-stage guillotine cutting restriction divides the panel into levels. The basic idea behind this model is to partition the assignments of items to panels into two, namely the ones that initialize each level and the ones that assigned to levels, excluding the first items of levels. Levels are the strips that are obtained after the first horizontal guillotine cut on the panel. Figure 3.2 depicts the levels in 2D guillotine cutting example.

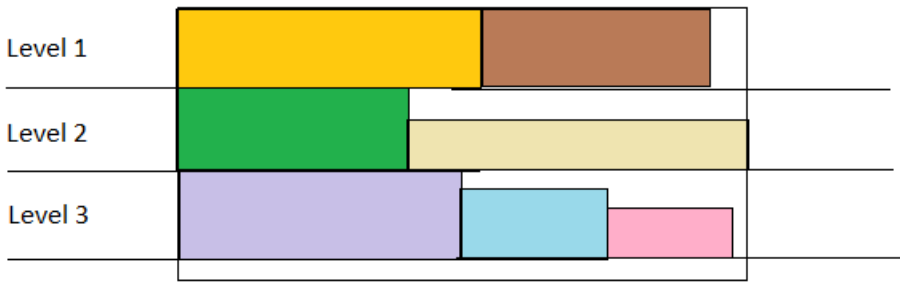


Figure 3.2 Representation of Levels in 2D Guillotine Cutting

The decision variables for the assignment of additional items (the ones other than the first item) to a level are defined only for the smaller or equal width items than the first item. The very first necessity of the approach is the ordered set of item types. The item

types are sorted with respect to their non-increasing order of widths. For each item, a possible level is defined and for each level, decision variables are defined for possible assignments. Therefore, possible levels/strips are defined as many as total demand. To construct the relation between the panels and levels in formulation, each level assigned to a panel would be assigned as the "name" of the panel used. By this way, the item number in the left upper corner would be the name of the panel used.

We use the following notation as in Furini and Malaguti (2013) .

Indices

i, k : index for items or level/shelf/strip or panels. (if item i initializes a level, then name/number of that level is also i. If level k initializes a panel, i.e. the level is the first

of all levels assigned to that panel, then the name/number of the panel is also k. {1….ñ} where ñ : total number of items $= \sum_{j=1}^{n} d_j$

j : index for item types {1…n}

h : index for panel types

Parameters

d_j : demand of item type j

α_j : possible levels that item j can be cut in. $\alpha_j = \sum_{s=1}^{j} d_s$ (By the structure of the model, the first item in a level must have the largest width)

β_k : the very first item type (not item, item type) in level k. $\beta_k = \min\{r : 1 \leq r \leq n, \alpha_r \geq k\}$ In any level k, the item types in range $[\beta_k, n]$ can be cut.

In short, α values relate item types to item/level name/numbers, β values relate panel and level numbers to item types.

C_h : cost of using a panel type h.

W_h : width of a panel type h.

L_h: length of a panel type h.

l_j : length of an item

w_j : width of an item

We additionally define the following parameters:

r_j : revenue of item type j.

num_h: total number of available panels of type h.

To define rotation of items, we introduce the following parameters:

rot_j : Decision maker can change lengths with widths and define a new item. If such a case occurs, i.e. a new item is created as a rotated counterpart of some other item, then the value for rot_j=1, stating j is a rotated counterpart of some other item.

rot_0_j If the item is rotated item which is created by decision maker, then rot_0_j gives the index of the original item.

rot_1_j gives the index of rotated counterpart of item j.

Decision Variables

y_{ih} : binary decision variable that represents if level i is assigned to a panel type h or not. i ∈ {1… ñ} and h ∈ {1… m}

x_{ijh} : integer decision variable that represents the number of items of type j packed into level i in a panel of class h. i ∈ {1… ñ-1}, h ∈ {1… m}, j ∈ {β_i… ñ}.

q_{kh} : binary decision variable that takes value of 1 if panel k is a member of panel h class. (This DV also shows that there is a level k that initializes a panel, so the

name/number of the panel is k. This DV further shows that item k is the very first item in the panel k of class h. (Item k is located on the left upper corner of the panel if this DV is 1) k ∈ {1… ñ} and h ∈ {1… m}

z_{kih} : From DV q, we can detect one level that is assigned to a panel (the one that initializes the panel). For the other levels that are also assigned to panel k, we use DV z. This DV takes value 1 if level i is allocated to panel k of class h. k ∈ {1… ñ-1}, h ∈ {1… m}, i ∈ {k… ñ}.

Figure 3.3 shows the assignments of items to a panel and the associated decision variables for an example instance with four items and following specifications:

Table 3.1 *Item Specification of the Example*

Item Type (j)	Demand	α_j
1	4	4
2	3	7
3	5	12
4	8	20

i	β_i
1-4	1
5-7	2
8-12	3
13-20	4

Suppose we have assignments of the items to a panel of type 1 as in the left side of Figure 3.3. Then, the decision variables may take the values as in the right side of Figure 3.3. Note that different q and z variables can take value of 1. For example, for the assignments in Figure 3.3 one of the $q_{i,1}$ can take value of 1, as long as i is between 5 and 7. Similarly, different $z_{k,i,1}$ values can take value of 1, as long as the i values are compatible with the α_j of each item type. (i ∈ [8,12] for the second level and i ∈ [13,20] for the third level)

Item 2	Item 2	Item 1	Item 1
Item 3	Item 4	Item 1	Item 1
Item 4	Item 3	Item 2	Item 4

$q_{5,1} = 1$	$x_{5,2,1} = 1$	$x_{5,1,1} = 2$	
$z_{5,8,1} = 1$	$x_{8,4,1} = 1$	$x_{8,1,1} = 2$	
$z_{5,13,1} = 1$	$x_{13,3,1} = 1$	$x_{13,2,1} = 1$	$x_{13,4,1} = 1$

Figure 3.3 Sample Representation of Assignments and Decision Variables in Model 1

The constraint set taken from Furini and Malaguti (2013) is stated below:

Constraints

(1) Demand Constraint: The total number of item j cut should satisfy the demand. x variables count the number of items of type j assigned to levels excluding the first items of levels, while y variables count the levels that have item j as the first item.

$$\sum_{h}\sum_{i=1}^{\alpha_j} x_{ijh} + \sum_{h}\sum_{i=\alpha_{j-1}+1}^{\alpha_j} y_{ih} \geq d_j \qquad \forall j \in \{1 \dots n\} \qquad (1)$$

(2) Maximum length constraints: Total length of items assigned to a level should be less than or equal to the length of the panel.

$$\sum_{j=\beta_i}^{n} x_{ijh} * l_j \leq \left(L_h - l_{\beta_i}\right) * y_{ih} \quad \forall \mathrm{i} \in \{1 \dots \tilde{\mathrm{n}} - 1\} \text{ and } \mathrm{h} \in \{1 \dots m\} \qquad (2)$$

(3) *Total width constraints*: Sum of widths of items that initializes levels in a panel should less than or equal to the width of that panel type.

$$\sum_{i=k+1}^{\tilde{\mathrm{n}}} z_{kih} * w_{\beta_i} \leq \left(W_h - w_{\beta_k}\right) * q_{kh} \quad \forall \mathrm{k} \in \{1 \dots \tilde{\mathrm{n}} - 1\}, \mathrm{h} \in \{1 \dots m\} \quad (3)$$

(4) *Panel-level relationship constraints*: If there is a level i defined, then it should be assigned to a panel or should initialize a panel.

$$\sum_{k=1}^{\mathrm{i}-1} z_{kih} + q_{ih} = y_{ih} \qquad \forall \mathrm{i} \in \{1 \dots \tilde{\mathrm{n}}\} \text{ and } \mathrm{h} \in \{1 \dots m\} \tag{4}$$

We modify the model of Furini and Malaguti (2013) so as to include the rotatable items.

In our problem, total number of real items and the rotated counterparts cut should be less than or equal to the demand of that item. Number of items that initialize the levels and number of items assigned on those levels constitutes the total number of items cut.

(5) Demand Constraint

If $rot_j = 0$*:*

$$\sum_{h}\sum_{i=1}^{\alpha_j} x_{ijh} + \sum_{h}\sum_{i=1}^{\alpha_{rot\_1_j}} x_{irot\_1_jh} + \sum_{h}\sum_{i=\alpha_{rot\_1_{j-1}}+1}^{\alpha_{rot\_1_j}} y_{ih} + \sum_{h}\sum_{i=\alpha_{j-1}+1}^{\alpha_j} y_{ih} \leq d_j \quad \forall j$$

We introduce *the panel availability constraint* as

$$\sum_{k=1}^{\tilde{\mathrm{n}}} q_{kh} \leq num_h \qquad \mathrm{h} \in \{1 \dots m\} \tag{6}$$

Total number of panels used of a type cannot exceed the total available number of panels of the same type.

Our objective function includes the total net revenue, i.e. profit.

(7) *Objective Function*

$$max(\, Total\ Revenue - \ Total\ Cost)$$

$$Total\ Revenue\ = \sum_{h} \sum_{i=1}^{ñ-1} \sum_{j=\beta_i}^{n} r_j * x_{ijh} + \sum_{h} \sum_{i=1}^{ñ} y_{ih} * r_{\beta_i}$$

$$Total\ Cost\ = \sum_{h} \sum_{k=1}^{ñ} q_{kh} * c_h$$

Model 1 is efficient only when the demands of the items, thereby ñ is low. Even for small sized problem instances with high demand ñ may become so high, thereby inflating the number of decision variables. Recognizing this drawback as mentioned in Furini and Malaguti (2013), we introduce Model 2 that is discussed next. We defined fittable strips for each panels and maximum number of panels that can be used, in place of fittable strip and feasible panel for each item. We keep the idea of partition in the formulation.

3.2.2. Model 2

Model 2 defines fittable strips for each panel and maximum number of panels that can be used, in place of fittable strips and feasible panels for each item. The idea of partitioning is kept and three sets (in place of four) of decision variables are used. The

reduction is due to the decision variable set that relates the levels to the panels and panel types. The number of those decision variables is $ñ \times ñ \times num_h$ in Model 1, can be very big when the item demands are high.

We also derive some properties to define fittable strips for each panel and maximum number of panels that can be used to cut all items. To find the maximum number panels, we propose the following procedure:

Procedure

Step 1. Let J the set of items in their non-increasing order of widths. Let panel k be a single panel of type h and a_j be the number of items of type j placed on panel k.

The levels represent the number of levels assigned to panel k

let k = 0, levels = 0.

Starting from the first item of set J, the levels of panel k are created by the following rule:

for j in J:

while $a_j \leq d_j$ and $\sum_{j \in J} \sum_{i=1}^{a_j} w_j \geq W_h$

add one item j to panel k, $a_j = a_j + 1$

Update demands as $d'_j = d_j - a_j$

Update number of levels, levels= levels +1

$L_{levels} = L_h - l_j$

Step 2 Let D' be the updated demand vector and let J' be the set of items in their non-decreasing order of widths.

Starting from the first item of J', the levels are filled by items using the following rule:

for i in levels:

for j in J':

while $a_j \leq d'_j$ and $\sum_{j \in J'} \sum_{i=1}^{a_j} l_j \geq L_i$

add one item j to panel k, $a_j = a_j + 1$

Update demands as $d'_j = d_j - a_j$

Step 3. levels = 0, k=k+1

If $k \geq num_h$ or D' is empty, stop. $num_h = k$

Otherwise, go to step 1.

In step 1, the non-increasing width ordered items are taken as the first items of the levels. The demands of the items are updated and in Step 2, the items are assigned to those predefined levels. In each step, demands are updated and updated demand set D' is used thereafter. Feasibility is guaranteed as the widths are always greater than or equal to the widths of items in set D'. The iterations are continued until the number of panels used reaches the number of available panels for type h, or until demand set D' becomes empty. If the iteration count reaches the number of available panels, then there is no reduction in the number of available panels. Otherwise, the number of available panel of type h is updated as all demand can be satisfied with fewer panels than the original value.

To define fittable strips for each panel instead of defining strips for each item, we calculate upper bound of levels for item type j in panel type h, $UBLevel_{jh}$:

$$UBLevel_{jh} = min\left\{d_{jh}, \left\lfloor \frac{W_h}{w_j} \right\rfloor\right\} \; where$$

$$d_{jh} = \min\{d_j, \left\lfloor \frac{W_h}{w_j} \right\rfloor * \left\lfloor \frac{L_h}{l_j} \right\rfloor\}$$

This new formulation requires the following sets of decision variables, where indices i and j represent item types, h represents panel types and k represents panels.

q_k : represents if panel k is used or not. k ∈ {1…. $\sum_h num_h$}

for each panel type h, $\alpha_h = \sum_{s=1}^{h} num_s$ and

for each panel, $\beta_k = \min\{r : 1 \leq r \leq h,\ \alpha_r \geq k\}$ is defined.

z_{imk}: binary variables which take value of 1 if item i initializes a level in panel k, where i $\in \{1, \ldots, n\}$, k $\in \{1, \ldots, \sum_h num_h\}$ and m $\in \{1, \ldots, UBLevel_{i\,\beta_k}\}$. m is the level index of item i in panel k.

x_{jimk}: is the integer decision variables that represent the number of item type j that is assigned to level (i,m,k) where i and j $\in \{1, \ldots, n\}$, k $\in \{1, \ldots, \sum_h num_h\}$ and m $\in \{1, \ldots, UBLevel_{i\,\beta_k}\}$, if $w_j \leq w_i$, i.e., the variable is only defined when item j can be assigned to level i, where the width of the level is greater than or equal to width of item j.

Figure 3.4 shows the assignments of items to a panel and the decision variables for that assignment.

Item 2	Item 2	Item 1	Item 1
Item 3	Item 4	Item 1	Item 1
Item 3	Item 3	Item 2	Item 4

$z_{2,1,1} = 1$	$x_{2,2,1,1} = 1$	$x_{1,2,1,1} = 2$	
$z_{3,1,1} = 1$	$x_{4,3,1,1} = 1$	$x_{1,3,1,1} = 2$	
$z_{3,2,1} = 1$	$x_{3,3,2,1} = 1$	$x_{2,3,2,1} = 1$	$x_{4,3,2,1} = 1$

Figure 3.4 Sample Representation of Assignments and Decision Variables in Model 1

Note that the q variables are independent of the first level of each panel. They are limited with the available amount of panels and q_k takes value of 1 if k'th panel of type h is used.

Model 2 is defined by the following constraint sets and objective function.

Constraints

$$\sum_j x_{jimk} * l_j + z_{imk} * l_i \leq z_{imk} L_{\beta_k} \ \forall (i, m, k) \qquad (8)$$

$$\sum_{i,m} z_{imk} * w_i \leq q_k W_{\beta_k} \quad \forall\, k \tag{9}$$

$$\sum_{i,m,k} x_{jimk} + \sum_{m,k} z_{jmk} \leq d_j \quad \forall\, j \tag{10}$$

Objective Function

$$\max \sum_{j,i,m,k} x_{jimk} * r_j + \sum_{i} z_{imk} * r_i - \sum_{k} c_{\beta_k} q_k \tag{11}$$

Constraint set (8) ensures that the total length of items assigned to a level cannot exceed the length of that panel. Constraint set (9) avoids the assignments whose total width exceeds the width of the panel. Constraint set (10) states that the each item cut is limited by its demand. Objective function maximizes total profit over all items and all panels.

We do not need a constraint set that relates panel types to panels and levels in the panels (as constraint set (4) of Model 1). Moreover, we do not include available number of panels constraint set, as this constraint is implicitly used in defining the panel set.

Comparing Model 1, we get good reductions in number of decision variables. Parametric comparison of number of decision variables in Model 1 and Model 2 can be seen in Table 3.2.

Table 3.2 *Parametric Comparison of Number of Decision Variables*

Model 1		**Model 2**	
Sets of DVs	**# of DVs**	**Sets of DVs**	**# of Decision Variables**
x_{ijh}	ñ × n × p	x_{jimk}	$\sum_h \min\left\{d_j, \left\lfloor \frac{W_h}{w_j} \right\rfloor\right\} \times n^2 \times num_h$
y_{ih}	ñ × p	z_{imk}	$\sum_h \min\left\{d_i, \left\lfloor \frac{W_h}{w_i} \right\rfloor\right\} \times n \times num_h$
q_{kh}	ñ × p	q_k	$\sum_h num_h$
z_{kih}	ñ × ñ × p		
Total	ñp(n+ñ+2)	Total	$\sum_h num_h(\min\left\{d_j, \left\lfloor \frac{W_h}{w_j} \right\rfloor\right\} n + n^2 \min\left\{d_j, \left\lfloor \frac{W_h}{w_j} \right\rfloor\right\} + 1)$

3.3. Properties of Optimal Solution

In this section, we propose some properties of optimal solutions and discuss their incorporations to the mathematical models.

Our properties are of two types:

1. Dominance

2. Update the parameters (demand and supply quantities)

3.3.1. Dominance Properties

We first define item and panel domination concepts. Based on our definitions we present the theorems that state the effects of the dominance properties on the solution.

3.3.1.1. Item Domination

Rotation case: Item i dominates item j when either

$$r_i \geq r_j,\ w_j \geq w_i,\ l_j \geq l_i \quad \text{or} \quad r_i \geq r_j,\ w_j \geq l_i,\ l_j \geq w_i.$$

No rotation case: Item i dominates item j if $r_i \geq r_j,\ w_j \geq w_i,\ l_j \geq l_i$.

Figure 3.5 illustrates item domination where revenue of item 2 is greater:

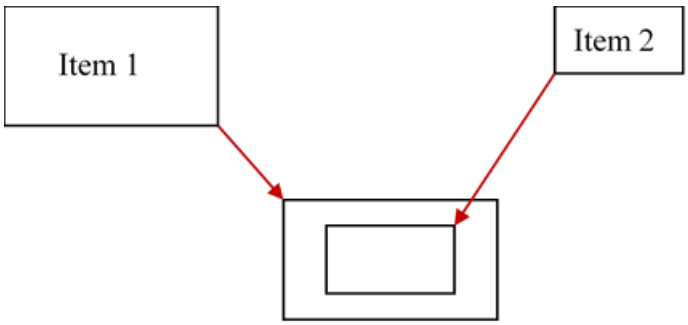


Figure 3.5 Item Domination Representation

In the figure, item 2 dominates item 1, because the width and length of item 2 is smaller and its revenue is higher.

Theorem 1: If item i dominates item j then the following condition holds:

If the amount of item i cut is less than its demand d_i then the amount of item j cut is zero.

Proof: Assume a solution in which $x_i < d_i$ and $x_j > 0$, i.e., a solution that contradicts with the condition of the theorem. Replacing one unit of item i by one unit of item j improves the objective function by $r_i - r_j$ units. The improvements can be continued and by unit replacements of item i and item j, till x_i reaches to its upper limit of d_i or x_j reaches to its lower limit of zero, whichever is earlier. Hence a solution in which

$x_i < d_i$ and $x_j > 0$, i.e., that contradicts with the condition of the theorem cannot be optimal.

Item Domination Property for Model 1

We incorporate the item domination condition to Model 1 through the following binary variable

g_j = 1 if item j is cut; 0 otherwise

We modify the demand constraints for all pairs of dominating item i and dominated item j. Note that the level index "i" is changed with "m", to avoid confusion.

$$\sum_h \sum_{m=1}^{\alpha_j} x_{mih} + \sum_h \sum_{m=1}^{\alpha_{rot\_1_i}} x_{mrot\_1_i h} + \sum_h \sum_{m=\alpha_{rot\_1_{i-1}}+1}^{\alpha_{rot\_1_i}} y_{mh} + \sum_h \sum_{m=\alpha_{i-1}+1}^{\alpha_i} y_{mh} \geq d_i g_j \quad (12)$$

$$\sum_h \sum_{m=1}^{\alpha_j} x_{mjh} + \sum_h \sum_{m=1}^{\alpha_{rot\_1_j}} x_{mrot\_1_j h} + \sum_h \sum_{m=\alpha_{rot\_1_{j-1}}+1}^{\alpha_{rot\_1_j}} y_{mh} + \sum_h \sum_{m=\alpha_{j-1}+1}^{\alpha_j} y_{mh} \leq d_j g_j \quad (13)$$

To avoid binary variables, one may use a weaker application of item domination in Model 1.

If $d_j \leq d_i$ then the amount of item i cut must be greater than or equal to the amount of item j cut in the optimal solution. If $d_j > d_i$ then the difference between the amount of item j and item i must be smaller than or equal to the difference in the demand values of these items. Two new constraint sets can be introduced to Model 1 as:

$$If\ d_j \leq d_i : cut_i \geq cut_j$$

$$If\ d_j > d_i : cut_j - cut_i \leq d_j - d_i$$

where cut_i (cut_j) is the amount of item i (item j) cut.

Item Domination Property for Model 2

We incorporate item domination property to Model 2 as follows if item j dominates item p.

$$\sum_{i,m,k} x_{jimk} + \sum_{m,k} z_{jmk} \geq d_j g_p \qquad (14)$$

$$\sum_{i,m,k} x_{pimk} + \sum_{m,k} z_{pmk} \leq d_p g_p \qquad (15)$$

To avoid binary variables, one may use a weaker application of item domination in Model 2.

$$If\ d_p \leq d_j :\ cut_j \geq cut_p$$

$$If\ d_p > d_j :\ cut_p - cut_j \leq d_p - d_j$$

where cut_p (cut_j) is the amount of item p (item j) cut.

Example

Suppose that we have two panel types with the following parameters:

W_1 = 125	L_1= 100	C_1 = 300	num_1 = 3
W_2= 90	L_2= 90	C_2 = 310	num_2 = 2

We have four item types with the following parameters:

w_1 = 40	l_1= 50	r_1= 40	d_1= 15
w_2= 30	l_2= 40	r_2= 50	d_2= 15
w_3= 50	l_3= 50	r_3= 45	d_3= 15
w_4= 35	l_4 = 30	r_4= 55	d_4= 15

When rotations are not allowed, item 2 dominates item 1 and item 3. Item 4 also dominates item 1 and item 3. There is no dominance relation between item 2 and item 4. When rotations are allowed, item 4 also dominates item 2.

The example verifies that with rotations more dominance conditions could be established.

When the demands of all item types are the same one can argue that in the optimal solution the following relations hold:

$x_2 \geq x_1$ and $x_2 \geq x_3$

$x_4 \geq x_1$ and $x_4 \geq x_3$

If rotations are allowed:

$x_4 \geq x_2$

If the amount of dominating item 4 cut would be greater than the amount of dominated item 2 cut, then the objective function may be improved by 5 units, by replacing item 2 with item 4. Same interpretation is valid for all dominating items i with $x_i \leq d_i$. g_1 and g_3 are defined for dominated item types. Then the following constraints hold:

$x_2 \geq d_2 g_1$ and $x_1 \leq d_1 g_1$ for dominating item 2 and dominated item 1

$x_2 \geq d_2 g_3$ and $x_3 \leq d_3 g_3$ for dominating item 2 and dominated item 3

$x_4 \geq d_4 g_1$ and $x_1 \leq d_1 g_1$ for dominating item 4 and dominated item 1

$x_4 \geq d_4 g_3$ and $x_3 \leq d_3 g_3$ for dominating item 4 and dominated item 3

$x_4 \geq d_4 g_2$ and $x_2 \leq d_2 g_2$ also hold if rotations are allowed.

3.3.1.2. Panel Domination

Panel r dominates panel s if $C_s \geq C_r$, $W_r \geq W_s$, $L_r \geq L_s$

Figure 3.6 illustrates panel domination where cost of panel 1 is smaller:

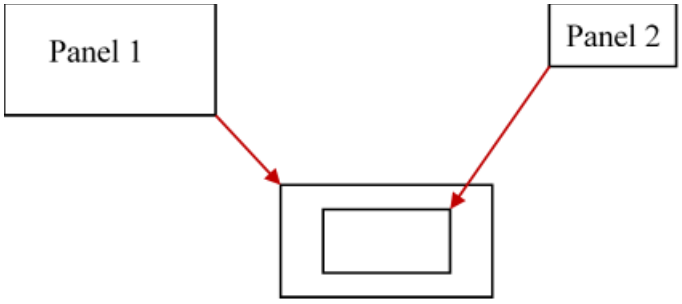


Figure 3.6 Panel Domination Representation

Panel 1 dominates panel 2 because it has lower cost with higher area($W_1 \geq W_2$, $L_1 \geq L_2$).

Theorem 2: If panel r dominates panel s then the following condition holds:

If the number of used panel r is less than its available amount then no panel of type s is used.

Proof: Assume a solution in which $y_r < num_r$ and $y_s > 0$ i.e., a solution that contradicts with the condition of the theorem. Replacing one piece of panel r with one piece of panel s improves the objective function by $C_s - C_r$ units. The improvements can be realized one by one replacements of panel r and panel s, till y_r reaches to its available amount or y_s reduces to zero. Hence a solution in which $y_r < num_r$ and $y_s > 0$ i.e., contradicts with the condition of the theorem, cannot be optimal.

Panel Domination Property for Model 1

We incorporate the following constraint to explain panel domination condition.

$$num_r - \sum_k q_{kr} \leq num_r(1 - q_{ts}) \;\; \forall\, t = 1 \ldots . TD(s) \; where \; TD(s) = \sum_j d_{js} \quad (16)$$

d_{js} is the demand of item type j defined for a panel type s, which will be explained in Section 3.4.2.

Panel Domination Property for Model 2

If panel r dominates panel s, the following constraint is introduced to Model 2:

$$num_r - \sum_{k=\alpha_{r-1}+1}^{\alpha_r} q_k \leq num_r(1 - q_t) \quad t \in \{\alpha_{s-1} + 1 \dots \alpha_s\} \qquad (17)$$

Example

Recall the example given for item domination. The panel parameters are defined as:

W_1 = 125	L_1= 100	C_1 = 300	num_1 = 3
W_2= 90	L_2= 90	C_2 = 310	num_2 = 2

Recognize that the panel 1 has larger in both dimensions, while its cost is smaller. Hence, panel 1 dominates panel 2.

Suppose that in a solution, all panels of type 2 are used whereas only two panels of type 1 are used. The assignments on panel type 2 are always feasible for panel type 1 as it has larger dimensions. Therefore, changing the panel types of the assignments increases the revenue by ($C_2 - C_1$ = 310-300) 10 units. Hence, the solution cannot be feasible if the number of type 1 panels, y_1, is smaller than num_1 and number of type 2 panels used, y_2, is greater than 0.

Also, by changing panel type 1 with panel type 2, one can gain space to cut more items. In the example, if panel 2 was fully utilized and then changed with panel 1, one can define a new level with width 35 and length 100 and can cut three units of items 4 with additional revenue of 165 (55x3) units.

To incorporate those in the formulation, one can use the following constraints, where q_{rm} is the binary variable that represents the usage of panel m of type r.

$num_1 - y_1 \leq num_1(1 - q_{21})$

$num_1 - y_1 \leq num_1(1 - q_{22})$

3.3.2. Updating the Parameters

We try to reduce the demand and availability amounts from their specified values d_j and num_h. Our reduction procedures have two effects on the model. First the model is defined relative to tighter parameter values. Second, those updates might lead to the reduction in the problem size by eliminating some non-promising items and / or non-promising panels.

3.3.2.1. Reduction in the Demand Values

We define the demand figures relative to panel type h (panel dependent demand) and let d_{jh} = maximum amount of item j that can be cut from panel h.

i. No rotation case

We create z variables for limited items, but not all items and all panels. For the item domain of z variable, we use $\min\left\{d_j, num_h\left\lfloor\frac{W_h}{w_j}\right\rfloor\right\}$, ($num_h\left\lfloor\frac{W_h}{w_j}\right\rfloor$ is the maximum number of levels of j that can be assigned to panel type h) instead of d_j and get fair reduction in the number of z variables.

We also update the demands for panels. The maximum amount of item j that can be cut from one panel of type h is $\left\lfloor\frac{W_h}{w_j}\right\rfloor\left\lfloor\frac{L_h}{l_j}\right\rfloor$. Hence at most $num_h\left\lfloor\frac{W_h}{w_j}\right\rfloor\left\lfloor\frac{L_h}{l_j}\right\rfloor$ units of item j can be cut from all panels of type h.

This follows $d_{jh} = \min\left\{d_j, num_h\left\lfloor\frac{W_h}{w_j}\right\rfloor\left\lfloor\frac{L_h}{l_j}\right\rfloor\right\}$ as the production above demand is forbidden.

The following example illustrates the computation of d_{jh} values for no rotation case.

Example

Recall the example for item domination with the following parameters:

w_1 = 40	l_1= 50	r_1= 40	d_1= 15
w_2= 30	l_2= 40	r_2= 50	d_2= 15
w_3= 50	l_3= 50	r_3= 45	d_3= 15
w_4= 35	l_4 = 30	r_4= 55	d_4= 15

The panels have the following properties:

W_1 = 125	L_1= 100	C_1 = 300	num_1 = 3
W_2= 90	L_2= 90	C_2 = 310	num_2 = 2

$$d_{11} = \min\left\{d_1, num_h \left\lfloor\frac{W_1}{w_1}\right\rfloor \left\lfloor\frac{L_1}{l_1}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{125}{40}\right\rfloor\left\lfloor\frac{100}{50}\right\rfloor\right\} = 15$$

$$d_{12} = \min\left\{d_1, num_h \left\lfloor\frac{W_2}{w_1}\right\rfloor \left\lfloor\frac{L_2}{l_1}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{90}{40}\right\rfloor\left\lfloor\frac{90}{50}\right\rfloor\right\} = 4$$

$$d_{21} = \min\left\{d_2, num_h \left\lfloor\frac{W_1}{w_2}\right\rfloor \left\lfloor\frac{L_1}{l_2}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{125}{30}\right\rfloor\left\lfloor\frac{100}{40}\right\rfloor\right\} = 15$$

$$d_{22} = \min\left\{d_2, num_h \left\lfloor\frac{W_2}{w_2}\right\rfloor \left\lfloor\frac{L_2}{l_2}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{90}{30}\right\rfloor\left\lfloor\frac{90}{40}\right\rfloor\right\} = 12$$

$$d_{31} = \min\left\{d_3, num_h \left\lfloor\frac{W_1}{w_3}\right\rfloor \left\lfloor\frac{L_1}{l_3}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{125}{50}\right\rfloor\left\lfloor\frac{100}{50}\right\rfloor\right\} = 12$$

$$d_{32} = \min\left\{d_3, num_h \left\lfloor\frac{W_2}{w_3}\right\rfloor \left\lfloor\frac{L_2}{l_3}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{90}{50}\right\rfloor\left\lfloor\frac{90}{50}\right\rfloor\right\} = 2$$

$$d_{41} = \min\left\{d_4, num_h \left\lfloor\frac{W_1}{w_4}\right\rfloor \left\lfloor\frac{L_1}{l_4}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{125}{35}\right\rfloor\left\lfloor\frac{100}{30}\right\rfloor\right\} = 15$$

$$d_{42} = \min\left\{d_4, num_h \left\lfloor\frac{W_2}{w_4}\right\rfloor \left\lfloor\frac{L_2}{l_4}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{90}{35}\right\rfloor\left\lfloor\frac{90}{30}\right\rfloor\right\} = 12$$

ii. Rotation Case

Once rotation is allowed, for the item domain of z variable, we use $\min\left\{d_j, num_h\left\lfloor\frac{WW_h}{\min\{l_j, w_j\}}\right\rfloor\right\}$, in place of d_j.

The maximum amount of item j that can be cut from all panels of type h is $num_h\left\lfloor\frac{W_h L_h}{l_j w_j}\right\rfloor$.

This follows, $d_{jh} = \min\{d_j, num_h\left\lfloor\frac{W_h L_h}{l_j w_j}\right\rfloor\}$

The following example illustrates the computation of d_{jh}values for rotation case.

Example

For the same example where the items can be rotated,

$d_{11} = \min\left\{d_1, num_1\left\lfloor\frac{W_1 L_1}{w_1 l_1}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{12500}{2000}\right\rfloor\right\} = 15$

$d_{12} = \min\left\{d_1, num_2\left\lfloor\frac{W_2 L_2}{w_1 l_1}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{8100}{2000}\right\rfloor\right\} = 8$

$d_{21} = \min\left\{d_2, num_1\left\lfloor\frac{W_1 L_1}{w_2 l_2}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{12500}{1200}\right\rfloor\right\} = 15$

$d_{22} = \min\left\{d_2, num_2\left\lfloor\frac{W_2 L_2}{w_2 l_2}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{8100}{1200}\right\rfloor\right\} = 12$

$d_{31} = \min\left\{d_3, num_1\left\lfloor\frac{W_1 L_1}{w_3 l_3}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{12500}{2500}\right\rfloor\right\} = 15$

$d_{32} = \min\left\{d_3, num_2\left\lfloor\frac{W_2 L_2}{w_3 l_3}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{8100}{2500}\right\rfloor\right\} = 6$

$d_{41} = \min\left\{d_4, num_1\left\lfloor\frac{W_1 L_1}{w_4 l_4}\right\rfloor\right\} = \min\left\{15,\ 3\left\lfloor\frac{12500}{1050}\right\rfloor\right\} = 15$

$d_{42} = \min\left\{d_4, num_2\left\lfloor\frac{W_2 L_2}{w_4 l_4}\right\rfloor\right\} = \min\left\{15,\ 2\left\lfloor\frac{8100}{1050}\right\rfloor\right\} = 14$

In either case, we try to improve, hence reduce d_{jh} values by considering the revenue brought by item j and cost incurred by panel h.

We find the maximum amount that can be cut from one panel of type h in any optimal solution, as follows:

If k_{jh} units of item j are cut from panel h then an upper bound on the maximum profit via panel h is

$$k_{jh}r_j + r_{max}\left\lfloor \frac{W_h L_h - k_{jh} w_j l_j}{min_i\{l_i w_i\}} \right\rfloor$$

The above expression is an upper bound as the maximum revenue is assumed together with minimum area for the area freed by item j.

We say k_{jh} is an upper bound on the number of units of item j that can be cut from panel h if

$$k_{jh}r_j + r_{max}\left\lfloor \frac{W_h L_h - k_{jh} w_j l_j}{min_i\{l_i w_i\}} \right\rfloor > C_h \; and$$

$$(k_{jh}+1)r_j + r_{max}\left\lfloor \frac{WW_h LL_h - (k_{jh}+1) w_j l_j}{min_i\{l_i w_i\}} \right\rfloor \leq C_h$$

And update d_{jh} as follows

$$d_{jh} = \min\{d_{jh}, k_{jh} num_h\}$$

If k_{jh}, thereby d_{jh}, is zero then we do not assign any item of type j to panel h. If k_{jh} = 0 for all h, i.e. $\sum_h k_{jh} = 0$ then we eliminate item j.

Assume S = {j | $\sum_h k_{jh} = 0$ } then we eliminate all items in S, update the item set and reduce the number of items to N -|S|.

We update d_{jh} = $\min\{d_{jh}, d_j\}$ in all appropriate places.

The following example illustrates the d_{jh}updates.

Example

Recall the example for the item domination. To calculate k_{jh} we follow an iterative process. Starting from $k_{jh} = 0$ we calculate $r_{max}\left\lfloor\frac{W_h L_h - k_{jh} w_j l_j}{min_i\{l_i w_i\}}\right\rfloor$ until it becomes smaller than or equal to C_h. We take item 3 to show the updates on panel based demands.

Suppose $k_{31} = 0$ as a starting point. Calculate

$$k_{jh} r_j + r_{max}\left\lfloor\frac{W_h L_h - k_{jh} w_j l_j}{min_i\{l_i w_i\}}\right\rfloor = 55\left\lfloor\frac{12500}{1050}\right\rfloor = 605 > C_h = 300$$

$$(k_{jh}+1) r_j + r_{max}\left\lfloor\frac{W_h L_h - (k_{jh}+1) w_j l_j}{min_i\{l_i w_i\}}\right\rfloor = 45 + 55\left\lfloor\frac{12500 - 2500}{1050}\right\rfloor$$
$$= 45 + 495 = 540 > C_h = 300$$

Increase $k_{31}$by 1 and calculate

$$(k_{jh}+1) r_j + r_{max}\left\lfloor\frac{W_h L_h - (k_{jh}+1) w_j l_j}{min_i\{l_i w_i\}}\right\rfloor = 45x2 + 55\left\lfloor\frac{12500 - 5000}{1050}\right\rfloor$$
$$= 90 + 385 = 475 > C_h = 300$$

Continue increasing $k_{31}$by 1 as the procedure has not reached the critical point yet.

$$(k_{jh}+1) r_j + r_{max}\left\lfloor\frac{W_h L_h - (k_{jh}+1) w_j l_j}{min_i\{l_i w_i\}}\right\rfloor = 45x3 + 55\left\lfloor\frac{12500 - 7500}{1050}\right\rfloor$$
$$= 135 + 220 = 355 > C_h = 300$$

$$(k_{jh}+1) r_j + r_{max}\left\lfloor\frac{W_h L_h - (k_{jh}+1) w_j l_j}{min_i\{l_i w_i\}}\right\rfloor = 45x4 + 55\left\lfloor\frac{12500 - 10000}{1050}\right\rfloor$$
$$= 180 + 110 = 290 < C_h = 300$$

Cutting four units of item 3 from the panel type 1 is not profitable. Therefore, $k_{31} = 3$.

$$d_{jh} = \min\{d_{jh}, k_{jh} num_h\}$$

For the problem where rotations allowed:

$$d_{31} = \min\{15, 9\} = 9$$

For the problem where rotations are not allowed:

$$d_{31} = \min\{12,9\} = 9$$

Same procedure is implemented to find k_{32}. Suppose $k_{32} = 0$ as a starting point. Calculate,

$$k_{jh} r_j + r_{max} \left\lfloor \frac{W_h L_h - k_{jh} w_j l_j}{min_i\{l_i w_i\}} \right\rfloor = 55 \left\lfloor \frac{8100}{1050} \right\rfloor = 385 > C_h = 310$$

$$(k_{jh} + 1) r_j + r_{max} \left\lfloor \frac{W_h L_h - (k_{jh} + 1) w_j l_j}{min_i\{l_i w_i\}} \right\rfloor = 45 + 55 \left\lfloor \frac{8100 - 2500}{1050} \right\rfloor$$
$$= 45 + 275 = 320 \geq C_h = 310$$

Increasek_{32} by 1, i.e., set $k_{32} = 1$. Calculate

$$(k_{jh} + 1) r_j + r_{max} \left\lfloor \frac{W_h L_h - (k_{jh} + 1) w_j l_j}{min_i\{l_i w_i\}} \right\rfloor = 55 \left\lfloor \frac{8100 - 5000}{1050} \right\rfloor = 90 + 110$$
$$= 200 < C_h = 310$$

Hence, k_{32}= 1.

$$d_{jh} = \min\{d_{jh}, k_{jh} num_h\}$$

For the problem where rotations allowed:

$$d_{32} = \min\{6, 2\} = 2$$

For the problem where rotations are not allowed:

$$d_{jh} = \min\{2,2\} = 2$$

We define the demand constraint relative to panel types and keep the original demand constraint if $\sum_h d_{jh} > d_j$. Hence, if $\sum_h d_{jh} > d_j$ the demand constraint below should be in the formulation.

$$\sum_h \sum_{i=1}^{\alpha_j} x_{ijh} + \sum_h \sum_{i=1}^{\alpha_{rot\_1_j}} x_{irot\_1_{jh}} + \sum_h \sum_{i=\alpha_{rot\_1_{j-1}}+1}^{\alpha_{rot\_1_j}} y_{ih} + \sum_h \sum_{i=\alpha_{j-1}+1}^{\alpha_j} y_{ih} \leq d_j \quad \forall j$$

Define $\alpha_{jh} = \sum_{s=1}^{j} d_{sh}$

Define $\beta_{kh} = \min\{r : 1 \leq r \leq n,\ \alpha_{rh} \geq k\}$

If $d_{jh} < d_j$,

$$\sum_{i=1}^{\alpha_{jh}} x_{ijh} + \sum_{i=1}^{\alpha_{rot\_1_jh}} x_{irot\_1_{jh}} + \sum_{i=\alpha_{rot\_1_{j-1}h}+1}^{\alpha_{rot\_1_jh}} y_{ih} + \sum_{i=\alpha_{j-1}h+1}^{\alpha_{jh}} y_{ih} \leq d_{jh} \quad \forall\, h \text{ and } j$$

For item 3 of the example, we let x_{3h} be the total items cut in both rotated and not rotated way from panel type h.

Then, the following constraints hold:

$x_{31} \leq d_{31} = 9$

$x_{32} \leq d_{32} = 2$

3.3.2.2. Reductions in the Number of Panels, num_h

Recall that num_h is the available number of panels of type h. We reduce num_h by considering the fact that some of the panels may not be used up to their available amount. In doing so, we define M_{ih} as an upper bound on the number of panels to be used to produce all units of item i. Accordingly,

$$M_{ih} = \left\lceil \frac{d_i}{\left\lfloor \frac{W_h}{w_i} \right\rfloor \left\lfloor \frac{L_h}{l_i} \right\rfloor} \right\rceil$$

is the maximum number of panels of type h needed to produce all units of item i.

When rotations are allowed,

$$M_{ih} = \left\lceil \frac{d_i}{\left\lfloor \frac{W_h L_h}{l_i w_i} \right\rfloor} \right\rceil$$

From panel domination, we know that panel h is used only when all panels of dominating panel types are used. Let D_h be the set of panels that dominate panel h. This follows if $\sum_{r \in D_h} num_h$ panels are used, then panel h could be used. Hence the number of panels of type h needed over the existing dominating ones is $\max\{0, \sum_i M_{ih} - \sum_{r \in D_h} num_r\}$ and num_h is updated as:

$$num_h = min\{max\left\{0, \sum_i M_{ih} - \sum_{r \in D_h} num_r\right\}, num_h\}$$

The following example illustrates the num_h updates.

Example

Recall the example for item domination. Suppose we have another panel type with the W = L = 90 and C = 320. Thus, the panel set becomes:

W_1 = 125	L_1 = 100	C_1 = 300	num_1 = 3
W_2 = 90	L_2 = 90	C_2 = 310	num_2 = 2
W_3 = 90	L_3 = 90	C_3 = 320	num_3 = 30

Recognize that the new panel type is dominated by the first two types.

For the case when rotations are not allowed, M_{ih} values are calculated as follows:

$$M_{13} = \left\lceil \frac{d_1}{\left\lfloor \frac{W_3}{w_1} \right\rfloor \left\lfloor \frac{L_3}{l_1} \right\rfloor} \right\rceil = \left\lceil \frac{15}{\left\lfloor \frac{90}{40} \right\rfloor \left\lfloor \frac{90}{50} \right\rfloor} \right\rceil = 8$$

$$M_{23} = \left\lceil \frac{d_2}{\left\lfloor \frac{W_3}{w_2} \right\rfloor \left\lfloor \frac{L_3}{l_2} \right\rfloor} \right\rceil = \left\lceil \frac{15}{\left\lfloor \frac{90}{30} \right\rfloor \left\lfloor \frac{90}{40} \right\rfloor} \right\rceil = 3$$

$$M_{33} = \left\lceil \frac{d_3}{\left\lfloor \frac{W_3}{w_3} \right\rfloor \left\lfloor \frac{L_3}{l_3} \right\rfloor} \right\rceil = \left\lceil \frac{15}{\left\lfloor \frac{90}{50} \right\rfloor \left\lfloor \frac{90}{50} \right\rfloor} \right\rceil = 15$$

$$M_{43} = \left\lceil \frac{d_4}{\left\lfloor \frac{W_3}{w_4} \right\rfloor \left\lfloor \frac{L_3}{l_4} \right\rfloor} \right\rceil = \left\lceil \frac{15}{\left\lfloor \frac{90}{35} \right\rfloor \left\lfloor \frac{90}{30} \right\rfloor} \right\rceil = 3$$

$$num_3 = min\{max\left\{0, \sum_i M_{i3} - \sum_{r \in D_3} num_r\right\}, num_3\}$$

$$num_3 = min\{max\{0, (8 + 3 + 15 + 3) - (3 + 2)\}, 30\}$$

$$num_3 = 24$$

If rotations are allowed, M values are calculated as $M_{13} = 4$ $M_{23} = 3$ $M_{33} = 5$ $M_{43} = 3$. Then,

$$num_3 = min\{max\{0, (4 + 3 + 5 + 3) - (3 + 2)\}, 30\}$$

$$num_3 = 10$$

We improve num_h by considering the profit brought by any feasible solution, say Z_B. Z_B can be found through a simple heuristic rule or a decision maker might have faced with a similar instance before so that she/he some idea about the total profit.

One upper bound on the total revenue is $\sum_j d_j * r_j$ where all items are cut. This follows

$$\sum_j d_j * r_j - Total\ Panel\ Cost \geq Z_B$$

$$If \sum_{r \in D_h} num_r * C_r \geq \sum_j d_j * r_j - Z_B$$

then panel h is eliminated. This is due to the fact that panel h could be used if all dominating panels are cut, and all dominating panels when cut cannot lead to a solution that beats Z_B.

$$If \sum_{r \in D_h} num_r * C_r < \sum_j d_j * r_j - Z_B$$

then the following holds

$$\sum_{r \in D_h} num_r * C_r + tC_h \leq \sum_j d_j * r_j - Z_B$$

Hence, upper bound on number of panels of type h is found by the following equation:

$$num_h = \left\lfloor \frac{\sum_j d_j * r_j - Z_B - \sum_{r \in D_h} num_r C_r}{C_h} \right\rfloor$$

We use updated num_h in defining d_{jh} and stating the following constraint.

$$\sum_{k=1}^{\tilde{n}} q_{kh} \quad \leq \quad num_h$$

The following example illustrates the num_hupdates.

Example

The upper bound for the problem is calculated as follows:

$$\sum_j d_j * r_j = 15 * 40 + \ 15 * 50 + 15 * 45 + 15 * 55 = 2850$$

Recall that panel type 3 is dominated by both panel 1 and panel 2. Hence, $D_3 = \{1,2\}$

$$\sum_{r \in D_h} num_r * C_r = 3 \times 300 + 2 \times 310 = 1520$$

Suppose that decision maker somehow knows the cutting assignments that leads to a profit of 500. If this value would be smaller than or equal to 1520, then we know that using panel type 3 would produce worse assignments. However, panel 3 can be used

for the example problem to some level. We find this level by:

$$num_h = \left\lfloor \frac{\sum_j d_j * r_j - Z_B - \sum_{r \varepsilon D_h} num_r C_r}{C_h} \right\rfloor$$

$$num_3 = \left\lfloor \frac{2850 - 500 - 1520}{320} \right\rfloor$$

$$num_3 = 2$$

We eliminate panel h if num_h reduces to zero. Moreover we eliminate panel h if the condition stated by the following theorem holds.

Theorem 3: In an optimal solution panel h will not be used if the following condition holds:

$$\max\{r_i\}\, min\left\{\left\lfloor \frac{W_h L_h}{min_j\{w_j l_j\}} \right\rfloor, \left\lfloor \frac{W_h}{min_j\{w_j\}} \right\rfloor \left\lfloor \frac{L_h}{min_j\{l_j\}} \right\rfloor\right\} \leq C_h$$

Proof: An upper bound on the number of type h panel

$$min\left\{\left\lfloor \frac{W_h L_h}{min_j\{w_j l_j\}} \right\rfloor, \left\lfloor \frac{W_h}{min_j\{w_j\}} \right\rfloor \left\lfloor \frac{L_h}{min_j\{l_j\}} \right\rfloor\right\}$$

Hence, $\max\{r_i\}\, min\left\{\left\lfloor \frac{W_h L_h}{min_j\{w_j l_j\}} \right\rfloor, \left\lfloor \frac{W_h}{min_j\{w_j\}} \right\rfloor \left\lfloor \frac{L_h}{min_j\{l_j\}} \right\rfloor\right\}$ is an upper bound on the revenue that can be generated via a single panel of type h. If this revenue is no more than C_h then panel h will never be used as its cost outweighs the maximum revenue that it could generate.

Theorem 4: In an optimal solution panel h will not be used if the items are rotatable and the following condition holds:

$$\max\{r_i\}\, min\left\{\left\lfloor\frac{W_h L_h}{min_j\{w_j l_j\}}\right\rfloor, \left\lfloor\frac{W_h}{min_j\{w_j, l_j\}}\right\rfloor \left\lfloor\frac{L_h}{min_j\{w_j, l_j\}}\right\rfloor\right\} \leq C_h$$

Proof: The proof is omitted as it directly follows that of Theorem 3.

The following example illustrates the application of the Theorem 3 and 4.

Example

Consider the example for item domination. The upper bound on the number of type 3 panel is found as follows:

If rotations are possible:

$$min\left\{\left\lfloor\frac{8100}{1050}\right\rfloor, \left\lfloor\frac{90}{30}\right\rfloor \left\lfloor\frac{90}{30}\right\rfloor\right\} = \min\{7,9\} = 7$$

If rotations are not allowed:

$$min\left\{\left\lfloor\frac{8100}{1050}\right\rfloor, \left\lfloor\frac{90}{30}\right\rfloor \left\lfloor\frac{90}{30}\right\rfloor\right\} = \min\{7,9\} = 7$$

Hence, the upper bound on the revenue would be $7\times\max\{r_i\}=7\times55=385$ for item rotations allowed case. For non-rotated items, the upper bound on the revenue is also $7\times\max\{r_i\}= 385$. Panel type 3, which is dominated by both panel type 1 and panel type 2 cannot be eliminated.

3.4. Further Improvements in Model 2

We recognize some properties of the optimal solution and propose some pre-processing procedures and valid inequalities for our models. In this section, we will briefly discuss on the new valid inequalities proposed for Model 2. Also, a reduction technique in the number of decision variables x is presented.

3.4.1. Ordering Constraints

Margot (2010) states that "An integer linear program (ILP) is symmetric if its variables can be permuted without changing the structure of the problem." The symmetry elimination strategy of Model 1 was defining partition the items into two, ones that initialize the levels, and the ones that are assigned to levels. Same strategy is kept in Model 2 as well. We accommodate further symmetry elimination strategies in Model 2.

Recall that in Model 2, we define panel decision variables for each panel type as many as their available quantities.

For each panel type h, $\alpha_h = \sum_{s=1}^{h} num_s$ and

For each panel, $\beta_k = \min\{r : 1 \leq r \leq h,\ \alpha_r \geq k\}$ is defined.

Theorem 5: For a panel type h, there exists an optimal solution in which

$$q_k \geq q_{k+1} \quad \forall\, k \in \{\alpha_{h-1} + 1 \dots \alpha_h\}$$

Proof: Suppose that in the optimal solution, the proposed constraint is violated, i.e. $q_{k+1} > q_k$. It implies that some of the panels of type h are not used. Due to the fact that each $k \in \{\alpha_{h-1} + 1 \dots \alpha_h\}$ represents same panel type, changing panel k+1 with k is possible and does not change the optimal solution. Hence, via the constraint, the symmetric solutions, are eliminated, but not the optimal solution.

Recall that in our new formulation, we defined fittable strips for each panel. The maximum number of levels/strips for a panel is defined dependent on both item and panel type and can be find as:

$$UBLevel_{jh} = min\left\{d_{jh}, \left\lfloor \frac{W_h}{w_j} \right\rfloor\right\}$$

Theorem 6: For a panel type h, there exists an optimal solution in which

$$z_{imk} \geq z_{i(m+1)k} \quad \forall\, i,k \; and \; m \in \{1, \dots, UBLevel_{i\,\beta_k}\}$$

Proof: Suppose that in the optimal solution, the proposed constraint is violated, i.e. $z_{i(m+1)k} > z_{imk}$. It implies that some of the possible strips of item i is not used. Due to the fact that each $m \in \{1, \dots, UBLevel_{i\,\beta_k}\}$ represents levels with same characteristics, i.e. all of them are initialized by item i, changing level m+1 with m is possible and does not change the optimal solution. Hence, by including the constraint, one can only eliminate some symmetric solutions, not the optimal solution.

3.4.2. Reduction in x Variables

Recall that z variables represent the levels while the x variables represent the assignments to those levels. We define x variables only if the width of the item is smaller than or equal to the width of the level. Luckily, Model 2 gives room to more improvements.

Recall L_h is the length of the panel and l_j is the length of the item. If $\lfloor (L_h - l_i)/l_j \rfloor < 1$, then it is not necessary to define a decision variable for the assignment of item j to level i as there would be no integer solution. Recognizing this, when the length of item j and the first item of the level exceed the capacity of the panel, the decision variable is not defined.

3.5. Improved Models in Summary

Both improved models benefit from the presolving techniques that we suggest in Properties of Optimal Solutions. Constraint sets and the objective functions are presented below to clarify the differences between them.

The performances of the proposed strong and weak inequalities for Item Domination are analyzed on benchmark problem instances. In majority of the instances, the addition of extra binary variables is not justified by the amount of improvement that

the strong item elimination property brings. Hence, we decide not to include strong inequalities in our improved models. Moreover, the weak inequalities do not lead to any reduction, so we did not include them as well.

For the sake of completeness, we give the improved versions of Model 1 and Model 2.

Improved Model 1

$$\max \quad \sum_{h}\sum_{i=1}^{ñ-1}\sum_{j=\beta_i}^{n} r_j * x_{ijh} + \sum_{h}\sum_{i=1}^{ñ} y_{ih} * r_{\beta_i} - \sum_{h}\sum_{k=1}^{ñ} q_{kh} * c_h \qquad (18)$$

$$\sum_{h}\sum_{i=1}^{\alpha_j} x_{ijh} + \sum_{h}\sum_{i=\alpha_{j-1}+1}^{\alpha_j} y_{ih} \leq d_j \quad \forall j \qquad (19)$$

if $\sum_h d_{jh} > d_j$

$$\sum_{h}\sum_{i=1}^{\alpha_j} x_{ijh} + \sum_{h}\sum_{i=\alpha_{j-1}+1}^{\alpha_j} y_{ih} \leq d_j \quad \forall j \qquad (20)$$

If $d_{jh} \leq d_j$,

$$\sum_{i=1}^{\alpha_{jh}} x_{ijh} + \sum_{i=\alpha_{j-1}h+1}^{\alpha_{jh}} y_{ih} \leq d_{jh} \quad \forall\, h \,and\, j \qquad (21)$$

$$num_r - \sum_{k} q_{kr} \leq num_r(1 - q_{ts}) \quad \forall\, t \qquad (22)$$

$$\sum_{k=1}^{ñ} q_{kh} \leq num_h \qquad (23)$$

Improved Model 2

$$\max \sum_{j,i,m,k} x_{jimk} * r_j + \sum_{i} z_{imk} * r_i - \sum_{k} c_{\beta_k} q_k \quad (24)$$

$$\sum_{j} x_{jimk} * l_j + z_{imk} * l_i \leq z_{imk} L_{\beta_k} \quad \forall (i,m,k) \quad (25)$$

$$\sum_{i,m} z_{imk} * w_i \leq q_k W_{\beta_k} \quad \forall k \quad (26)$$

$$\sum_{i,m,k} x_{jimk} + \sum_{m,k} z_{jmk} \leq d_j \quad \forall j \quad (27)$$

$$num_r - \sum_{k=\alpha_{r-1}+1}^{\alpha_r} q_k \leq num_r(1 - q_t) \quad t \in \{\alpha_{s-1} + 1 \dots \alpha_s\} \quad (28)$$

$$z_{imk} \geq z_{i(m+1)k} \quad \forall i,k \text{ and } m \quad (29)$$

$$q_k \geq q_{k+1} \quad \forall k \quad (30)$$

3.6. Heuristics

The satisfactory behavior of the mathematical model for the small sized instances has motivated us to use it to solve large sized instances. In doing so, we decompose the problem into small sub-problems and solve each problem optimally by improved Model 2. Improved Model 2 is selected due to its notably better performance over Model 1. Basically, we propose two types of decomposition:

1) Panel Type Decomposition
2) Panel Unit Decomposition

In panel type decomposition approach, we order the panels according to their cost/area values. Note that our objective maximizes total profit thereby asking for cheaper

panels and higher units of item cut. Therefore cheaper panels with larger area (to accommodate many more item cuts) are preferable. To take both cost and area concerns into account, we give priority to the panel type having smallest cost/area and then second smallest and so on. We terminate whenever if all panel types or all items are cut. Below is the stepwise description of the panel type based decomposition algorithm.

3.6.1. Algorithm Panel Type Based Decomposition

Step 0 - Order the panel types according to their non-decreasing order of cost/area values, i.e.

$$\frac{C_1}{A_1} \leq \frac{C_2}{A_2} \leq \frac{C_3}{A_3} \leq \cdots \leq \frac{C_m}{A_m}$$

where, m = number of panel types, A_r is the area of panel type r, and C_r is the cost of panel type r.

Set r = 1 and z = 0.

N is the set of all items

Step 1 - Solve Improved Model 2 for panel type r with num_r panels and with set N.

Let S_{rk} be the set of items in the k_{th} panel of type r.

Z_H be the profit of the solution

$Z_H = Z_H + Z_r$

$num_r = O_{num_r}$

N = N / $\bigcup_k S_{rk}$

Step 2 - Improvement Step

Starting from the least costly unassigned panel types (among the panels r+1,….m) if there exists one type such that $C_s < C_r$ (thereby $A_s < A_r$) and could accommodate all items in S_{rk} for any individual panel k then assign all items in S_{rk} to one panel of s and let

$num_r = num_r + 1$

$num_s = num_s - 1$

$Z_H = Z_H + C_r - C_s$

Continue with Step 2 until no transfers are possible.

If $num_r \geq 1$ and N is not empty, then go to Step 1.

Step 3 : If N is empty or r = m, stop. Otherwise,

r = r + 1 and go to Step 1.

In the improvement step, we improve the solution if any other cheaper panel type could be used without sacrificing from the total revenue. If such a panel type exists we empty the already used panel with not yet used panel, and resolve for the already used panel with the not yet cut items.

The example below illustrates the execution of the algorithm.

Example

Suppose that we have three panel types with the following parameters:

W_1 = 104	L_1 = 100	C_1 = 70	num_1 = 2
W_2 = 80	L_2 = 80	C_2 = 50	num_2 = 2
W_3 = 100	L_3 = 100	C_3 = 75	num_3 = 2

We have three item types with the following parameters:

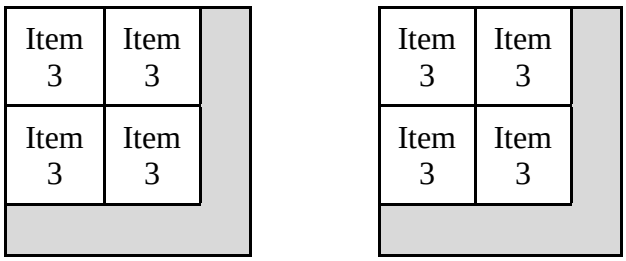
w_1 = 54	l_1 = 50	r_1 = 40	d_1 = 20
w_2 = 25	l_2 = 25	r_2 = 11	d_2 = 32
w_3 = 40	l_3 = 40	r_3 = 45	d_3 = 8

Step - 0 Panels are ordered by their cost/area values.

$$\frac{70}{10400} \leq \frac{75}{10100} \leq \frac{50}{6400}$$

The order of panels is Panel 1 – Panel 3 – Panel 2.

Step – 1 Solving Improved Model 2 for two panels of type 1 gives optimal assignments to the panels are as shown below. The grey area is the residual area on the panels.



$$Z_H = Z_H + 8 \times 45 \ - 2 \times 70 = 220$$

Step 2 – The total length and the total width of the assignments are both 80, which could fit to a panel of type 2 having lower cost.

$num_1 = num_r + 2 = 2$

$num_s = num_s - 2 = 0$

$Z_H = 220 + 2 \times 20 = 260$

Algorithm backs to Step 1 as $num_1 \geq 1$.

Step 1 - Resolving for the panel type 1 gives the following solution:

Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2

Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2

$Z_H = 260 + 32 \times 11 - 2 \times 70 = 472$

Step 2 is skipped as there is no smaller cost alternative to panel 1, other than panel 2 and panel 2 is no more available.

Step 1 - Solving for panel type 3 (the second panel type in the ordered list) for two panels ($num_3 = 2$), the assignments obtained in each of the iterations are as follows:

Item 1	Item 1

$Z_H = 472 + 2 \times (2 \times 40 - 75) = 482$

There are no alternative panel type left, hence, the algorithm terminates.

The objective function value is 482. If the improvement step had not implemented, then the algorithm would give an objective function of 422.

Algorithm panel type decomposition somewhat dispels the exponential nature of the problem by solving relatively small subproblems. We observe that the model may have computational troubles when the number of panels of a considered type is high. Based on this observation we proposed another decomposition based algorithm that decomposes the problem into panel units, but not panel types. Algorithm panel type decomposition might consider up to m subproblems whereas unit based decomposition might have to deal with $\sum_{r=1}^{m} num_r$ subproblems. The number of subproblems are considerably higher, however much easier to solve, as each subproblem of panel r uses $num_r = 1$.

Below is the stepwise description of the unit based decomposition algorithm.

3.6.2. Algorithm Panel Unit Based Decomposition

Step 0 - Order the panel units according to their nondecreasing order of cost/area values, i.e.

$$\frac{C_1}{A_1} \leq \frac{C_2}{A_2} \leq \frac{C_3}{A_3} \leq \cdots \leq \frac{C_p}{A_s}$$

where, m = number of panel types, $p = \sum_{r=1}^{m} num_r$, A_k is the area of panel unit k, C_k is the cost of panel unit k and r represents panel types.

Set k = 1, r = 1 and z = 0

N = set of all items

Step 1 - Solve improved model 2 for panel k and set N. Let S_k be the set of items in the solution and Z_k be the profit of the solution.

$Z_H = Z_H + Z_k$

N = N / S_k

If $S_k = \emptyset$, then

k = $\sum_{s=1}^{r} num_s$

r = r + 1

Step 2 – If there exists another panel t such that t > k and $C_t < C_r$ and could accommodate all items in S_k, then empty panel r by transferring the items in set S_k to panel t. If there are more than one t, then select the one having smallest cost. If panel k is emptied and $N \neq \emptyset$,

$Z_H = Z_H + C_k - C_t$

Remove panel t from further considerations and go to Step 1.

Step 3 – If N is empty, or k = s, then stop. Otherwise,

k = k + 1,

If k > $\sum_{s=1}^{r} num_s$, then

r = r + 1

Go to Step 1.

We propose another decomposition based algorithm, which can be regarded as a slight variant of Algorithm Unit Based Decomposition. The main structure of the Algorithm Unit Based Decomposition is kept, except that the panels are ordered according to their cost values and the assignments that can be done only one panel type are prioritized. To clarify the prioritization, assume there are two panel types, r and s where $C_r < C_s$. The algorithm starts with panels of type r. Assume also that $L_s < L_r$

and $W_s < W_r$, hence there are some assignments which can only be done to panel type r.

Let X_r represents the total length of assignments of a level and Y_r represents the total width of assignments of a panel. To prioritize the assignments specific to panel r, the constraint sets below are introduced to the model:

$$X_r \geq L_s + 0.1 \quad (31)$$

$$Y_r \geq W_s + 0.1 \quad (32)$$

In doing so, the panel specific assignments are selected as long as possible so as to maximize the usage of marginal benefit -larger area- of a panel type. When it is not possible, i.e., model with constraints (31) and (32) could not find a solution, any assignment is accepted.

Algorithm Modified Unit Based Decomposition skips the improvement step as the assignments done to a panel is infeasible for the other panels as long as possible. When it is feasible for the other panel types, i.e. there are no panel specific assignments left, the exchange would bring negative or zero profit, as the Algorithm use the panels in the order of non-increasing cost.

Below is the stepwise description of the unit based decomposition algorithm.

3.6.3. Algorithm Modified Panel Unit Based Decomposition

Step 0 - Order the panel units according to their non-decreasing order of cost values, i.e.

$$C_1 \leq C_2 \leq C_3 \leq \cdots \leq C_s$$

where, m = number of panel types, $s = \sum_{r=1}^{m} num_r$ and C_k is the cost of panel k and r represents panel types.

Set k = 1, r =1 and z = 0

N = set of all items

Step 1 – Solve the Improved Model 2 with the following constraints for panel k and set N.

Let panel k is from type r and L_s be the panel with maximum length in the set having smaller length than panel type r. Similarly, let W_s be the panel with maximum width in the set having smaller width than panel type r. We add following constraints to the model:

$$(33)\ \sum_{j} x_{jimk} * l_j + z_{imk} * l_i \geq z_{imk} L_s \ \ \forall\ (i,m)$$

$$(34)\ \sum_{i,m} z_{imk} * w_i \geq q_k W_s$$

Note that if panel k has the minimum length, the first constraint drops. If it has the minimum width, then the second constraint drops.

Let S_k be the set of items in the solution and Z_k be the profit of the solution.

$Z_H = Z_H + Z_k$

N = N / S_k

If $S_k \neq \emptyset$, skip Step 2 and go to Step 3.

Step 2 – Relax the constraints (33) and (34) and solve the improved Model 2.

$Z_H = Z_H + Z_k$

N = N / S_k

If $S_k = \emptyset$ and N $\neq \emptyset$, then

k = $\sum_{s=1}^{r} num_s + 1$

r = r + 1

Go to Step 1.

If $S_k \neq \emptyset$ and N $\neq \emptyset$:

If k+1> $\sum_{s=1}^{r} num_s$, then

k = k + 1

r = r + 1 and go to Step 1.

Else,

k = k + 1 and go to Step 2.

Step 3 – If N is empty, or k = s, then stop. Otherwise,

k = k + 1,

If k > $\sum_{s=1}^{r} num_s$, then

r = r + 1

Go to Step 1.

The starting point of the algorithm, i.e., the assignments to the first panel, may change the solution significantly. We see through experimentation that starting with the second best solution for the first panel type may increase the overall performance of the algorithm. As the solution times are small, solving the algorithm by two different starting points can well be justified. One can avoid the best solution for the first panel in the second run of the algorithm.

Example

Recall the same example.

Step 0 – Panels are ordered according to their costs. The order of the panels:

Panel 2 – Panel 1 – Panel 3

Step 1 – Solve the improved model 2 for a single panel of type 2. L_2 and W_2 are the minimum values of the lengths and widths, respectively. Hence, there are no panel specific assignments.

Improved model 2 is solved two times and each time the following assignments are returned:

Item 3	Item 3
Item 3	Item 3

Z_H and N are updated.

At the end of second run, $Z_H = 260, D_1 = 20\ D_2 = 32\ D_3 = 0$

r = r + 1 = 2

k = 3

The algorithm continues to solve the Improved Model 2 for k = 3 which is of type 1. Panel type 2 was considered, so panel types 3 and 1 are compared. $W_1 = 104 > W_3 = 100$, and $L_1 = 100 = L_3 = 100$. Hence, the algorithm prioritizes the assignments which have total width of $W > W_3$ by introducing the constraint below:

$$\sum_{i,m} z_{im3} * w_i \geq 100 + 1$$

For the first panel of type 1, the following assignment is obtained:

Item 1		Item 1	
Item 2	Item 2	Item 2	Item 2
Item 2	Item 2	Item 2	Item 2

$Z_H = 260 + 168 - 70 = 358,$

$D_1 = 18\ D_2 = 24\ D_3 = 0$

There is no need to relax the constraint as a feasible assignment is reached. When solving for the second panel of type 1, the same assignments are reached.

$Z_H = 358 + 168 - 70 = 456,$

$D_1 = 16\ D_2 = 16\ D_3 = 0$

Step 3 – k = 4+1 = 5 and algorithm continues with Step 1.

Step 1 – Panel type 3 is the only panel left, there is no need to force the model to do panel specific assignments.

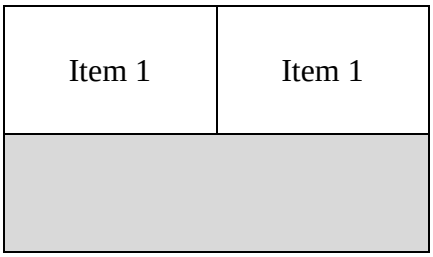
When solving for a panel of type 3, we get the following assignments.

Item 2	Item 2	Item 2	Item 2	
Item 2	Item 2	Item 2	Item 2	
Item 2	Item 2	Item 2	Item 2	
Item 2	Item 2	Item 2	Item 2	

$Z_H = 456 + 176 - 75 = 557,$

$D_1 = 16\ D_2 = 0\ D_3 = 0$

Improved Model 2 for k+1 of type 3 returns the following assignments:



$D_1 = 14\ D_2 = 0\ D_3 = 0$

$Z_H = 557 + 80 - 75 = 562$

Step 3 – There are no panel left, so algorithm terminates.

Heuristic 1 uses Algorithm Panel Based Decomposition. Heuristic 2 runs Algorithm Unit Based Decomposition and Modified Algorithm Unit Based Decomposition and takes the best solution.

CHAPTER 4

COMPUTATIONAL EXPERIMENTS

In this chapter, we discuss the computational experiments that are designed to evaluate the performances of our mathematical models and heuristic algorithms. To solve all models, we use a non-commercial solver SCIP version 6.0.1. The number of threads is set to the default value of the solver. For the heuristic algorithms and pre-solving operations of the model, we use Python 3.6 as programming language. The experiment is conducted on a computer with Intel i7-6700HQ CPU at 2.6 GHz and 8 GB RAM memory.

The performance of SCIP 6.0.0 is tested on benchmark instances by Mittelmann (2018). The summary of the results can be seen in Table 4.1. SCIPS and SCIPC differ by their LP Solver. SCIPS uses SoPlex as LP solver, while SCIPC uses CPLEX. When the number of threads is more than one, SCIP is named as Fiber SCIP and abbreviated as FSCIP in the table. For different threads, the number of instances solved, unscaled and scaled CPU times of different solvers are given in Table 4.1. The scaled CPU times are obtained by equating the fastest solver performance to 1 and the others' proportional to it. Although we used SCIP 6.0.1, we present the performance of older version because the latest available study we could reach concerns the SCIP 6.0.0.

Table 4.1 The Summary of Performance Comparison of SCIP and Other Solvers

1 thr	CBC	CPLEX	GUROBI	SCIPC	SCIPS	XPRESS	MATLAB	SAS
unscaled	1639	72.2	41.6	239	330	83.1	3002	121
scaled	39	1.74	1	5.75	7.94	2.00	72.2	2.90
solved	53	87	87	83	76	86	32	84
4 thr	CBC	CPLEX	FSCIPC	FSCIPS	GUROBI	XPRESS	MIPCL	SAS
unscaled	843	36.4	240	294	24.2	40.3	177	72.6
scaled	34.8	1.5	9.9	12.1	1	1.66	7.29	3.00
solved	66	86	80	79	87	87	84	85
12 thr	CBC	CPLEX	FSCIPC	FSCIPS	GUROBI	XPRESS	MIPCL	SAS
unscaled	668	37.5	247	328	25.2	39.5	165	85.4
scaled	27	1.49	9.8	13.0	1	1.57	6.53	3.39
Solved	69	87	78	76	87	87	82	82

In Section 4.1, we report on the instance features and data generation. In Section 4.2.1 and Section 4.2.2 we test the effects of improvement mechanisms on Model 1 and Model 2, respectively. We compare the performances of Model 1 and Model 2 in Section 4.2.3. Finally, in Section 4.2.4 we discuss the performances of the heuristic algorithms relative to the best available solutions.

4.1. Instance Features and Data Generation

We use 42 instances taken from the literature and that are available online as in Furini and Malaguti (2013). The first 12 instances are originally for the bin-packing problem by Cintra et. al. (2008). Uniform integer demand between 0 and 100 and three types of panels with dimensions (L,W), (1.2L, 0.8W) and (1.1L, 0.9W) are used in these instances.

The other 30 instances are proposed by Hifi and Roucairol (2001) for two dimensional knapsack problem. These include demand figures.

The maximum and minimum item widths (wmax and wmin) and item lengths (lmax and lmin) and maximum and minimum panel width and length (W and L) together

with the number of item types(n) and total demand (ñ) for all 42 instances are given in Table 4.2.

Table 4.2 *Features of the Problem Instances*

Instance No	**n**	ñ	**lmax**	**lmin**	**wmax**	**wmin**	**Lmax**	**Lmin**	**Wmax**	**Wmin**
1	10	669	167	66	184	86	300	275	225	200
2	20	982	168	18	186	68	300	275	225	200
3	30	1489	176	63	186	71	300	275	225	200
4	50	2751	184	62	179	63	300	275	225	200
5	10	645	364	132	356	145	600	550	450	400
6	20	1064	355	132	372	134	600	550	450	400
7	30	1626	365	129	374	147	600	550	450	400
8	50	2363	362	131	374	127	600	550	450	400
9	10	590	673	292	688	341	1200	1100	900	800
10	20	830	730	269	742	260	1200	1100	900	800
11	30	1298	674	266	745	274	1200	1100	900	800
12	50	2081	746	254	723	269	1200	1100	900	800
13	20	62	33	9	43	11	60	55	54	48
14	20	53	33	12	42	14	72	66	54	48
15	20	46	35	15	43	14	84	77	72	64
16	20	35	33	9	43	11	108	99	63	56
17	20	45	69	13	63	12	158	145	90	80
18	30	63	69	13	63	12	158	145	90	80
19	10	19	31	11	31	9	74	68	49	44
20	10	18	20	1	14	2	24	22	18	16
21	30	65	69	18	63	12	156	143	117	104
22	35	75	57	19	54	18	156	143	117	104
23	35	90	96	31	112	35	180	165	157	140
24	25	82	58	20	80	28	120	110	112	100
25	25	67	78	25	66	21	150	137	94	84
26	35	63	93	34	104	34	174	159	148	132
27	40	96	170	59	130	45	320	293	186	165
28	35	75	57	19	59	18	156	143	117	104
29	10	51	54	15	65	13	152	139	88	78
30	10	32	54	15	65	13	152	139	88	78
31	22	60	109	35	101	38	303	278	219	195

Table 4.2 (cont'd).

32	40	90	108	33	135	38	315	289	216	192
33	10	18	20	1	14	2	58	53	18	16
34	35	76	31	10	43	10	78	71	68	60
35	5	18	54	18	65	13	152	139	88	78
36	10	23	55	9	39	4	84	77	36	32
37	10	24	47	13	27	4	84	77	36	32
38	30	78	31	10	43	10	66	60	76	68
39	20	50	44	14	49	16	118	108	89	79
40	20	62	43	11	33	9	84	77	36	32
41	10	23	31	9	35	7	48	44	63	56
42	20	62	33	9	43	11	48	44	63	56

We generate some other parameters: the revenues of items, costs of panels and available number of panels.

We set the revenues using the areas of items from discrete uniform distribution as:

$$r_j = U\ [\max(10, \frac{l_j \times w_j}{10{,}000}), (100, \frac{l_j \times w_j}{100})]$$

According to the above scheme, the revenues of the items are somewhat area dependent. When the areas are small, the revenues are generated between 10 and 100, regardless of their areas. For big items, the areas do define the revenues. The selected constants 100 and 10,000 are compatible with the item areas and revenues observed in the marble company.

The panel costs are generated from discrete uniform distribution as follows:

$$C_h = U\ [\frac{A_h}{A_1} \times max_i\{r_i\}, \frac{A_h}{A_1} \times 2 \times max_i\{r_i\}]$$

where $max_i\{r_i\}$ is the maximum revenue over all items and A_h is the area of panel h.

According to the above scheme, even for the item with maximum revenue, cutting only one item from a panel is avoided. In our interpretation, cutting one item from a panel, that would be trimming. We also avoid too high panel costs, by putting an upper limit of 2 x maximum revenue of items. Areas are important factor in setting the cost of a panel. Therefore, for each panel we set a coefficient that is proportional to its area. The smallest area is taken as a baseline and coefficient $\frac{A_h}{A_1}$ is used for panel h.

To set available number num_h to each panel type, we use the objective function values (the total area used by the assigned items) of the solutions in Furini and Malaguti (2013). We divide the total area (obj) to the smallest panel area and get an overestimate on the total number of panels used in their solutions. Using this overestimate $\left\lceil \frac{Obj}{A_1} \right\rceil$, we set num_h to one panel type as follows:

$$num_h = \left\lceil \frac{Obj}{m \times A_1} \right\rceil$$

num_h values of a particular instance are the same for all panel types.

We set the number of different panel types to 2 and 3.

4.2. Analysis of the Results

We evaluate the performance of the mathematical models by their CPU times. We put a time limit of 7200 seconds for the execution of all models and report on the number of instances solved to optimality (out of 42 instances) within this limit. Moreover we give the total number of integer decision variables for each instance.

In our preliminary tests, for both models, we observe that in majority of the instances, the addition of extra binary variables is not justified by the amount of improvement that the item elimination property brings. Hence, we decided not to include the item elimination property in our runs.

4.2.1. Performance of Model 1 and Its Improved Version

In this section, we discuss the performance of Model 1 and its improved version and report the results in Table 4.3.

Table 4.3 *Performance of Model 1 and its Improved Version*

Instance Features			**3 Panel Types**				**2 Panel Types**			
			Model 1		**Improved Model 1**		**Model 1**		**Improved Model 1**	
Instance No	N	ñ	Number of integer DVs	CPU Time	Number of integer DVs	CPU Time	Number of integer DVs	CPU Time	Number of integer DVs	CPU Time
1	10	669	688638	32	619194	14	459092	12	412796	14
5	10	645	639429	7200	548669	7200	426286	863	338146	563
10	20	830	1066842	443	1011453	259	711228	233	674302	242
13	20	62	8238	14	8238	16	5492	12	5492	14
14	20	53	6147	3008	4632	1185	4098	328	2916	175
15	20	46	4971	79	3100	62	3314	34	1912	20
16	20	35	3189	652	2203	69	2126	32	1440	7
17	20	45	4686	3197	3237	1948	3124	94	1972	39
18	30	63	9153	7200	6629	7200	6102	2663	4087	522
19	10	19	969	2	798	3	646	7	532	7
20	10	18	915	13	753	6	610	3	502	5
24	25	82	13431	7200	7492	7200	8954	656	4518	106
26	35	63	9663	7200	8575	7200	6442	414	5549	357
30	10	32	2274	28	1650	16	1516	7	1100	8
33	10	18	915	21	753	7	610	7	502	6
35	5	18	774	4	435	1	516	1	290	3
36	10	23	1389	2	499	1	926	0	293	1
37	10	24	1470	185	799	8	980	1	496	1
39	20	50	5718	7200	4414	7200	3812	878	2826	727
40	20	62	8343	377	8015	703	5562	38	5234	35
41	10	23	1323	23	1059	7	882	3	684	3
42	20	62	8238	7200	5492	7200	5492	2	5492	2

When there are three panel types, Model 1 and its improved version could solve 16 out of 42 instances to optimality. We observe that the improvement mechanisms had enhanced the efficiency of the model in 13 instances. In the two instances, the performances are almost similar; Model 1 runs less than two seconds earlier than its

improved version which can be attributed to the time difference in the model construction phase of the formulations. The only instance that the original formulation significantly dominates the formulation with improvements is Instance 40. We see that the difference in number of the decision variables is small. When the instance is analyzed, we find that the panel specific demands are all equal to the original item demand. Hence no dominance relation between the panels could be established. Shortly, improvement strategies had not worked for the instance, the extra effort spent to test them increased the CPU time. This may be due to the change in the structure of the parameter and constraint sets.

When there are only two panel types, Improved Model 1 dominates Model 1 in terms of CPU time in all instances once too small differences are ignored. Both models could find the optimal solution in 22 out of 42 instances in two hours.

Table 4.4 reports on the average performances of Model 1 and improved version over all 42 instances. The average CPU times also include the instances that cannot be solved in 7200 seconds. Number of unsolved instances out of 42, average number of decision variables and the average CPU times of the instances solved by both models are also reported.

Table 4.4 *Overall Performance Measures for Model 1 and Improved Model 1*

	3 Panel Types		**2 Panel Types**	
	Model 1	**Improved Model 1**	**Model 1**	**Improved Model 1**
Average CPU Time	4649.5	4559.64	3578.2	3496.6
Average CPU Time of the Instances Solved by Both Models	505	269.1	285.7	129.9
Number of Unsolved Instances	26	26	20	20
Average Number of Integer Decision Variables Solved by Both Models	112568	104176	75355	66867.3

As can be observed from the table, the problems with 3 panel types are harder to solve than those with 2 panel types due to higher number decision variables that add burden to the solutions of the mathematical models.

4.2.2. Performance of Model 2 and Its Improved Version

The performance results of Model 2 and its improved version are presented in Table 4.5. When there are three panel types, Model 2 can solve 19 out of 42 instances while its improved version solves 26 of them. Seven additional instances are solved to optimality thanks to the effects of improvement techniques. Excluding a few instances with small differences, Improved Model 2 reached optimal solutions faster than Model 2. For the instances that both models could solve, the average CPU times are 1029 and 116 seconds, for Model 1 and its improved version, respectively. That means the improvement techniques lead to about 88% reduction in CPU times over the solved instances.

When there are two panel types, Model 2 could solve 22 instances, where Improved Model 2 could solve 32 of them in two hours. Improvements are effective on the CPU times of the solutions, i.e., improved model dominates the other in almost all instances.

Table 4.5 *Performance of Model 2 and its Improved Version*

Instance Features			**3 Panel Types**				**2 Panel Types**			
			Model 2		**Improved Model 2**		**Model 2**		**Improved Model 2**	
Instance No	N	ñ	Number of DVs	CPU Time	Number of DVs	CPU Time	Number of DVs	CPU Time	Number of DVs	CPU Time
1	10	669	16146	14	13884	14	10608	2	9438	4
5	10	645	17172	7200	10746	383	9936	8	7182	27
7	30	1626	278308	7200	179088	7200	179088	7200	146288	1139
10	20	830	75055	56	41735	56	47685	13	31195	11
13	20	62	3644	442	6000	50	2380	82	2112	18
14	20	53	1790	357	3516	73	1150	295	1150	40
15	20	46	1113	6	2226	2	697	6	697	2
16	20	35	880	14	1760	15	578	6	578	2
17	20	45	1156	248	2288	28	721	1	721	6
18	30	63	2353	6893	2341	129	1486	45	1486	10
19	10	19	291	0	291	0	194	1	194	1
20	10	18	303	0	190	0	202	0	137	1
21	30	65	2755	7200	2734	636	1757	7200	1757	154
22	35	75	3848	7200	2493	7200	2493	7200	2493	421
24	25	82	2626	892	2566	144	1636	23	1628	19
25	25	67	3088	7200	2946	1772	2020	7200	1974	318
26	35	63	4892	4470	4320	1082	3102	186	2900	52
28	35	75	3775	7200	3775	6411	2423	7200	2423	557
29	10	51	591	7200	591	23	378	7200	378	16
30	10	32	459	2	459	1	306	1	306	1
31	22	60	1784	7200	1784	407	1164	395	1164	19
32	40	90	4907	7200	3171	7200	3171	2234	3171	103
33	10	18	303	1	303	2	202	1	202	0
34	35	76	6468	7200	4082	7200	4082	7200	4082	713
35	5	18	126	0	126	0	84	0	84	0
36	10	23	216	1	199	0	128	0	120	0
37	10	24	319	4	302	2	200	1	193	0
38	30	78	5384	7200	3360	7200	3360	7200	3342	1894
39	20	50	1515	7200	1515	199	981	172	981	23
40	20	62	3635	6168	3395	355	2355	652	2290	77
41	10	23	377	0	335	0	244	1	231	0
42	20	62	6072	7200	2970	7200	3834	7200	2154	29

Table 4.6 reports on the average performances of Model 2 and improved version over all 42 instances. The average CPU times also include the instances that cannot be solved in 7200 seconds. Number of unsolved instances out of 42, average number of decision variables and the average CPU times of the instances solved by both models are also reported.

Table 4.6 *Overall Performance Measures for Model 2 and Improved Model 2*

	3 Panel Types		**2 Panel Types**	
	Model 2	**Improved Model 2**	**Model 2**	**Improved Model 2**
Average CPU Time	4408.8	3046.7	3355.4	1849
Average CPU Time of the Instances Solved by Both Models	1029.9	102.8	179.4	18.1
Number of Unsolved Instances	23	16	19	10
Average Number of Integer Decision Variables Solved by Both Models	6088.6	4538.7	3878.7	2963.5

As in Model 1, in Model 2 and its improved version the instances with 3 panel types are harder to solve than those with 2 panel types due to the higher number decision variables.

4.2.3. Comparison of Improved Model 1 and Improved Model 2

We compare the improved versions of both formulations. The performance measures of the formulations are presented in Table 4.7. One can see the significant difference between the number of decision variables as Model 1 uses decision variables for each unit of item, but not item type. That reduces symmetry in the model while increasing its complexity. The decision variables of Model 2 is defined for each panel unit so more efficient reductions can be done, as the number of panels is much less than total demand. Model 2 uses fittable items for panel units and reduces the symmetry by introducing symmetry breaking inequalities.

As expected, our experiments reveal significantly better performance of Model 2 in terms of CPU times.

When there are three panel types, Model 2 solves an additional 10 instances over the ones solved by Model 1. Average CPU time of the solved instances by Model 1 is 269 seconds. For the same instances, Model 2 runs in 37.4 seconds on average. Improved Model 2 provides better CPU times over all instances with one exception.

For two panel types, similar observations are made. Summary of the performance measures of both models are given in Table 4.7.

Table 4.7 *Comparison of Improved Model 1 and Improved Model 2*

Instance Features			**3 Panel Types**				**2 Panel Types**			
			Improved Model 1		**Improved Model 2**		**Improved Model 1**		**Improved Model 2**	
Instance No	N	ñ	Number of DVs	CPU Time	Number of DVs	CPU Time	Number of DVs	CPU Time	Number of DVs	CPU Time
1	10	669	619194	14	13884	14	412796	14	9438	4
5	10	645	548669	7200	10746	383	338146	563	7182	27
7	30	1626	4049469	7200	179088	7200	2699646	7200	146288	1139
10	20	830	1011453	259	41735	56	674302	242	31195	11
13	20	62	8238	16	6000	50	5492	14	2112	18
14	20	53	4632	1185	3516	73	2916	175	1150	40
15	20	46	3100	62	2226	2	1912	20	697	2
16	20	35	2203	69	1760	15	1440	7	578	2
17	20	45	3237	1948	2288	28	1972	39	721	6
18	30	63	6629	7200	2341	129	4087	522	1486	10
19	10	19	798	3	291	0	532	7	194	1
20	10	18	753	6	190	0	502	5	137	1
21	30	65	8245	7200	2734	636	5212	7200	1757	154
22	35	75	10998	7200	2493	7200	7074	7200	2493	421
24	25	82	7492	7200	2566	144	4518	106	1628	19
25	25	67	8312	7200	2946	1772	5393	7200	1974	318
26	35	63	8575	7200	4320	1082	5549	357	2900	52
28	35	75	10795	7200	3775	6411	6872	7200	2423	557
29	10	51	2423	7200	591	23	1517	7200	378	16
30	10	32	1650	16	459	1	1100	8	306	1
31	22	60	6096	7200	1784	407	3949	7200	1164	19
32	40	90	14006	7200	3171	7200	8977	7200	3171	103
33	10	18	753	7	303	2	502	6	202	0
34	35	76	12270	7200	4082	7200	7867	7200	4082	713
35	5	18	435	1	126	0	290	3	84	0
36	10	23	499	1	199	0	293	1	120	0
37	10	24	799	8	302	2	496	1	193	0
38	30	78	11993	7200	3360	7200	7692	7200	3342	1894
39	20	50	4414	7200	1515	199	2826	727	981	23
40	20	62	8015	703	3395	355	5234	35	2290	77
41	10	23	1059	7	335	0	684	3	231	0
42	20	62	5492	7200	2970	7200	5492	2	2154	29

Table 4.8 reports on the average performances of improved Model 1 and improved Model 2 over all 42 instances. The average CPU times also include the instances that cannot be solved in 7200 seconds. Number of unsolved instances out of 42, average number of decision variables and the average CPU times of the instances solved by both models are also reported.

Table 4.8 *Overall Performance Measures for Improved Model 1 and Improved Model 2*

	3 Panel Types		**2 Panel Types**	
	Improved Model 1	**Improved Model 2**	**Improved Model 1**	**Improved Model 2**
Average CPU Time	4559.6	3023.4	3496.5	1849
Average CPU Time of the Instances Solved by Both Models	269.1	37.4	129.9	14.7
Number of Unsolved Instances	26	16	20	10
Average Number of Integer Decision Variables Solved by Both Models	104176	4813.063	66867.32	2999.05

Table 4.8 reveals that improved Model 2 solves much more instances in considerably less CPU time. Hence it is used to solve the subproblems defined in our heuristic algorithms.

4.2.4. Computational Results for Heuristic Algorithms

In this section, we give the results of panel type based decomposition and panel unit based decomposition heuristics results and report the results in Table 4.9.

Panel type based decomposition heuristic is referred as Heuristic 1 in Table 4.9. The table resides the CPU times and the performance values. For panel unit based decomposition heuristics, the best results of the two algorithms, original and the modified one, are referred to as Heuristic 2. Modified Panel Unit Based Decomposition Algorithm is solved with two different starting points, and the best result is taken as the solution.

To avoid long solution times, we set an optimality gap of 1% of the model solutions in Heuristic 1. The performances of the heuristics are measured by their CPU times and its objective function value as a ratio of the best known objective function value (we simply refer to this ratio as performance). Best known objective function value is the optimal objective function value if optimal solution is available, if not it is the lower bound value returned at the termination limit by any model (Model 1 or Model 2). If even no lower bound is returned by the mathematical models, performance entries are left blank in the table. The instances that could be solved only when there are 2 panel types available are marked as one star next to the instance number. If the instance was solved to optimality for the two cases, i.e., 3 panel types available and 2 panel types available, it is marked with two stars. If there is no optimal solution available for both cases, the instance number is not marked.

Table 4.9 *Results of Heuristic Algorithms*

Instance No	3 panel types				2 panel types			
	CPU Time		Performance		CPU Time		Performance	
	Heuristic 1	Heuristic 2	Heuristic 1	Heuristic 2	Heuristic 1	Heuristic 2	Heuristic 1	Heuristic 2
1**	6	8	100%	100%	3	2	100%	100%
2	690	98	88%	89%	379	48	98%	95%
3	1852	125	96%	106%	1848	62	85%	89%
4	7200	488	-	-	3778	143	-	-
5**	147	11	99%	97%	4	4	100%	73%
6	1889	76	97%	98%	1809	44	100%	100%
7*	1843	115	-	-	524	7	100%	100%
8	3708	1378	-	-	2180	1141	104%	145%
9	2	26	72%	87%	7	12	100%	99%
10**	35	18	100%	100%	11	1	100%	100%
11	605	43	113%	113%	1938	29	123%	124%
12	7200	511	-	-	7200	88	-	-
13**	42	8	98%	95%	16	4	100%	94%
14**	1	8	88%	86%	1	3	97%	93%
15**	2	19	95%	95%	1	5	93%	94%
16**	3	39	93%	97%	2	9	95%	95%
17**	2	32	93%	95%	1	8	92%	94%

Table 4.9 (Cont'd).

18**	1	42	97%	97%	4	15	95%	95%
19**	0	6	100%	100%	0	3	100%	100%
20**	0	3	96%	86%	0	1	94%	95%
21**	11	198	96%	96%	16	173	95%	95%
22*	20	575	97%	98%	2	142	99%	94%
23	48	46	95%	96%	32	19	96%	94%
24**	3	18	90%	94%	1	4	96%	96%
25**	5	28	92%	94%	8	10	98%	96%
26**	11	48	91%	91%	4	16	100%	98%
27	19	69	85%	95%	50	36	94%	95%
28**	20	222	97%	96%	21	113	99%	99%
29**	2	23	99%	99%	1	10	100%	100%
30**	2	31	100%	100%	2	25	84%	85%
31**	2	28	97%	97%	5	23	96%	96%
32*	26	198	94%	96%	7	47	97%	98%
33**	1	6	100%	100%	0	3	100%	100%
34*	30	73	96%	95%	17	24	95%	95%
35**	0	2	100%	100%	0	1	100%	100%
36**	0	2	77%	76%	0	1	100%	100%
37**	0	2	94%	90%	0	1	100%	100%
38*	20	58	89%	91%	39	19	98%	96%
39**	5	62	95%	95%	7	39	92%	92%
40**	2	9	97%	91%	12	5	90%	88%
41**	0	3	94%	100%	0	2	100%	100%
42*	20	6	95%	91%	6	3	98%	89%

When there are three panel types, Heuristic 1 can solve 40 out of 42 instances in our time limit of 7200 seconds. The average CPU time over the solved instances is 277 seconds. All instances can be solved with Heuristic Algorithm 2 in 7200 seconds with an average CPU time of 113 seconds. In 19 out of 42 instances, Heuristic 2 outperforms Heuristic 1, while in 12 out of 42 instances Heuristic 1 gives better results. For 11 instances, both algorithms return the same solutions.

When there are two panel types, performance and CPU times of the heuristics are improved as expected. The solution times are decreased while the results become closer to the best available solutions.

Table 4.10 summarizes the performance measures of the heuristic algorithms.

Table 4.10 *Overall Performances of the Heuristic Algorithms*

	3 panel types		**2 panel types**	
	Heuristic 1	**Heuristic 2**	**Heuristic 1**	**Heuristic 2**
Average CPU Time	606,5	113	556	56
Maximum CPU Time	7200	1378	7200	1141
Average Performance	94%	95%	95%	97%
Minimum Performance	77%	76%	84%	73%
Number of Unsolved Instances	2	0	2	0
Number of Instances that the Heuristic Gives Better Results*	23	30	23	31

*including the ties

The average CPU time of Heuristic 2 is much better than that of Heuristic 1. However, the algorithms are not much different in terms of their closeness to the optimal solutions. The average CPU time of Heuristic 1 is about 10 times higher while Heuristic 2 performs better in terms of objective function value on average. In vast majority of the instances, Heuristic 2 returns better objective function values. Minimum, i.e., worst case performances of the algorithms are close. Heuristic 2 can solve all instances in less than 1400 seconds, while Heuristic 1 cannot solve 2 of the instances in two hours.

CHAPTER 5

APPLICATION TO SILKAR MINING INCORPORATION

This study is originated from an industrial project conducted in Silkar Mining Corporation. The corporation was established in 1994 at Bilecik to serve in natural stone sector under the AKDO brand.

Silkar Corporation leads to its sector by offering variety of marble, granite and ceramics products. The company not only has domestic customers in service and manufacturing sector in Turkey but also serves to more than 35 countries including United States, Far East, Europe and North Africa. For the more detailed information about the company we refer the reader to the company's website www.silkar.com.

The main raw materials are marble, granite or ceramic and enter to the factory from their natural sources in the forms of blocks as shown in Figure 5.1.

Figure 5.1 The main raw materials – marble blocks

These blocks are then refined so as to obtain proper three dimensional blocks. The proper dimensional blocks are then cut layer by layer to get a number of two dimensional objects. The blocks are cut at specified heights, usually equal heights. The cut two dimensional objects with required length and width are referred as panels. The number of cuts made on block h is the number of panels of type h, that is num_h according to our notation. The two dimensions of panel h are length L_h and width W_h. Figure 5.2 shows the panels of two dimensions that are obtained from three dimensional blocks.

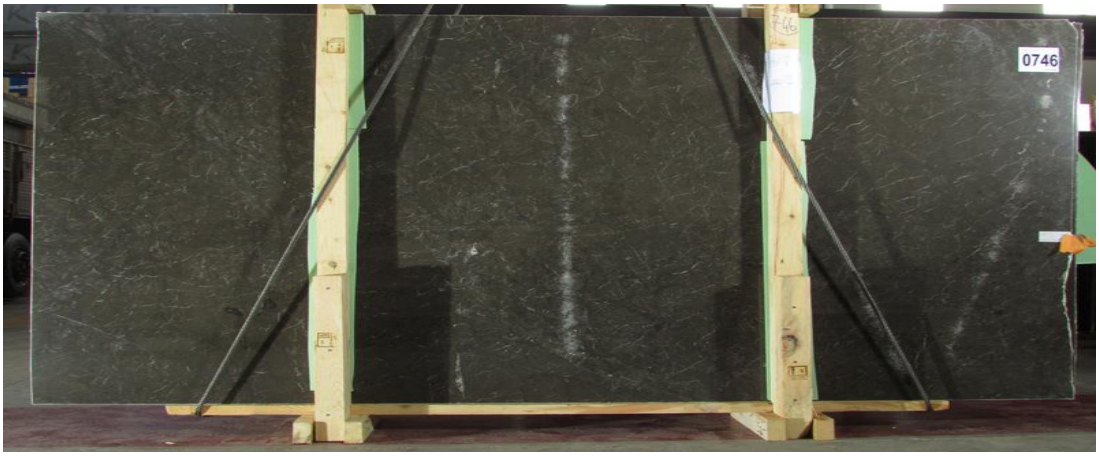


Figure 5.2 A two dimensional panel

In addition to the two dimensional data L_h and W_h, the panels are specified by the cost data C_h. C_h is the cost of obtaining a single panel of type h and is a function of its area $L_h \times W_h$ and its quality level. A larger area panel may have lower cost if it has lower quality grade.

The small objects, so called items, are obtained from the panels of limited availability. All those small items have customers that ask several of them to form their final products.

Figure 5.3 shows a small cut item with specified length and width and the placement of those small items for the final product of floor coverings.

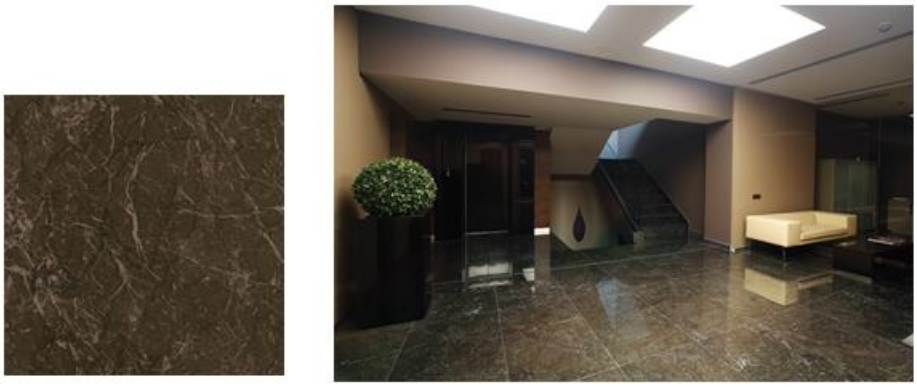


Figure 5.3 A small item and final products

Note from the above figure that the processed item is a small piece and many of these small pieces together form the final product like all floor covering of an international hotel. The customer, say hotel purchasing manager, had placed an offer for all the building floor coverings. The floor coverings are made up hundreds of small rectangular (or square) pieces, hence hundreds of units of small items are cut from the panels.

Each item has a specified demand projected from the final product specifications. As any order is for a specific customer, the amount cut above its requirement is forbidden, i.e., the total cut of any item can never be greater than its demand.

The revenues of items depend on the area of the small item, its quality and prestige of the customer.

For any specified period, the company wants to maximize the total profit (total revenue brought by the small items-total cost incurred by used panel) without exceeding the item demand and panel availability.

To solve the real life instances observed in the company, we propose Decomposition Heuristic 2 as it provides high quality solutions in reasonable solution times.

A real illustrative instance has 2 customer types. Customer 1 places an order to get 289 pieces which can be grouped into 16 different item types. To cut these pieces, the

company can use 12 different panel types. The order of customer 2 is for 482 items of 16 different item types. These pieces can be cut from 14 different panel types.

Table 5.1 gives the features of the customer orders. Table 5.2 tabulates the data for the available panel types.

Due to the quality restrictions of the orders, the first 12 panel types can be used for customer 1, while the rest is used for customer 2 demands.

The technological data (dimensions, demands, availabilities) are directly taken from the company. The cost data (item revenues and panel costs) are generated according to our data generation scheme discussed in Chapter 4.

Table 5.1 *Item Data of Customer Orders*

Customer 1 - Items				
Item Type	W	L	d	r
Customer 1 Item 1	59.5	79	14	39
Customer 1 Item 2	59.5	113	50	63
Customer 1 Item 3	59.5	108.7	7	60
Customer 1 Item 4	37.5	59.5	28	70
Customer 1 Item 5	35	30	10	54
Customer 1 Item 6	35	59.5	7	83
Customer 1 Item 7	30	79	30	77
Customer 1 Item 8	30	30	5	74
Customer 1 Item 9	30	59.5	7	64
Customer 1 Item 10	30	113	10	81
Customer 1 Item 11	30	37.5	4	60
Customer 1 Item 12	30	108.7	5	44
Customer 1 Item 13	20	30	50	37
Customer 1 Item 14	20	59.6	30	69
Customer 1 Item 15	20	30	4	27
Customer 1 Item 16	20	59.5	28	88

Customer 2 – Items				
Item Type	W	L	d	r
Customer 2 Item 1	68.5	99	14	81
Customer 2 Item 2	68.5	90	7	48
Customer 2 Item 3	60	61.8	9	88
Customer 2 Item 4	58.3	75	120	16
Customer 2 Item 5	58.3	74	197	91
Customer 2 Item 6	58.3	98	24	85
Customer 2 Item 7	42.6	59.5	14	10
Customer 2 Item 8	30	99	14	73
Customer 2 Item 9	30	87	8	25
Customer 2 Item 10	30	83	8	56
Customer 2 Item 11	24.3	30	7	44
Customer 2 Item 12	24.3	59.5	28	87
Customer 2 Item 13	24.3	87	8	18
Customer 2 Item 14	24.3	83	8	14
Customer 2 Item 15	20	87	8	76
Customer 2 Item 16	20	83	8	93

Table 5.2 *Panel Data*

Panels				
Panel Type	W	L	#	C
1	130	205	2	111
2	135	165	1	176
3	135	210	8	199
4	135	250	4	210
5	140	270	1	215
6	140	280	5	269
7	150	270	1	266
8	150	285	1	346
9	155	280	3	307
10	155	290	10	373
11	170	240	1	522
12	170	240	14	533
13	120	203	40	124
14	108	171	52	161
15	138	157	56	160
16	140	242	57	195
17	190	240	58	331
18	127	234	58	334
19	124	167	64	300
20	116	277	64	301
21	160	282	65	312
22	147	230	66	439
23	100	150	68	387
24	129	182	68	503
25	157	208	72	590
26	150	214	74	552

The decomposition heuristic 2 gives the following feasible assignments to each individual panel.

The assignments of customer 1 orders are given in the following page.

Panel Type 2 Assignments

Panel 1

Level 1:	Item 4	Item 4	Item 4	Item 4
Level 2:	Item 4	Item 4	Item 4	Item 4
Level 3:	Item 7	Item 7	Item 7	
Level 4:	Item 7	Item 7	Item 7	

Panel 2

Level 1:	Item 7	Item 7
Level 2:	Item 10	Item 12
Level 3:	Item 10	Item 12
Level 4:	Item 12	Item 12

Panel 3

Level 1:	Item 2	Item 2
Level 2:	Item 2	Item 2

Panel 4

Level 1:	Item 2	Item 2
Level 2:	Item 2	Item 2

Panel Type 3 Assignments

Panel 1

Level 1:	Item 5	Item 5	Item 5	Item 5	Item 5	Item 6	
Level 2:	Item 13	Item 13	Item 13	Item 13	Item 13	Item 14	
Level 3:	Item 13	Item 13	Item 13	Item 13	Item 13	Item 13	Item 13
Level 4:	Item 14	Item 13	Item 13	Item 13	Item 13	Item 13	
Level 5:	Item 14	Item 13	Item 13	Item 13	Item 13	Item 13	
Level 6:	Item 15	Item 14	Item 14	Item 14			

Panel 2

Level 1:	Item 6	Item 6	Item 6	
Level 2:	Item 13	Item 14	Item 14	Item 14
Level 3:	Item 14	Item 14	Item 14	
Level 4:	Item 14	Item 14	Item 14	
Level 5:	Item 15	Item 14	Item 14	Item 14
Level 6:	Item 15	Item 14	Item 14	Item 14

Panel 3

Level 1:	Item 6	Item 6	Item 6
Level 2:	Item 7	Item 9	Item 9

Level 3: Item 7 Item 9 Item 14
Level 4: Item 14 Item 14 Item 14
Level 5: Item 14 Item 14 Item 14

Panel 4
Level 1: Item 4 Item 4 Item 7
Level 2: Item 4 Item 4 Item 7
Level 3: Item 9 Item 7 Item 9
Level 4: Item 9 Item 7 Item 9

Panel 5
Level 1: Item 4 Item 4 Item 7
Level 2: Item 4 Item 4 Item 7
Level 3: Item 10 Item 7
Level 4: Item 10 Item 7

Panel 6
Level 1: Item 4 Item 4 Item 7
Level 2: Item 4 Item 4 Item 7
Level 3: Item 10 Item 7
Level 4: Item 10 Item 7

Panel 7
Level 1: Item 4 Item 4 Item 7
Level 2: Item 4 Item 4 Item 7
Level 3: Item 7 Item 10
Level 4: Item 7 Item 10

Panel 8
Level 1: Item 4 Item 4 Item 7
Level 2: Item 4 Item 4 Item 7
Level 3: Item 7 Item 10
Level 4: Item 7 Item 10

Panel Type 4 Assignments

Panel 1
Level 1: Item 1 Item 1 Item 2
Level 2: Item 1 Item 1 Item 2

Panel Type 5 Assignments

Panel 1

Level 1:	Item 1	Item 1	Item 3
Level 2:	Item 1	Item 1	Item 3

Panel Type 6 Assignments

Panel 1

Level 1:	Item 5	Item 5	Item 5	Item 5	Item 5
Level 2:	Item 13	Item 13	Item 13	Item 13	Item 13
Level 3:	Item 13	Item 13	Item 13	Item 16	
Level 4:	Item 14	Item 13	Item 13	Item 13	
Level 5:	Item 16	Item 13	Item 13	Item 13	
Level 6:	Item 16	Item 13	Item 13	Item 13	

Panel Type 9 Assignments

Panel 1

Level 1:	Item 3	Item 1	Item 1
Level 2:	Item 3	Item 1	Item 1
Level 3:	Item 12		

Panel Type 12 Assignments

Panel 1

Level 1:	Item 8	Item 8	Item 8	Item 8	Item 8	Item 11
Level 2:	Item 13	Item 13	Item 16	Item 16		
Level 3:	Item 13	Item 13	Item 16	Item 16		
Level 4:	Item 15	Item 13	Item 16	Item 16		
Level 5:	Item 16	Item 16	Item 16			
Level 6:	Item 16	Item 16	Item 16			

Panel 2

Level 1:	Item 11	Item 11	Item 11	Item 13	Item 16
Level 2:	Item 13	Item 13	Item 16	Item 16	
Level 3:	Item 13	Item 13	Item 16	Item 16	
Level 4:	Item 14	Item 16	Item 16		
Level 5:	Item 16	Item 16	Item 16		
Level 6:	Item 16	Item 16	Item 16		

The assignments of customer 2 orders are given below.

Panel Type 13 Assignments

Panel 1

Level 1:	Item 10	Item 12	Item 12	
Level 2:	Item 11	Item 11	Item 12	Item 16
Level 3:	Item 12	Item 11	Item 11	Item 16
Level 4:	Item 15	Item 16		
Level 5:	Item 16	Item 16		

Panel 2

Level 1:	Item 10	Item 12	Item 12	
Level 2:	Item 11	Item 11	Item 12	Item 12
Level 3:	Item 12	Item 12	Item 12	
Level 4:	Item 15	Item 16		
Level 5:	Item 16	Item 16		

Panel 3

Level 1:	Item 11	Item 12	Item 12
Level 2:	Item 12	Item 12	Item 12
Level 3:	Item 13	Item 12	
Level 4:	Item 14	Item 12	Item 12
Level 5:	Item 15	Item 15	

Panel 4

Level 1:	Item 8	Item 12	
Level 2:	Item 10	Item 12	Item 12
Level 3:	Item 12	Item 12	Item 12
Level 4:	Item 14	Item 12	Item 12

Panel 5

Level 1:	Item 8	Item 8
Level 2:	Item 10	Item 15
Level 3:	Item 12	Item 15
Level 4:	Item 15	Item 15

Panel 6

Level 1:	Item 3	Item 3	Item 5
Level 2:	Item 8	Item 8	
Level 3:	Item 10	Item 8	

Panel 7
Level 1: Item 3 Item 3 Item 5
Level 2: Item 8 Item 8
Level 3: Item 10 Item 8

Panel 8
Level 1: Item 3 Item 3 Item 5
Level 2: Item 8 Item 8
Level 3: Item 10 Item 8

Panel 9
Level 1: Item 3 Item 3 Item 5
Level 2: Item 9 Item 8
Level 3: Item 10 Item 8

Panel 10
Level 1: Item 3 Item 5 Item 7
Level 2: Item 5 Item 5

Panels 11-34
Level 1: Item 5 Item 5
Level 2: Item 6 Item 5

Panels 35-40
Level 1: Item 4 Item 5
Level 2: Item 5 Item 5

Panel Type 15 Assignments

Panels 1-25
Level 1: Item 5 Item 5
Level 2: Item 5 Item 5

Panels 26-32
Level 1: Item 1
Level 2: Item 1

Panel Type 16 Assignments

Panel 1
Level 1: Item 2 Item 2 Item 7
Level 2: Item 2 Item 2 Item 7

Panel 2
Level 1: Item 2 Item 2 Item 7
Level 2: Item 9 Item 9
Level 3: Item 9 Item 9

Panel 3
Level 1: Item 9 Item 13
Level 2: Item 9 Item 9
Level 3: Item 13 Item 13
Level 4: Item 13 Item 13
Level 5: Item 14 Item 13

For customer 1, a total of 18 panels are used to cut 244 items. The objective function value of the solution is calculated as:

Total cost of panels used = 3580

Total revenue gained by items cut = 15,309

Total profit = 15,309 – 3580 = 11,729

For customer 2, a total of 75 panels are used to cut 387 items. The objective function value of the solution is calculated as:

Total cost of panels used = 10,665

Total revenue gained by items cut = 28,251

Total profit = 28,251– 10,665 = 17,586

CHAPTER 6

CONCLUSIONS

In this study, we consider a two stage two dimensional cutting stock problem with multiple stock sizes. We consider both rotatable and non-rotatable items of specified demand.

We develop two mathematical models one of which is modified from a similar study in the literature. We give some optimality conditions and valid inequalities to enhance the efficiency of the models.

We also propose decomposition based heuristic procedures that decompose the problem into subproblems according to the panel types and individual panels and solve the subproblems by the best mathematical model we propose.

The results of our computational study have revealed that the model that we develop from scratch outperforms the one extended from the literature. We could find optimal solutions to the problem of size 20 items and 3 panels in two hours. We observe that the number of panel types, number of panels, number of item types and the demand figures are significant parameters that affect the speed of the achieving optimal solutions.

We find that our decomposition based heuristics could handle large sized instances with up to 50 items and 3 panels in two hours and produce solutions that have profits that are very close to the optimal values.

Future research may consider the development of optimization algorithms that make more efficient use of decomposition idea, like Benders' decomposition methods.

The multi criteria extensions of our model may also be worth-studying. We consider a linear combination of two performance measures: total revenue generated by the customer orders and total cost incurred by the panel costs. A multi criteria study might look for the trade-offs between total revenue and total cost or number of panels used.

REFERENCES

Alvarez-Valdes, R., Parajon, A., & Tamarit, J. M. (2002). A computational study of LP-based heuristic algorithms for two-dimensional guillotine cutting stock problems. *OR Spectrum*, *24*(2), 179-192.

Alvarez-Valdés, R., Parreño, F., & Tamarit, J. M. (2008). Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, *35*(4), 1065-1083.

Alvarez-Valdés, R., Parreño, F., & Tamarit, J. M. (2013). A GRASP/Path relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, *40*(12), 3081-3090.

Amossen, R. R., & Pisinger, D. (2010). Multi-dimensional bin packing problems with guillotine constraints. *Computers & Operations Research*, *37*(11), 1999-2006.

Bekrar, A., Kacem, I., & Chu, C., (2007). A comparative study of exact algorithms for the two dimensional strip packing problem. *J. Ind. Syst. Eng.*, **1**(2), 151–170.

Boschetti, M. A., & Montaletti, L. (2010). An exact algorithm for the two-dimensional strip-packing problem. *Operations Research*, *58*(6), 1774-1791.

Cintra, G. F., Miyazawa, F. K., Wakabayashi, Y., & Xavier, E. C. (2008). Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, *191*(1), 61-85.

Clautiaux, F., Jouglet, A., Carlier, J., & Moukrim, A. (2008). A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, *35*(3), 944-959.

Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. *Handbook of combinatorial optimization*, 455-531.

De Carvalho, J. V. (2002). LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, *141*(2), 253-273.

Delorme, M., Iori, M., & Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, *255*(1), 1-20.

Dolatabadi, M., Lodi, A., & Monaci, M. (2012). Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research*, 39(1), 48-53.

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

Fekete, S. P., Schepers, J., & Van der Veen, J. C. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, *55*(3), 569-587.

Fleszar, K. (2016). An exact algorithm for the two-dimensional stage-unrestricted guillotine cutting/packing decision problem. *INFORMS Journal on Computing*, *28*(4), 703-720.

Furini, F., & Malaguti, E. (2013). Models for the two-dimensional two-stage cutting stock problem with multiple stock size. *Computers & Operations Research*, *40*(8), 1953-1962.

Furini, F., Malaguti, E., Durán, R. M., Persiani, A., & Toth, P. (2012). A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, *218*(1), 251-260.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations research*, *9*(6), 849-859.

Gilmore, P. C., & Gomory, R. E. (1963). A linear programming approach to the cutting stock problem—Part II. *Operations research*, *11*(6), 863-888.

Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., P., Gottwald, R. L., ... & Müller, B. (2018). The SCIP optimization suite 6.0.

Hong, S., Zhang, D., Lau, H. C., Zeng, X., & Si, Y. W. (2014). A hybrid heuristic algorithm for the 2D variable-sized bin packing problem. *European Journal of Operational Research*, *238*(1), 95-103.

Lodi, A., Martello, S., & Vigo, D. (2004). Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization*, *8*(3), 363-379.

Macedo, R., Alves, C., & De Carvalho, J. V. (2010). Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, *37*(6), 991-1001.

Margot, F. (2010). Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008* (pp. 647-686). Springer, Berlin, Heidelberg.

Martello, S., & Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management science*, *44*(3), 388-399.

Martinovic, J., Scheithauer, G., & de Carvalho, J. V. (2018). A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems. *European Journal of Operational Research*, *266*(2), 458-471.

Mittelman, H. D. (2018). Latest Benchmark Results. *INFORMS Annual Conference*, Phoenix, AZ, USA.

Pisinger, D., & Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, *2*(2), 154-167.

Pisinger, D., & Sigurd, M. (2007). Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, *19*(1), 36-51.

Riehme, J., Scheithauer, G., & Terno, J. (1996). The solution of two-stage guillotine cutting stock problems having extremely varying order demands. *European Journal of Operational Research*, *91*(3), 543-552.

Silva, E., Alvelos, F., & de Carvalho, J. V. (2010). An integer programming model for two-and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research*, *205*(3), 699-708.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational research*, *183*(3), 1109-1130.

Wei, L., Oon, W. C., Zhu, W., & Lim, A. (2013). A goal-driven approach to the 2D bin packing and variable-sized bin packing problems. *European Journal of Operational Research*, *224*(1), 110-121.