공학석사학위논문

Integer programming models and exact methods for the two-dimensional two-staged knapsack problem

2차원 2단계 배낭문제에 대한 정수계획모형 및 최적해법

2021 년 2 월

서울대학교 대학원

산업공학과

강 수 호

Integer programming models and exact methods for the two-dimensional two-staged knapsack problem

2차원 2단계 배낭문제에 대한 정수계획모형 및 최적해법

지도교수 이 경 식

이 논문을 공학석사 학위논문으로 제출함

2021 년 1 월

서울대학교 대학원

산업공학과

강 수 호

강수호의 공학석사 학위논문을 인준함

2021 년 1 월

위 원 장 홍 성 필 (인)

부위원장 이 경 식 (인)

위 원 홍 유 석 (인)

Abstract

Integer programming models and exact methods for the two-dimensional two-staged knapsack problem

Suho Kang
Department of Industrial Engineering
The Graduate School
Seoul National University

In this thesis, we study integer programming models and exact algorithms for the two-dimensional two-staged knapsack problems, which maximizes the profit by cutting a single rectangular plate into smaller rectangular items by two-staged guillotine cuts. We first introduce various integer programming models, including the strip-packing model, the staged-pattern model, the level-packing model, and the arc-flow model for the problem. Then, a hierarchy of the strength of the upper bounds provided by the LP-relaxations of the models is established based on theoretical analysis. We also show that there exists a polynomial-size model that has not been proven yet as far as we know. Exact methods, including branch-and-price algorithms using the strip-packing model and the staged-pattern model, are also devised. Computational experiments on benchmark instances are conducted to examine the strength of upper bounds obtained by the LP-relaxations of the models and evaluate the performance of exact methods. The results show that the staged-pattern model gives a

competitive theoretical and computational performance.

Keywords: Integer Programming, Two-dimensional two-staged knapsack problem, Dantzig-Wolfe decomposition, Branch-and-price algorithm

Student Number: 2019-26644

Contents

List of Tables

List of Figures

Chapter 1

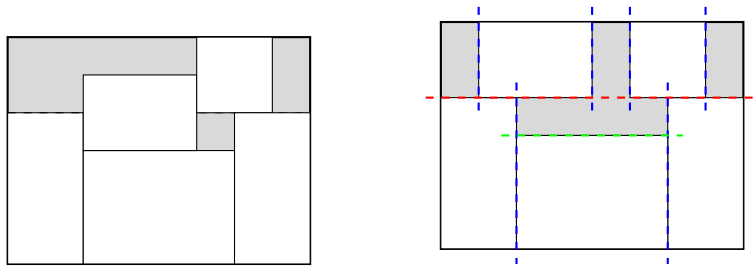
Introduction

The two-dimensional two-staged knapsack problem (2TDK) [18], which is also known as the two-dimensional two-staged cutting problem [2], maximizes a profit by cutting smaller rectangular items on a single rectangular plate without conflicts. It is a variant of the two-dimensional knapsack problem [5] with additional constraints for cutting. Conventionally, a two-dimensional knapsack problem assumes that the items have to be cut in the orthogonal style: if an item is cut from the large plate, the vertical side and the horizontal side of the item must be parallel to the vertical side and the horizontal side of the large plate, respectively. There is an additional constraint related to the guillotine cut: the plate must divide into two rectangles when being cut. Furthermore, two-staged guillotine cuts refer to the constraint that the first-stage cuts and the second-stage cuts are orthogonal. For example, in this thesis, vertical cuts can be made after all horizontal cuts are carried out.

We refer to the separate plates produced after the first-stage cuts as strips or levels. Among items in each strip, there is always the highest one. This item, which we refer to as a strip-defining item in the thesis, determines the height of the strip. There is no need for strips to be higher than their strip-defining items. Therefore, we assume that the height of the strip-defining item of each strip determines the height

of the strip. Once a strip is fixed, one should cut it vertically to produce items. We consider the inexact case, i.e., allowing trimming after the second-stage of guillotine cuts. These assumptions are practically meaningful in various industries that use sheets of material such as glass, paper, wood, and metal [10].

A simple example of two-staged guillotine cutting is represented in Figure 1.1. Both parts of Figure 1.1 are feasible two-dimensional orthogonal cuttings. On the other hand, only the right part of Figure 1.1 is a feasible two-staged two-dimensional guillotine cutting. The first-stage cuts, the second-stage cuts, and trimming are illustrated as red, blue, and green dashed lines, respectively.



(a) An infeasible case. (b) A feasible case.

Figure 1.1: An example of two-staged two-dimensional guillotine cutting.

1.1 Problem Description

Formally, the problem is defined as the following. Consider a large rectangular plate S with height H and width W. There are n different types of small rectangular items. For any natural number m, let $I_m = \{1, \ldots, m\}$. Then, for each $i \in I_n = \{1, \ldots, n\}$, an item of type i has height h_i which is less than H, a width w_i which is less than W, a profit p_i, and an upper bound (i.e., demand) d_i which denotes the maximum number of usage of item type i. We assume that all the values are integers.

If some d_i are lower than $\lfloor \frac{H}{h_i} \rfloor \lfloor \frac{W}{w_i} \rfloor$, we call the problem is constrained. If all components of d are not smaller than $\lfloor \frac{H}{h_i} \rfloor \lfloor \frac{W}{w_i} \rfloor$, we call the problem is unconstrained [13]. Also, one should separate smaller rectangles from a larger one by two-stage guillotine cuts. Our objective is to maximize the profit that can be obtained from S. For simplicity, we assume that the first-stage cut is performed horizontally, and the second-stage cut is performed vertically. Furthermore, we do not allow rotations of smaller rectangles and assume that $H \geq h_1 \geq h_2 \geq \ldots \geq h_n$. We denote an instance of the 2TDK by a tuple (n, H, W, h, w, d, p) in this thesis.

If the problem is unconstrained, each strip used in the optimal solution has the highest profit among strips with the same height. Therefore, it can be reduced to a two-stage knapsack problem: determine the most valuable strips defined by each item, and then pack them along the vertical side within the overall height of H.

In the case of the constrained 2TDK, there exists a simple reduction from the 3-PARTITION problem to the problem. The 3-PARTITION problem, which is well-known to be NP-hard in the strong sense [11], consists of a positive integer m, a set $S = I_{3m}$, a bound $B \in \mathbf{Z}_+$, and a value $k_i \in \mathbf{Z}_+$ for each $i \in S$ such that $B/4 < k_i < B/2$ and $\sum_{i \in S} k_i = mB$. The problem is to determine whether there exists a partition of S composed of m disjoint sets $S_1, \ldots, S_m$ such that for $1 \leq j \leq m$, $\sum_{i \in S_j} k_i = B$ and $|S_j| = 3$.

Proposition 1.1. *The constrained 2TDK is NP-hard in the strong sense.*

Proof. For a given instance of the 3-PARTITION problem, there exists a partition of S whose each subset has three components and their sum is B, if and only if the 2TDK instance $(3m, m, B, h, w, d, p)$ with $(w_i, h_i, p_i, d_i) = (k_i, 1, 1, 1)$ for all $i \in I_{3m}$ has an optimal value of $3m$. □

1.2 Literature Review

The 2TDK has been researched since Gilmore and Gomory [12] suggested the pattern-based formulations for two-dimensional cutting stock problems (2D2SP) [25]. To be more specific, the 2TDK emerges as the slave (pricing) problem of the 2D2SP, implying that they share a similar structure [10]. In this respect, solving the 2TDK is a relevant issue of solving the 2D2SP, while extending the state-of-the-art from the 2D2SP to the 2TDK has been scarce. With this necessity, we start this section by reviewing studies on the 2D2SP.

After the seminar work of Gilmore and Gomory [12], a few research suggested various integer linear programming models for the 2D2SP. Macedo *et al.* [20], for example, introduced the so-called arc-flow formulation, which is originated from the one-dimensional cutting stock problem. The arc-flow model requires some graphs along the vertical or horizontal side of the large plate and represents the usage of items as a feasible flow. Mrad *et al.* [22] proposed the pattern-based formulation for the 2D2SP, which includes both width patterns and height patterns. Silva *et al.* [23] suggests the one-cut model, and the close relationship between the one-cut model and the arc-flow model is well described in [21]. Lastly, for a contemporary review of mathematical models for the 2D2SP, we refer readers to Kwon *et al.* [17], which also settled the theoretical hierarchy between the LP-relaxation values of previous models.

However, unlike the 2D2SP, few models have been proposed for the 2TDK. Gilmore and Gomory [12] developed a strip packing formulation, while two integer linear programming models based on a concept of level packing are devised by Lodi and Monaci [18]. Lodi and Monaci [18] solved their models by adding valid

inequalities to break symmetry and compared their LP-relaxation values with the strip packing model computationally, which was solved by the column generation technique. Yet, the analysis of its computational results was insufficient, and the theoretical relationship between a strip packing formulation and a level packing model has been hardly addressed.

Several exact methods have been proposed to solve problems to optimality. Hifi [13] categorized two-dimensional knapsack problems and solved the unconstrained 2TDK by dynamic programming. Belov and Scheithauer [2] implemented a branch-and-cut-and-price algorithm using Chvatal-Gomory and Gomory mixed-integer cuts, based on the strip packing formulation. The work of Belov and Scheithauer [2] seems effective in the sense that its algorithm could guarantee optimality within considerable time, but it failed to solve all instances proposed by Hifi and Roucaircol [15] to optimality. In addition, even though previous research tested their models computationally, more elaborate experiments are required for investigating the pros and cons of each method.

In the aspect of heuristic algorithms for the 2TDK, Hifi and M'Hallah [14] suggested strip generation algorithms based on the greedy algorithm and the hill-climbing algorithm. Conducting computational experiments to evaluate the effectiveness of their models, it verified that its algorithm could find near-optimal solutions in a relatively short time. Alvarez-Valdes *et al.* [1] developed heuristic algorithms with bottom-left procedures and path relinking methods. It was a time-efficient algorithm, even finding a better solution to the instance that the branch-and-cut-and-price algorithm failed to find the optimal solution within a relatively longer time [2]. However, no theoretical guarantees have been made for the effectiveness of

heuristics.

Despite sharing the two-staged guillotine cutting constraint, extending models from the 2D2SP to the 2TDK have not been proposed much yet. Even for the existing models and methods, their strengths and weakness have not been fully investigated both computationally and theoretically. We, therefore, have tried to overcome the limitation and fill the unanswered gap from previous studies.

1.3 Contributions

In this study, questions which have been discussed insufficiently from the perspective of the 2TDK are analyzed thoroughly. Our contributions can be summarized as follows:

(a) We extend some existing models for the 2D2SP to the 2TDK and add a set of valid inequalities to the level-packing model proposed by Lodi and Monaci [18]. The added inequalities enhance the LP-relaxation value and ease analyzing the relationship between other models.

(b) In addition, the LP-relaxation values of models are analyzed theoretically. We derive that the LP-relaxation values of some models satisfy a specific inequality and prove that there is a tight example.

(c) Furthermore, we propose a novel polynomial-size formulation for the first time, using a result provided by Eisenbrand and Shmonin [8].

(d) Finally, we propose exact algorithms for the 2TDK. We construct branch-and-price algorithms for the pattern-based models and present a branch-and-cut algorithm for the modified level-packing model. Various computation experiments are conducted to analyze the characteristics of each model in their real usage.

1.4 Organization of the Thesis

This thesis is organized as follows. In Chapter 2, we shortly derive the complexity of solving the 2TDK and introduce various integer linear programming models for the 2TDK. The relationship between the LP-relaxation values among them is discussed in Chapter 3. We also suggest a polynomial-size model for the 2TDK for the first time. Then, We propose exact algorithms for some models in Chapter 4. In Chapter 5, we compare the LP-relaxation values computationally and analyze their performance of solving problems to optimality. This thesis ends up with the main conclusion in Chapter 6.

Chapter 2

Integer Programming Models for the 2TDK

2.1 Pattern-based Model

We start this section by introducing the staged-pattern model from Mrad *et al.* [22] and Kwon *et al.* [17]. The main idea of this formulation is to utilize the concept of a width pattern and a height pattern. We represent a width pattern q as an n-dimensional nonnegative integer vector $a_q = (a_{q1}, \ldots, a_{qn})$, where a_{qi} is the number of pieces of ith item type for each $i \in I_n$ in a width pattern q. For a given n-dimensional vector u, we define $P_W(u)$ as a set of width patterns q satisfying the following:

$$P_W(u) = \{q | \sum_{i \in I_n} w_i a_{qi} \leq W, a_q \leq u\}.$$

Likewise, we represent a height pattern r as an n-dimensional nonnegative integer vector $b_r = (b_{r1}, \ldots, b_{rn})$ where b_{ri} is the number of pieces of ith item type used for each $i \in I_n$ in a height pattern r. For a given n-dimensional vector v, we define $P_H(v)$ as a set of height patterns r satisfying the following:

$$P_H(v) = \{r | \sum_{i \in I_n} h_i b_{ri} \leq H, b_r \leq v\}.$$

As a special case, we define $P_W(\infty)$ and $P_H(\infty)$ as the following:

$$P_W(\infty) = \{q | \sum_{i \in I_n} w_i a_{qi} \leq W\},$$

and

$$P_H(\infty) = \{r | \sum_{i \in I_n} h_i b_{ri} \leq H\}.$$

Note that $P_W(\infty)$ and $P_H(\infty)$ corresponds to the unconstrained case. In addition, for a given n-dimensional vector u and a width pattern $q \in P_W(u)$, let $t(q)$ denote the minimum index which is in a support of a_q. For a strip corresponding to the width pattern q, the strip-defining item is an item of type $t(q)$. Also, the height of the strip corresponds to $h_{t(q)}$.

In the example of Figure 2.1 and Figure 2.2, the left part of Figure 2.1 shows that two first-stage cuts divide the large plate into three strips. The right part of Figure 2.1 illustrates second-stage cuts and trimming to each strip. Then, the first-stage cuts are equivalent to a single height pattern, and second-stage cuts including trimming are equivalent to a single width pattern. The result of cuts is the same as Figure 2.2.

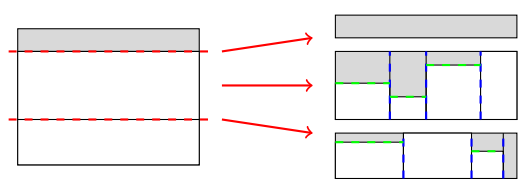


Figure 2.1: An illustration of two-stage guillotine cutting.

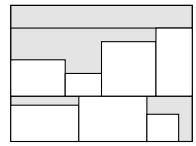


Figure 2.2: An illustration of the large plate.

Utilizing these concepts, we represent the staged-pattern model for the 2TDK as follows:

$$
\begin{aligned}
\mathrm{SM}(u,v): \quad \text{maximize} \quad & \sum_{q\in P_W(u)} \sum_{i\in I_n} p_i a_{qi} x_q \\
\text{subject to} \quad & \sum_{q\in P_W(u)} a_{qi} x_q \le d_i, \quad \forall i \in I_n, && (2.1)\\
& \sum_{r\in P_H(v)} b_{ri} y_r \ge \sum_{q\in P_W(u), t(q)=i} x_q, \quad \forall i \in I_n, && (2.2)\\
& \sum_{r\in P_H(v)} y_r \le 1, && (2.3)\\
& x_q \in \mathbf{Z}_+, \quad \forall q \in P_W(u), \\
& y_r \in \mathbf{Z}_+, \quad \forall r \in P_H(v). && (2.4)
\end{aligned}
$$

A decision variable x_q denotes how much width pattern q has been used, and a decision variable y_r represents the amount of height pattern r having been used. Constraints (2.1) limit the maximum usage of each item type to its demand, and a constraint (2.2) indicates that usage of a width pattern is only available when a chosen height pattern allows the usage of the corresponding strip-defining item.

On the other hand, the strip packing model originated from Gilmore and Gomory [12] only exploits the concept of the width pattern. We propose the mathematical

formulation of the strip packing model as follows:

$$
\begin{aligned}
\mathrm{PM}(u): \quad \text{maximize} \quad & \sum_{q \in P_W(u)} \sum_{i \in I_n} p_i a_{qi} x_q \\
\text{subject to} \quad & \sum_{q \in P_W(u)} a_{qi} x_q \leq d_i, \quad \forall i \in I_n, && (2.5) \\
& \sum_{q \in P_W(u)} h_{t(q)} x_q \leq H, && (2.6) \\
& x_q \in \mathbf{Z}_+, \quad \forall q \in P_W(u).
\end{aligned}
$$

A decision variable x_q has the same meaning as x_q in $\mathrm{SM}(u, v)$. Constraints (2.5) describe the demand constraint, and a constraint (2.6) indicates that the total height cannot exceed H. Note that $\mathrm{PM}(u)$ becomes a valid formulation for the 2TDK when $u \geq d$. Similarly, $\mathrm{SM}(u, v)$ becomes a valid formulation when $u \geq d$ and $v \geq d$. Besides, because the number of components is exponentially many in a pattern set $P_W(u)$ (resp. $P_H(v)$) as $u \geq d$ (resp. $v \geq d$), $\mathrm{PM}(u)$ and $\mathrm{SM}(u, v)$ are not polynomial-size formulations.

Then, for a given 2TDK instance (n, H, W, h, w, d, p), let z^* be the optimal objective value of any valid formulation for the 2TDK and z_{LP}^{model} be the optimal objective value of the LP-relaxation of the corresponding *model* of the 2TDK. For example, $z_{\mathrm{LP}}^{\mathrm{PM}(u)}$ is the optimal objective value of the LP-relaxation value of $\mathrm{PM}(u)$. With these symbols, the following proposition hold:

Proposition 2.1. *$z^* \leq z_{LP}^{SM(d,d)} \leq z_{LP}^{PM(d)}$ and $z_{LP}^{SM(d,d)} \leq z_{LP}^{SM(\infty,\infty)}$.*

Proof. For any feasible solution (x, y) in the LP-relaxation of $\mathrm{SM}(u, v)$, x is a feasible solution in the LP-relaxation of $\mathrm{PM}(u)$. Also, since $P_W(d) \subset P_W(\infty)$ and $P_H(d) \subset$

$P_H(\infty)$, $z_{\mathrm{LP}}^{\mathrm{SM}(d,d)} \leq z_{\mathrm{LP}}^{\mathrm{SM}(\infty,\infty)}$ holds. □

Interestingly, $z_{\mathrm{LP}}^{\mathrm{SM}(\infty,\infty)}$ is not always less than $z_{\mathrm{LP}}^{\mathrm{PM}(d)}$. We verify this through some benchmark instances in Chapter 5, which refute $z_{\mathrm{LP}}^{\mathrm{SM}(\infty,\infty)} \leq z_{\mathrm{LP}}^{\mathrm{PM}(d)}$.

Proposition 2.2. *SM(d, d) is still a valid formulation when constraints (2.4) are relaxed.*

Proof. Let a feasible solution (x^*, y^*) of $\mathrm{SM}(d, d)$ with constraints (2.4) relaxed. We will prove that x^* is a feasible solution of $\mathrm{PM}(d)$. Note that the following holds:

$$\sum_{q \in P_W(d)} h_{t(q)} x_q^* = \sum_{iinI_n} h_i \Big(\sum_{q \in P_W(d), t(q)=i} x_q^* \Big) \leq \sum_{i \in I_n} \sum_{r \in P_H(d)} h_i b_{ri} y_r^*.$$

Since $\sum_{i \in I_n} h_i b_{ri} \leq H$ for all $r \in P_H(d)$ and $\sum_{r \in P_H(d)} y_r^* \leq 1$, the next inequality satisfies:

$$\sum_{i \in I_n} \sum_{r \in P_H(d)} h_i b_{ri} y_r^* \leq \sum_{r \in P_H(d)} y_r^* \Big(\sum_{i \in I_n} h_i b_{ri} \Big) \leq \sum_{r \in P_H(d)} H y_r^* \leq H.$$

Therefore, x^* satisfies all constraints in $\mathrm{PM}(d)$.

Then, for any feasible solution x^* in $\mathrm{PM}(d)$, $(b_1^*, \ldots, b_n^*)$ defined as follows is in $P_H(d)$:

$$b_i = \sum_{q \in P_W(u), t(q)=i} x_q^*, \quad \forall i \in I_n.$$

Let $b_{r^*} = (b_1^*, \ldots, b_n^*)$. Then, let $y_{r^*} = 1$ and $y_r = 0$ for other $r \in P_H(d)$. It is clear that (x^*, y) is a feasible solution of $\mathrm{SM}(d, d)$ with constraints (2.4) relaxed. □

In the aspect of simplicity, when it comes to height patterns, we can corpo-

rate item types with the same height into a single item type. To elaborate, let $\{h_{(1)}, \ldots, h_{(m)}\}$ be the minimal representation of a height set $\{h_1, \ldots, h_n\}$. Then, redefine a height pattern γ as a height pattern vector $(b_{\gamma 1}, \ldots, b_{\gamma m}) \in \mathbf{Z}_+^m$ satisfying $\sum_{i \in I_m} h_{(i)} b_{\gamma i} \leq H$ and $b_{\gamma i} \leq \sum_{j \in I_n, h_j = h_{(i)}} d_j$ for $i \in I_m$. Let the height pattern set be $P_H^2(d)$. Then, the similar staged-pattern model SM-HA is represented as following:

$$
\begin{aligned}
\text{SM-HA}: \quad \text{maximize} \quad & \sum_{q \in P_W(d)} \sum_{i \in I_n} p_i a_{qi} x_q \\
\text{subject to} \quad & \sum_{q \in P_W(d)} a_{qi} x_q \leq d_i, \quad \forall i \in I_n, && (2.7) \\
& \sum_{r \in P_H^2(d)} b_{ri} y_r \geq \sum_{q \in P_W(u), h_{t(q)} = h_{(i)}} x_q, \qquad \forall i \in I_m, && (2.8) \\
& \sum_{r \in P_H^2(d)} y_r \leq 1, && (2.9) \\
& x_q \in \mathbf{Z}_+, \quad \forall q \in P_W(d), \\
& y_r \in \mathbf{Z}_+, \quad \forall r \in P_H^2(d).
\end{aligned}
$$

Note that SM-HA is a valid formulation, and the following proposition holds:

Proposition 2.3. $z_{LP}^{SM(d,d)} \leq z_{LP}^{SM\text{-}HA} \leq z_{LP}^{PM(d)}$.

Proof. Any feasible solution of the LP-relaxation of SM(d, d) is a feasible solution to the LP-relaxation of SM-HA. Also, a constraint (2.9) forces x in the LP-relaxation of SM-HA to satisfy the constraint (2.6). Therefore, $z_{\text{LP}}^{\text{SM-HA}} \leq z_{\text{LP}}^{\text{PM}(d)}$ holds. □

2.2 Arc-flow Model

With some modifications of the network model suggested by Macedo *et al.* [20], similar networks can be designed for the 2TDK. For each instance of the 2TDK, the arc-flow model generates $n+1$ directed acyclic graphs: $G^0 = (V^0, A^0)$, $G^1 = (V^1, A^1)$, $\ldots$, $G^n = (V^n, A^n)$. A graph G^0 represents how much each item has been used as a strip-defining item, and the other graphs represent strips that are characterized by the item type of corresponding indices. Graph G^0 is given by the set of nodes $V^0 = \{0, 1, \ldots, H\}$ and the set of arcs $A^0 = \{(a, b, k) | 0 \leq a < b \leq H, b - a = h_k, k \in I_n\} \cup N_H$, where $N_H := \{(i, i+1, 0) | i \in \{0, \ldots, H-1\}\}$. Note that N_H represents empty space. Likewise, for $j \in I_n$, graph G^j is given by the set of nodes $V^j = \{0, \ldots, W\}$ and the set of arcs $A^j = \{(a, b, i) | 0 \leq a < b \leq W, b - a = w_i, i \in \{j, \ldots, n\}\} \cup N_W^j$, where $N_W^j := \{(i, i+1, 0) | i \in \{0, \ldots, W-1\}\}$. A profit of each arc (a, b, i) in G^j, which we denote by $\pi^j_{(a,b,i)}$, is p_i for $i \in I_n$. For the other arcs, they have zero profits. Decision variables are the amount of flows in each arc, and the overall mathematical representation of an arc-flow model for the 2TDK is as follows:

$$\text{AF}: \quad \text{maximize} \quad \sum_{j \in I_n} \sum_{(a,b,i) \in A^j} \pi^j_{(a,b,i)} x^j_{(a,b,i)} \tag{2.10}$$

$$\text{subject to} \quad \sum_{(a,b,i) \in A^0} x^0_{(a,b,i)} - \sum_{(b,c,k) \in A^0} x^0_{(b,c,k)} = \begin{cases} -1 & \text{if } b = 0 \\ 0 & \text{if } b = 1, \ldots, H-1 \\ 1 & \text{if } b = H \end{cases}, \tag{2.11}$$

$$\sum_{(c,c+h_j,k) \in A^0} x^0_{(c,c+h_j,k)} - z^j = 0, \quad \forall j \in I_n, \tag{2.12}$$

$$\sum_{(d,e,i) \in A^j} x^j_{(d,e,i)} - \sum_{(e,f,k) \in A^j} x^j_{(e,f,k)} = \begin{cases} -z^j & \text{if } e = 0 \\ 0 & \text{if } e = 1, \ldots, W-1 \\ z^j & \text{if } e = W \end{cases}, \quad \forall j \in I_n, \tag{2.13}$$

$$\sum_{j \in I_n} \sum_{(f,f+w_i,i) \in A^j} x^j_{(f,f+w_i,i)} \leq d_i, \quad \forall i \in I_n, \tag{2.14}$$

All variables are nonnegative integers.

The objective function (2.10) describes the overall profit. Constraints (2.11), (2.12), and (2.13) determine how much flow can run in each graph. Lastly, constraints (2.14) represent demand constraints. Note that the size of AF is pseudo-polynomial since it depends on numeric values of H and W. For some arc-reduction techniques and motivations, see Martinovic *et al.* [21].

2.3 Level Packing Model

Lodi and Monaci [18] regarded items of the same type as separate item types which all have a unit demand and share the same sizes and profit. If the demand for each item is a unit amount, all strip-defining items are characterized by their types. In this view, Lodi and Monaci [18] suggested two formulations which divide d_i items of i type into d_i different types for all $i \in \{1, \ldots, n\}$. These new d_i types of items share the same height, width, profit, and demand value as $(h_i, w_i, p_i, 1)$.

Then, each 2TDK instance $\mathbf{I} = (n, H, W, h, w, d, p)$ transforms into new instance $\hat{\mathbf{I}}$ with the number of item types $N = \sum_{i=1}^{n} d_i$. In detail, we define $\alpha_j = \sum_{i=1}^{j} d_i$ for any $j \in I_n$ and $\beta_k = \min\{t : 1 \leq t \leq n, \alpha_t \geq k\}$ for any $k \in I_N$. Then, we express the transformed instance $\hat{\mathbf{I}}$ as the following:

$$\hat{\mathbf{I}} = (N, H, W, \hat{h}, \hat{w}, \hat{d}, \hat{p}),$$

where $\hat{h}_k = h_{\beta_k}, \hat{w}_k = w_{\beta_k}, \hat{d}_k = 1,$ and $\hat{p}_k = p_{\beta_k}$ for any $k \in I_N$. The level-packing formulation LM that was proposed in Lodi and Monaci [18] is as follows:

$$\text{LM}: \quad \text{maximize} \quad \sum_{j=1}^{N} p_{\beta_k} \sum_{k=1}^{j} x_{jk} \tag{2.15}$$

$$\text{subject to} \quad \sum_{k=1}^{j} x_{jk} \leq 1, \quad \forall j \in \{1, \ldots, N\} \tag{2.16}$$

$$\sum_{j=k+1}^{N} w_{\beta_j} x_{jk} \leq (W - w_{\beta_k}) x_{kk}, \quad \forall k \in \{1, \ldots, N\} \tag{2.17}$$

$$\sum_{k=1}^{N} h_{\beta_k} x_{kk} \leq H, \tag{2.18}$$

$$x_{jk} \in \{0, 1\}, \quad \forall k \in \{1, \ldots, N\}, \quad \forall j \in \{k, \ldots, N\}. \tag{2.19}$$

In this formulation, a decision variable x_{jk} represents the usage of jth item in a strip that is defined by stip-defining item of kth original type.

Against previous research including Belov and Scheithauer [2], we claim that the above model LM is a pseudo-polynomial-size model since the number of variables depends on a numeric value of the sum of demands, not the input size of it. We also add a set of constraints (2.21) that restricts the formulation to properly allocate items in any fractional used strip. These inequalities not only improve the quality of the LP-relaxation value but also ease comparing the structure of the formulation to other models. This modified model is named as ML, stands for modified level packing. The mathematical formulation of ML is as follows:

$$\text{ML}: \quad \text{maximize} \quad (2.15)$$

$$\text{subject to} \quad (2.15), (2.16), (2.17), (2.18), \text{ and } (2.19) \tag{2.20}$$

$$x_{jk} \leq x_{kk}, \quad \forall k \in \{1, \ldots, N\}, \quad \forall j \in \{k+1, \ldots, N\}. \tag{2.21}$$

Note that each of the constraints (2.21) is a valid inequality that sets the strip-defining item to the most used item in its corresponding strip.

Chapter 3

Theoretical Analysis of Integer Programming Models

In Section 3.1 and 3.2, we focus on four different types of models: a strip packing model, a staged-pattern model, an arc-flow model, and a modified level packing model. We thoroughly analyze the hierarchical relationship between the LP-relaxation of each model. In Section 3.3, we propose the polynomial-size formulation for the first time.

3.1 Upper Bounds of AF and SM(∞, ∞)

We summarize the equivalence between the upper bounds provided by the LP-relaxation of AF and SM(∞, ∞) as the following proposition:

Proposition 3.1. $z_{LP}^{AF} = z_{LP}^{SM(\infty,\infty)}$

Proof. As all graphs in the arc-flow model are acyclic, any feasible flow in $G_0, \ldots, G_n$ is decomposed into the simple paths from the start node to the end node. Since a possible simple path in G_0 is corresponding to a height-pattern in $P_H(\infty)$ and a possible simple path in $G_1, \ldots, G_n$ is corresponding to a width-pattern in $P_W(\infty)$. In addition, each pattern corresponds to some simple paths of a corresponding graph. Therefore, solutions to the LP-relaxations of AF and SM(∞, ∞) are convertible. □

3.2 Upper Bounds of ML, PM(d), and SM(d, d)

To directly compare instance $\mathbf{I}$ and instance $\hat{\mathbf{I}}$, define a set of width pattern vectors Q_W and a set of height pattern vectors Q_H as follows:

$$Q_W = \{a_q \in \mathbf{Z}_+^n | q \in P_W(d)\} = \{a \in \mathbf{Z}_+^n | \sum_{i \in I_n} w_i a_i \leq W, a \leq d\},$$

and

$$Q_H = \{b_r \in \mathbf{Z}_+^n | r \in P_H(d)\} = \{b \in \mathbf{Z}_+^n | \sum_{i \in I_n} h_i b_i \leq H, b \leq d\}.$$

Also, define a set of width pattern vectors $\hat{Q}_W$ and a set of height pattern vectors $\hat{Q}_H$ in the aspect of instance $\hat{\mathbf{I}}$:

$$\hat{Q}_W = \{\hat{a} \in \mathbf{B}^N | \sum_{k \in I_N} w_{\beta_k} \hat{a}_k \leq W\} = \{\hat{a}^1, \ldots, \hat{a}^{M_W}\},$$

and

$$\hat{Q}_H = \{\hat{b} \in \mathbf{B}^N | \sum_{k \in I_N} h_{\beta_k} \hat{b}_k \leq H\} = \{\hat{b}^1, \ldots, \hat{b}^{M_H}\},$$

where M_W and M_H correspond to $|\hat{Q}_W|$ and $|\hat{Q}_H|$, respectively.

Note that an element of $\hat{Q}_W$ is an "extended version" of an element of Q_W. To elaborate their relationship, we define an onto function $f : \hat{Q}_W \to Q_W$ as follows:

$$f(\hat{a})_i = \sum_{j \in I_N, \beta_j = i} \hat{a}_j, \forall i \in I_n, \forall \hat{a} \in \hat{Q}_W.$$

For any $i \in I_{M_W}$, there exists a width pattern $q \in P_W(d)$ such that $f(\hat{a}^i) = a_q$, and let $\hat{c}_i$ be the value of $h_{t(q)}$. Indeed, $\hat{c}_i$ is the height of the strip-defining item with the respect of the instance $\hat{\mathbf{I}}$. Then, the following formulation PM2 is a valid formulation for the 2TDK:

$$\begin{aligned}
\text{PM2}: \quad \text{maximize} \quad & \sum_{i=1}^{M_W} \sum_{k=1}^{N} p_{\beta_k} \hat{a}_k^i x_i \\
\text{subject to} \quad & \sum_{i=1}^{M_W} \hat{a}_k^i x_i \leq 1, \quad \forall k \in I_N, \\
& \sum_{i=1}^{M_W} \hat{c}_i x_i \leq H, \\
& x \in B^{M_W}.
\end{aligned}$$

To help understand, define an instance $\mathbf{I}_1 = (2, 3, 3, h, w, d, p)$ with $h = (2, 1)$, $w = (2, 1)$, $d = (1, 2)$, and $p = (4, 1)$. Then, the corresponding $\hat{\mathbf{I}}_1$ is defined as $(3, 3, 3, \hat{h}, \hat{w}, \hat{d}, \hat{p})$ with $\hat{h} = (2, 1, 1)$, $\hat{w} = (2, 1, 1)$, $\hat{d} = (1, 1, 1)$, and $\hat{p} = (4, 1, 1)$. Then, Q_W and $\hat{Q}_W$ are as follows:

$$Q_W = \{(1, 0), (1, 1), (0, 1), (0, 2)\},$$

and

$$\hat{Q}_W = \{(1, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 0), (0, 0, 1), (0, 1, 1)\}.$$

Note that for any $j \in I_n$, if $(\hat{a}_1, \ldots, \hat{a}_{\alpha_{j-1}+1}, \ldots, \hat{a}_{\alpha_j}, \ldots, \hat{a}_N) \in \hat{Q}_W$, then so does $(\hat{a}_1, \ldots, \hat{a}^*_{\alpha_{j-1}+1}, \ldots, \hat{a}^*_{\alpha_j}, \ldots, \hat{a}_N)$, where $(\hat{a}^*_{\alpha_{j-1}+1}, \ldots, \hat{a}^*_{\alpha_j})$ is any permutation of

$(\hat{a}_{\alpha_{j-1}+1}, \ldots, \hat{a}_{\alpha_j})$. We introduce some nontrivial properties related to PM(d).

Proposition 3.2. $z_{LP}^{PM2} = z_{LP}^{PM(d)}$

Proof. For each $q \in P_W(d)$, let $P(q) = \{i \in I_{M_W} | f(\hat{a}^i) = a_q\}$. The conversion from the feasible solution $x^{\mathrm{PM}(d)}$ of the LP-relaxation of PM(d) to the feasible solution x^{PM2} of the LP-relaxation of PM2 can be easily shown by setting $x_j^{\mathrm{PM2}} = x_q^{\mathrm{PM}(d)}/|P(q)|$, for $j \in I_{M_W}$ and $q \in P_W(d)$ satisfying $f(\hat{a}^j) = a_q$. Then, $\sum_{i=1}^{M_W} \hat{\alpha_k}^i x_i^{\mathrm{PM2}}$ $= (\sum_{q \in P_W(d)} a_{q\beta_k} x_q^{\mathrm{PM}(d)})/d_{\beta_k} \leq 1$, for all $k \in I_N$. The other direction is easily shown by setting $x_q^{\mathrm{PM}(d)} = \sum_{i \in P(q)} x_i^{\mathrm{PM2}}$. □

Proposition 3.3. $z_{LP}^{PM2} \leq z_{LP}^{ML}$.

Proof. For each $i \in I_{M_W}$, let $\hat{t}(i)$ denote the minimum index which is in the support of $\hat{a}^i$. Also, let the support of $\hat{a}^i$ be $\hat{S}_i$. Then, given the feasible solution x^{PM2} of the LP-relaxation of PM2, we can construct a feasible solution x^{ML} to the LP-relaxation of ML defined as follows:

$$x_{jk}^{\mathrm{ML}} = \sum_{i \in I_{M_W}, \hat{t}(i)=k, k \in \hat{S}_i} x_i^{\mathrm{PM2}}, \quad \forall k \in \{1, \ldots, N\}, \quad \forall j \in \{k, \ldots, N\}.$$

□

Then, for each $k \in I_N$, let $y_k = (x_{kk}, \ldots, x_{Nk})$. Define P_k as follows:

$$\begin{aligned} P_k = \{y_k \in [0,1]^{N-k+1} | & \sum_{j=k+1}^{N} w_{\beta_j} x_{jk} \leq (W - w_{\beta_k}) x_{kk}, \\ & x_{jk} \leq x_{kk}, \quad \forall j \in \{k+1, \ldots, N\}, \\ & x_{jk} \in [0,1], \quad \forall j \in \{k, \ldots, N\}\}. \end{aligned}$$

Proposition 3.4. *Extreme points in P_k, which satisfy $N-k+1$ linearly independent inequalities as equality, can be represented as the form of sy, where $s \in [0,1]$ is a scalar and elements of $y \in [0,1]^{N-k+1}$ are all binary except at most one component.*

Proof. We show that if there exists i, j such that $i \neq j$, $x_{ik} \neq x_{kk}$, $x_{ik} \neq 0$, $x_{jk} \neq x_{kk}$, and $x_{jk} \neq 0$, then corresponding y_k cannot be an extreme point in P_k. Due to the assumption, none of $x_{ik} = x_{kk}$, $x_{jk} = x_{kk}$, $x_{ik} = 0$, $x_{jk} = 0$, $x_{ik} = 1$, or $x_{jk} = 1$ is satisfied. Therefore, even though $\sum_{j=k+1}^{N} w_{\beta_j} x_{jk} = (W - w_{\beta_k}) x_{kk}$ satisfies, at most $N-k$ linearly independent constraints can be satisfied in equality. □

Proposition 3.5. $z_{LP}^{ML} \leq 2 z_{LP}^{PM2}$.

Proof. To prove the proposition, it is sufficient to describe the procedure to construct a feasible solution of the LP-relaxation of PM2 from a feasible solution of the LP-relaxation of ML with at least half of its original value. Since any point in a polytope is a linear combination of its extreme points, we focus on converting extreme points of P_k into width patterns.

By Proposition 3.4, extreme points in P_k can be represented as the form of sy, where $s \in [0,1]$ is a scalar and elements of $y \in [0,1]^{N-k+1}$ are all binary except at most one component. Also, there always exist s and y such that the first element of y is unit amount. If there is a fractional element in y, setting it to zero and adding $k-1$ zeros at the foremost of y transforms y into a feasible width pattern vector of the instance $\hat{\mathbf{I}}$ (i.e., an element of $\hat{Q}_W$). Note that P_k corresponds to the constraints (2.17), (2.21), and (2.19) for fixed $k \in I_N$.

For any $sy^c \in P_k$ whose components of y^c are all binary except at most one component, let the corresponding width pattern vector of y^c by dropping the frac-

tional component and adding $k-1$ zeros at the foremost be $\hat{a}^c \in \hat{Q}_W$. If there is no fractional component in y^c, then adding $k-1$ zeros in the foremost of y^c makes the vector exactly same as $\hat{a}^c$. If there is a fractional component in y^c, let the fractional index(starts from k) be e and its value be y^c_e. If $\sum_{l=k,l\neq e}^{N} p_{\beta_l} y^c_l \leq p_{\beta_e} y^c_e$, then one may use width-pattern of only using the item e(corresponding to the eth unit vector in $\mathbf{R}^N$) instead of y^c if one can double the profit, because $\sum_{l=k,l\neq e}^{N} p_{\beta_l} y^c_l + p_{\beta_e} y^c_e \leq 2p_{\beta_e} y^c_e \leq 2p_{\beta_e}$, and $h_e \leq h_k$. Otherwise, one may use width-pattern vector $\hat{a}^c$ if one can double the profit, because $\sum_{l=k,l\neq e}^{N} p_{\beta_l} y^c_l + p_{\beta_e} y^c_e \leq 2\sum_{l=k,l\neq e}^{N} p_{\beta_l} y^c_l$. Therefore, the above procedure constructs a desirable feasible solution in PM2. □

Applying similar analysis to PM2, define $\hat{Q}^W_H$ as following:

$$\hat{Q}^W_H = \{y \in [0,1]^{M_W} \mid \sum_{i=1}^{M_W} \hat{c}_i y_i \leq H\}.$$

Note that unlike $\hat{Q}_H$, each element in $\hat{Q}^W_H$ describes the amounts of width patterns with total height equal or less than H. Also, an extreme point of $\hat{Q}^W_H$ has at most one fractional support. Let $R = \{\hat{r}^1, \ldots, \hat{r}^{M_R}\}$ denote the set of extreme points of $\hat{Q}^W_H$. For each $j \in I_{M_R}$, $\hat{r}^j$ is an M_W-dimensional vector. Let $V \in \mathbf{R}^{M_W \times M_R}$ be a matrix which each of its column is corresponding to the element of R and $U \in \mathbf{B}^{N \times M_W}$ be a matrix which each of its column is corresponding to the vector in $\hat{P}_W$. The LP-relaxation of PM2 is equivalent to the following model PM3:

$$
\begin{aligned}
\text{PM3}: \quad \text{maximize} \quad & \hat{p}^T U V x \\
\text{subject to} \quad & UVx \leq \hat{d}, \\
& \sum_{j \in I_{M_R}} x_j \leq 1, \\
& x_j \geq 0, \quad \forall j \in I_{M_R}.
\end{aligned}
$$

Then, let SM2 be the formulation of PM3 with rounding down the components in V. Note that for any $j \in I_{M_R}$, $\lfloor \hat{r}^j \rfloor$ is a "extended version" of the height pattern.

$$
\begin{aligned}
\text{SM2}: \quad \text{maximize} \quad & \hat{p}^T U \lfloor V \rfloor x \\
\text{subject to} \quad & U \lfloor V \rfloor x \leq \hat{d}, \\
& \sum_{j \in I_{M_R}} x_j \leq 1, \\
& x_j \geq 0, \quad \forall j \in I_{M_R}.
\end{aligned}
$$

Let the optimal objective value of PM3 (resp. SM2) be z^{PM3} (resp. z^{SM2}). Then, the following properties hold:

Proposition 3.6. $z^{SM2} = z_{LP}^{SM(d,d)}$

Proof. The proof is similar to the case of Proposition 3.2. To elaborate in detail, let $R_H^W = \{\lfloor \hat{r}^1 \rfloor, \ldots, \lfloor \hat{r}^{M_R} \rfloor\}$. Note that an element in R_H^W corresponds to a column vector of $\lfloor V \rfloor$ and $R_H^W \in R$. For each $i \in I_{M_R}$ and $j \in I_{M_W}$, $\lfloor \hat{r}^i \rfloor$ is a M_W-dimensional vector, and $\lfloor \hat{r}_j^i \rfloor$ represents how much jth width pattern vector in $\hat{Q}_W$ is used within height H. For each $i \in I_{M_W}$, let q_i be the width pattern such that $f(\hat{a}^i) = a_{q_i}$. Define

the onto function $g : R_H^W \to Q_H$ as follows:

$$g(\lfloor \hat{r}^i \rfloor)_k = \sum_{j \in I_{M_W}, t(q_j)=k} \lfloor \hat{r}_j^i \rfloor, \quad \forall \hat{r}^i \in R_H^W.$$

The meaning of the transformation g is that, the transformed vector from R_H^W describes how strip-defining items of width pattern vectors are ordered vertically within height H. Also, since it is possible to pack only a single item in each strip, it is clear that g is an onto function. Lastly, for each $r \in P_H(d)$, define $\hat{P}(r)$ as the following:

$$\hat{P}(r) = \{\lfloor \hat{r} \rfloor \in R_H^W | g(\lfloor \hat{r} \rfloor) = b_r\}.$$

We then suggest the conversion from the feasible solution $(x^{\mathrm{SM}(d,d)}, y^{\mathrm{SM}(d,d)})$ of the LP-relaxation of $\mathrm{SM}(d,d)$ to the feasible solution x^{SM2} of SM2 and vice versa. For given x^{SM2}, the following $(x^{\mathrm{SM}(d,d)}, y^{\mathrm{SM}(d,d)})$ is a feasible solution of the LP-relaxation of $\mathrm{SM}(d,d)$ with the same objective value:

$$y_r^{\mathrm{SM}(d,d)} = \sum_{i \in \hat{P}(r)} x_i^{\mathrm{SM2}}, \quad \forall r \in P_H(d),$$

and

$$x_q^{\mathrm{SM}(d,d)} = \sum_{i \in I_{M_W}, \hat{a}^i \in P(q)} (\lfloor V \rfloor x^{\mathrm{SM2}})_i, \quad \forall q \in P_W.$$

Also, it can be verified that $z^{\mathrm{SM2}} \geq z_{\mathrm{LP}}^{\mathrm{SM}(d,d)}$ since equally distributing the corresponding height and width patterns in instance $\mathbf{I}$ to the patterns in $\hat{\mathbf{I}}$ makes the feasible solution of SM2. □

Proposition 3.7. $z_{LP}^{PM2} = z^{PM3} \leq 2z^{SM2}$

Proof. The proof is similar to the proof of Proposition 3.5. Either a fractional component or the rest components of any element $\hat{r} \in R$ has equal or more than half of the profits generated by $\hat{r}$. Given a feasible solution in PM3, choosing the more profitable part of each element in R constructs a feasible solution in SM2 with equal or more than half of the original objective value. □

Theorem 3.8. $z^* \leq z_{LP}^{SM(d,d)} \leq z_{LP}^{PM(d)} \leq z_{LP}^{ML} \leq 2z_{LP}^{PM(d)} \leq 4z_{LP}^{SM(d,d)}$

Proof. With the results of Proposition 2.1, 3.2, and 3.3, the following inequality is valid:

$$z^* \leq z_{\mathrm{LP}}^{\mathrm{SM}(d,d)} \leq z_{\mathrm{LP}}^{\mathrm{PM}(d)} \leq z_{\mathrm{LP}}^{\mathrm{ML}}.$$

Also, Proposition 3.5, 3.6, and 3.7 prove the rest of the theorem. □

The tight example of $z_{\mathrm{LP}}^{\mathrm{ML}} \leq 2z_{\mathrm{LP}}^{\mathrm{PM}(d)} \leq 4z_{\mathrm{LP}}^{\mathrm{SM}(d,d)}$ is easily constructed: let the large plate has a (width, height) pair of (2M, 2M) and assume that there are only 4 different types of items sharing the same (width, height) pair $(M + 1,\ M + 1)$ with unit profit and unit demand. When $M \to \infty$, then $z_{\mathrm{LP}}^{\mathrm{ML}} \to 4$, $z_{\mathrm{LP}}^{\mathrm{PM}(d)} \to 2$, and $z_{\mathrm{LP}}^{\mathrm{SM}(d,d)} = z^* = 1$.

3.3 Polynomial-size Model

Since all the previous models are not polynomial-size models, verifying the existence of a polynomial-size model for the 2TDK is needed. With the result from Eisenbrand and Shmonin [8] and the structure of PM(u), it is possible to construct a polynomial-size model. Eisenbrand and Shmonin [8] provides the upper bound on numbers of nonzero components in the optimal solution in integer linear programming problems.

Lemma 3.9. *Let* $\min\{c^T y | Ay \leq b, y \in \boldsymbol{Z}_+^n\}$ *be an integer program, where* $A \in \boldsymbol{Z}_+^{d\times n}$ *and* $b \in \boldsymbol{Z}_+^n$*. If this integer program has a finite optimum value with* γ*, then there exists an optimal value* $y^* \in \boldsymbol{Z}_+^n$ *with the number of nonzero components of* y^* *at most* $\sum_{i=1}^n log_2(b_i + 1) + log_2(\gamma + 1)$*.*

Proof. As $\min\{c^T y | Ay \leq b, y \in \mathbf{Z}_+^n\} = \min\{c^T y | Ay + z = b, y \in \mathbf{Z}_+^n, z \in \mathbf{Z}_+^n\}$, the proof ends by the result of [8]. □

With this lemma, the following proposition holds:

Proposition 3.10. *Let* p_{max} *and* d_{max} *indicate the maximum value of components in given* p *and* d*, respectively. Then, there exists an optimal solution in PM*(d) *with at most* $M = \lceil log_2(n) + log_2(p_{max}) + (n+1)log_2(d_{max}) + log_2(H) \rceil$ *nonzero components.*

Proof. Using **Lemma** 3.9, there exists an optimal solution in PM(d) with at most $\lceil nlog_2(d_{max}) + log_2(H) + log_2(nd_{max}p_{max}) \rceil = M$ nonzero components. □

Therefore, there exists an optimal solution constructed by a polynomial number of width patterns and a polynomial-size NP certificate. Furthermore, with at most M patterns considered, a nonlinear polynomial-size model for the 2TDK is easily

found:

$$\text{maximize} \quad \sum_{i \in I_M} \sum_{j \in I_n} (p_j s_{ij}) x_i \tag{3.1}$$

$$\text{subject to} \quad \sum_{i \in I_M} x_i s_{ij} \leq d_j, \quad \forall j \in I_n, \tag{3.2}$$

$$\sum_{j \in I_n} w_j s_{ij} \leq W, \quad \forall i \in I_M, \tag{3.3}$$

$$\sum_{i \in I_M} x_i l_i \leq H, \tag{3.4}$$

$$s_{ij} \leq d_j z_{ij}, \quad \forall i \in I_M, j \in I_n, \tag{3.5}$$

$$l_i \geq h_j z_{ij}, \quad \forall i \in I_M, j \in I_n, \tag{3.6}$$

$$x \in \mathbf{Z}_+^M, l \in \mathbf{Z}_+^M, s \in \mathbf{Z}_+^{M \times n}, z \in \mathbf{B}^{M \times n}. \tag{3.7}$$

A tuple of decision variables $(s_{i1}, \ldots, s_{in})$ corresponds to the ith auxiliary width pattern, a decision variable l_i denotes the height of the ith auxiliary width pattern, and a decision variable z_{ij} represents the sign of s_{ij}. Then, a decision variable x_i shows how much ith auxiliary width pattern has been used. With these decision variables, constraints (3.1) represent the nonlinear objective function, and a set of constraints (3.2) corresponds to the demand constraint which is nonlinear. Constraints (3.3) and (3.5) indicate that each pattern defined as the variables s_i should be the feasible width pattern, Constraints (3.6) determine the height of the strip, and constraints (3.4) describe the height constraint. Lastly, groups of constraints (3.7) restrain the variables to be integer.

Note that for $i \in I_M$, $s_i = (s_{i1}, \ldots, s_{in})$ corresponds to the temporary width pattern and l_i corresponds to the height of its strip-defining item. In addition, since

s_{ij}, x_i are bounded by d_{max} and l_i is bounded by H, one can represent integer variables as the weighted sum of polynomial-size numbers of binary variables. For example, integer variable s_{ij} can be expressed as the following:

$$s_{ij} = \sum_{k=1}^{\lceil log_2(d_{max}) \rceil} 2^{k-1} s_{ijk},$$

where each s_{ijk} is a binary variable. Furthermore, if x and y are two binary variables, then xy can be expressed as the new binary variable k_{xy} satisfying the following inequality:

$$x + y - 1 \leq k_{xy}, \quad k_{xy} \leq x, \quad k_{xy} \leq y. \tag{3.8}$$

After converting all the integer variables into the weighted sum of binary variables, we can apply procedure (3.8) for all the nonlinear forms in (3.1), (3.2), and (3.4). Despite its complex structure, this transformation leads to the polynomial-size integer linear formulation. For convenience, let $\hat{D} = \lceil log_2(d_{max}) \rceil$ and $\hat{H} = \lceil log_2(H) \rceil$. The explicit version of polynomial-size formulation is as follows (for convenience, we do not explicitly mention the range of indices):

$$
\begin{aligned}
\text{POLY}: \quad \text{maximize} \quad & \sum_{i=1}^{n}\sum_{m=1}^{M}\sum_{k=1}^{\hat{D}}\sum_{l=1}^{\hat{D}} 2^{k+l-2} p_i s_{ikml} \\
\text{subject to} \quad & \sum_{m=1}^{M}\sum_{k=1}^{\hat{D}}\sum_{l=1}^{\hat{D}} 2^{k+l-2} s_{ikml} \le d_i, \quad \forall i \\
& \sum_{i=1}^{n}\sum_{k=1}^{\hat{D}} 2^{k-1} w_i \bar{q}_{ik}^{m} \le W, \quad \forall m, \\
& \sum_{m=1}^{M}\sum_{l=1}^{\hat{D}}\sum_{t=1}^{\hat{H}} 2^{l+t-2} r_{mlt} \le H, \\
& \sum_{k=1}^{\hat{D}} 2^{k-1} \bar{q}_{ik}^{m} \le d_i z_i^m, \quad \forall i, m, \\
& \bar{q}_{ik}^{m} \le z_i^m, \quad \forall i, k, m, \\
& \sum_{t=1}^{\hat{H}} 2^{t-1} \bar{H}_{mt} \ge h_i z_i^m, \quad \forall i, m, \\
& s_{ikml} \ge \bar{q}_{ik}^{m} + x_{ml} - 1, \quad \forall i, k, m, l, \\
& s_{ikml} \le \bar{q}_{ik}^{m}, \quad \forall i, k, m, l, \\
& s_{ikml} \le x_{ml}, \quad \forall i, k, m, l, \\
& r_{mlt} \ge x_{ml} + \bar{H}_{mt} - 1, \quad \forall m, l, t, \\
& r_{mlt} \le x_{ml}, \quad \forall m, l, t, \\
& r_{mlt} \le \bar{H}_{mt}, \quad \forall m, l, t, \\
& \text{all variables } s, \bar{q}, \bar{H}, r, z \text{ are binary.}
\end{aligned}
$$

Although it yields a polynomial-size model, the real usage of this model is not recommended. Detailed results are covered in Chapter 5.

Chapter 4

Exact Methods

Models except POLY are either exponential-size or pseudo-polynomial-size formulations. Pattern-based formulations have an exponential number of patterns, and an arc-flow formulation has variables corresponding to the height and width of the large plate explicitly. Lastly, the size of a level-packing model increases exponentially proportional to the size of the demands of items.

There have been various approaches to deal with large-scale formulations. For example, we usually choose to generate needed columns in each stage in respect of pattern-based formulations. Using column generation technique, Belov and Scheithauer [2] derived a branch-and-cut-and-price algorithm for the 2TDK based on the formulation PM(d). Also, as not all of constraints (2.21) may be needed, the branch-and-cut approach may be effective. See [6] for general information of solving large-scale mixed integer programs.

In addition, there have been some research which focus on reducing the size of formulations. For instance, Martinovic *et al.* [21] suggests some techniques to avoid unnecessary variables in one-dimensional cutting stock problems. In this thesis, we focus on the implementation of each exact method.

4.1 Branch-and-price Algorithm for the Strip Packing Model

Based on column generation techniques for solving the LP-relaxation, research including [19] summarizes the general methodology and some technical issues about branch-and-price algorithms. Also, Belov and Scheithauer [2] and Lodi and Monaci [18] illustrate the column-generation procedure for the strip packing model.

The main idea of this approach for solving PM(d) is that we only solve the problem with the restricted set of width patterns, named P_W instead of solving PM(d) with all width patterns in $P_W(d)$. Therefore, in column generation approach, we solve the following relaxed problem RPM(P_W):

$$\text{RPM}(P_W): \quad \text{maximize} \quad \sum_{a \in P_W} \sum_{i \in I_n} p_i a_{qi} x_q \tag{4.1}$$

$$\text{subject to} \quad \sum_{q \in P_W} a_{qi} x_q \leq d_i, \quad \forall i \in I_n, \tag{4.2}$$

$$\sum_{q \in P_W} h_{t(q)} x_q \leq H, \tag{4.3}$$

$$x_q \geq 0, \quad \forall q \in P_W. \tag{4.4}$$

Note that constraints (4.1), (4.2), and (4.3) only involve width patterns in P_W. Also, as we solve the linear program, a set of constraints (4.4) replaces the integer constraints. To check whether additional width patterns are needed, a slave problem has to be solved. Let $\mu \in \mathbf{R}^n$ be the dual variables of the demand constraints (4.2) and ν be the dual variable of the height constraint (4.3). If the optimal objective value of the following subproblem SPM(j, μ) is above $\nu * h_j$ for some $j \in I_n$, create

the new width pattern defined by the optimal values of a in SPM(j, μ).

$$\text{SPM}(j,\mu): \quad \max \quad \sum_{i \in I_n} (p_i - \mu_i) a_i \tag{4.5}$$

$$\text{subject to } a_i \leq d_i, \quad \forall i \in I_n, \tag{4.6}$$

$$\sum_{i \in I_n} w_i a_i \leq W, \tag{4.7}$$

$$a_i = 0, \quad \forall i \in I_{j-1}, \tag{4.8}$$

$$a_j \geq 1, \tag{4.9}$$

$$a_i \in \mathbf{Z}_+, \quad \forall i \in I_n. \tag{4.10}$$

The objective function (4.5) represents reduced costs, a set of constraints (4.6) restricts the pattern to satisfy the demand constraint, a constraint (4.7) makes the pattern to meet the width constraint, constraints (4.8) and (4.9) indicate that the subproblem is correlated with finding the width pattern with strip-defining item type j, and constraints (4.10) restrain the pattern vector to be integral. Then, whenever the optimal objective value of SPM(j, μ) is above $\nu * h_j$, add the corresponding width pattern to P_W. Until there is no additional width pattern to be generated, solve RPM(P_W) and SPM(j, μ) iteratively. After this procedure, it is convinced that the optimal objective value of RPM(P_W) becomes the LP-relaxation value of PM(d).

Combining the column generation procedure with a branch-and-bound methodology yields a branch-and-price algorithm. After solving RPM(P_W) completely (i.e., no additional patterns are needed), if optimal solution $x^* \in \mathbf{R}^{|P_W|}$ is integral, it becomes the optimal solution of PM(d). However, if there exists any $q_F \in P_W$ such that $x^*_{q_F}$ has a fractional value, then we need to branch the node into two child nodes:

a left child with additional constraint that $x_{q_F} \leq \lfloor x^*_{q_F} \rfloor$ and a right child with additional constraint that $x_{q_F} \geq \lfloor x^*_{q_F} \rfloor + 1$. See Figure 4.1 for a simple illustration of branching procedure.

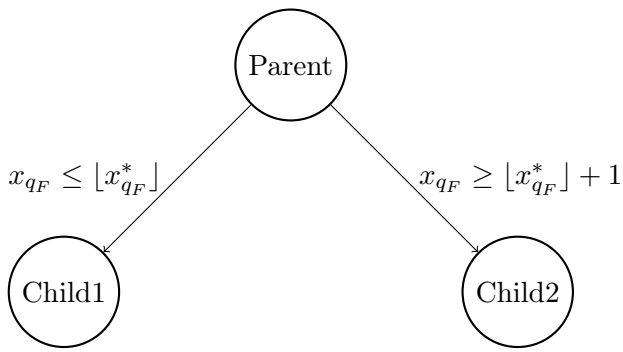


Figure 4.1: Branching strategy (variable dichotomy).

Then, an additional constraint that excludes the width pattern q_F to be generated again when solving the subproblem should be added to a left child node and child nodes of the left child node. Then, the following constraint should be added to $\mathrm{SPM}(t(q_F), \mu)$ of a left child node and child nodes of it:

$$a \neq a_{q_F}. \tag{4.11}$$

To express a constraint (4.11) as a linear constraint, we choose to transform each variable a_i in $\mathrm{SPM}(j, \mu)$ for any $i, j \in I_n$ into the weighted sum of binary variables $a^B_{i1}, \ldots, a^B_{i\hat{D}}$ so that $a_i = \sum_{k \in I_{\hat{D}}} 2^{k-1} a^B_{ik}$. Then, the subproblem $\mathrm{SPM}(j, \mu)$ can be represented as the following problem $\mathrm{SP}(j, \mu)$:

$$
\begin{aligned}
\mathrm{SP}(j,\mu): \quad \text{maximize} \quad & \sum_{i\in I_n}(p_i-\mu_i)a_i \\
\text{subject to} \quad & a_i \le d_i, \quad \forall i \in I_n, \\
& \sum_{i\in I_n} w_i a_i \le W, \\
& a_i = 0, \quad \forall i \in I_{j-1}, \\
& a_j \ge 1, \\
& a_i = \sum_{k\in I_{\hat{D}}} 2^{k-1} a^B_{ik}, \quad \forall i \in I_n, \\
& a^B_{ik} \in \mathbf{B}, \quad \forall i \in I_n, \quad \forall k \in I_{\hat{D}}.
\end{aligned}
$$

For $q_F \in Q_W$, there exists a unique binary vector $a^B_{q_F} = (a^B_{q_F 11}, \ldots, a^B_{q_F n\hat{D}})$ such that $a_{q_F i} = \sum_{k\in I_{\hat{D}}} 2^{k-1} a^B_{q_F ik}$. Then, (4.11) corresponds to the following inequality:

$$
\mathrm{NG}(q_F) := \sum_{i\in I_n, k\in I_{\hat{D}}, a^B_{q_F ik}=1} (a^B_{q_F ik} - a^B_{ik}) + \sum_{i\in I_n, k\in I_{\hat{D}}, a^B_{q_F ik}=0} (a^B_{ik} - a^B_{q_F ik}) \ge 1. \tag{4.12}
$$

Note that inequality (4.12) changes the structure of the problem. Both $\mathrm{SP}(j,\mu)$ and $\mathrm{SPM}(j,\mu)$ are knapsack problems, but adding inequality (4.12) complexes the problem.

We solve the subproblem by a solver offered by Xpress [9], and the best-bound strategy is selected as a search strategy. Also, we initialize P_W as the standard basis of $\mathbf{R}^n$. As a way to obtain decent lower bounds, we devise a simple heuristic that

solves an instance to optimality only using width patterns generated in the root node by the branch-and-bound method: i.e., solving RPM(P_W) to optimality. Besides, we make an effort to improve the lower bound by rounding down the fractional solution at each node. The overall procedure is summarized in Algorithm 1.

Algorithm 1 A branch-and-price algorithm for PM(d).

Input: An instance of the 2TDK
Initialize P_W;
Solve RPM(P_W) by the column generation method and get the optimal solution x^* and its optimal objective value UB;
if x^* is integral **then**
  return UB;
else
  Split the root node into two child nodes by the fractional component of x^*;
  Solve RPM(P_W) to optimality and save the optimal objective value as LB;
  $nd \leftarrow 2$;
  while $LB + 1 > UB$ or $nd = 0$ **do**
    Select the unsolved node whose parent node has a value of UB;
    Solve the selected node.
    if The selected node is feasible **then**
      Get the optimal solution x_N^* and its objective value UB_N of the node;
      if x_N^* is integral **then**
        if $LB < UB_N$ **then**
          $LB \leftarrow UB_N$.
        end if
      else
        Split the selected node into two child nodes;
        $nd \leftarrow nd + 2$;
        Update LB with rounding down the incumbent solution if possible;
      end if
    end if
    Update UB from the branch-and-bound tree;
    $nd \leftarrow nd - 1$;
  end while
  return LB;
end if

4.2 Branch-and-price Algorithm for the Staged-pattern Model

The overall structure of the branch-and-price scheme based on SM(d, d) is similar to that of PM(d) except for height pattern generation. To bolster the lower bound, we use the same heuristic devised for solving PM(d). Note that, for staged-pattern models, it is uncertain that all height patterns are generated at the root node. Therefore, even for staged-pattern models, we only utilize width patterns to construct RPM(P_W) and solve it through the branch-and-bound procedure. We propose two branch-and-price algorithms for the staged-pattern formulation: the standard scheme and the height-aggregated scheme.

4.2.1 The Standard Scheme

With restricted pattern set $P_W \in P_W(d)$ and $P_H \in P_H(d)$, let $\mu \in \mathbf{R}^n$ be the dual variable corresponding to the constraints (2.1), $v \in \mathbf{R}^n$ be the dual variable of the constraints (2.2), and ν be the dual variable of the constraint (2.3), respectively. To determine whether we need either additional height pattern or width pattern, we need to solve the $n + 1$ subproblems SPM(j, μ) for all $j \in I_n$ and a subproblem HPM(μ), which is defined as follows:

$$\text{HPM}(v): \quad \max \quad \sum_{i \in I_n} v_i b_i \tag{4.13}$$

$$\text{subject to } b_i \leq d_i, \quad \forall i \in I_n, \tag{4.14}$$

$$\sum_{i \in I_n} h_i b_i \leq H, \tag{4.15}$$

$$b_i \in \mathbf{Z}_+, \quad \forall i \in I_n. \tag{4.16}$$

Note that (4.13), (4.14), (4.15), and (4.16) makes the formulation as the bounded knapsack problem. If the objective value of $\text{SPM}(j, \mu)$ is above $v(j)$ for some $j \in I_n$, add the corresponding width pattern to P_W. If the objective value of $\text{HPM}(v)$ is above ν, add the corresponding height pattern to P_H. If any width pattern or height pattern is no more generated, it is convinced that we solve the LP-relaxation of $\text{SM}(d, d)$. Let x^* and y^* be the optimal solution after this column generation procedure. If x^* is integral, then it is convinced that we solve $\text{SM}(d, d)$ exactly as a set of constraints (2.4) can be relaxed.

If there exists $q_F \in P_W$ such that $x^*_{q_F}$ is fractional, we branch the problem as shown in Figure 4.1. For the left child node, inequality $NG(q_F)$ should be added to the subproblem $\text{SP}(t(q_F), \mu)$.

4.2.2 The Height-aggregated Scheme

In this scheme, we take advantage of SM-HA, which uses a minimal representation of height patterns. Denote the dual variable corresponding to the constraints (2.7), (2.8), and (2.9) by $\mu \in \mathbf{R}^m$, $v \in \mathbf{R}^M$, and $\nu \in \mathbf{R}$, respectively. To solve the

LP-relaxation of SM-HA by the column-generation procedure, $m+1$ subproblems SPM2(j,μ) for $j \in I_m$, and HPM2(ν) have to be solved to determine whether an additional width or height pattern is needed. SPM2(j,μ) and HPM2(ν) are defined as follows:

$$\begin{aligned}
\text{SPM2}(j,\mu): \quad \text{maximize} \quad & \sum_{i\in I_n}(p_i-\mu_i)a_i \\
\text{subject to} \quad & a_i \le d_i, \quad \forall i \in I_n, \\
& \sum_{i\in I_n} w_i a_i \le W, \\
& a_i = 0, \quad \forall i \in I_n \text{ such that } h_i > h_{(j)}, \\
& \sum_{i\in I_n, h_i = h_{(j)}} a_i \ge 1, \\
& a_i \in \mathbf{Z}_+, \quad \forall i \in I_n,
\end{aligned}$$

and

$$\begin{aligned}
\text{HPM2}(v): \quad \text{maximize} \quad & \sum_{i\in I_m} v_i b_i \\
\text{subject to} \quad & b_i \le \sum_{j\in I_n, h_j = h_{(i)}} d_j, \quad \forall i \in I_m, \\
& \sum_{i\in I_m} h_{(i)} b_i \le H, \\
& b_i \in \mathbf{Z}_+, \quad \forall i \in I_m.
\end{aligned}$$

In the aspect of the branch-and-price algorithm, it is similar to the case of the standard scheme that we branch the nodes by the fractional part of width pattern

usage, and (4.12) is added to the subproblem of the left child node. Although its LP-relaxation value is weaker than the LP-relaxation value of SM(d, d), it requires solving $m + 1$ subproblems at each iteration, and there is more freedom on selecting height pattern.

Furthermore, since the subproblem HPM(v) is a knapsack problem, for i_1 and $i_2 \in I_n$ such that $h_{i_1} = h_{i_2}$, the solution does not tend to choose both b_{i_1} and b_{i_2}. Therefore, some height-patterns that include various item types are relatively difficult to be generated. However, this phenomenon does not happen to the case of HPM2(v) since we only need to consider the summation of $b_{i_1} + b_{i_2}$. Therefore, we expect computational efficiency for the branch-and-price algorithm for SM-HA, which we discuss in Chapter 5.

We end up this subsection with the overall branch-and-price procedure for the staged-pattern models, as summarized in Algorithm 2. We present a pseudocode for the standard scheme since the two proposed schemes share a similar structure.

Algorithm 2 A branch-and-price algorithm for the staged-pattern models.

Input: An instance of the 2TDK
Initialize P_W and P_H;
Solve the LP-relaxation of the restricted master problem by the column generation method and get the optimal solution (x^*, y^*) and its optimal objective value UB;
if x^* is integral **then**
  return UB;
else
  Split the root node into two child nodes by the fractional component of x^*;
  Construct RPM(P_W) with width patterns at the root node;
  Solve RPM(P_W) to optimality, and save the optimal objective value as LB;
  $nd \leftarrow 2$;
  while $LB + 1 > UB$ **or** $nd = 0$ **do**
    Select the unsolved node whose parent node has a value of UB;
    Solve the selected node.
    if The selected node is feasible **then**
      Get the optimal solution x_N^* and its objective value UB_N of the node;
      if x_N^* is integral **then**
        if $LB < UB_N$ **then**
          $LB \leftarrow UB_N$.
        end if
      else
        Split the selected node into two child nodes;
        $nd \leftarrow nd + 2$;
        Update LB with rounding down the incumbent solution if possible;
      end if
    end if
    Update UB from the branch-and-bound tree;
    $nd \leftarrow nd - 1$;
  end while
  return LB;
end if

4.3 Branch-and-cut Algorithm for the Modified Level Packing Model

Although general solvers such as Xpress [9] can handle both LM and ML for instances with small N, more elaborate implementation should be considered for instances with large N. Furthermore, the number of inequalities (2.21) that we add to the formulation LM is about $\mathcal{O}(N^2)$, which can be computationally burdensome. Therefore, we devise a branch-and-cut algorithm (i.e., delayed constraint generation) for ML since not all inequalities may be needed to solve the problem exactly. To elaborate in detail, with a basic branch-and-bound scheme applied to LM, whenever we solve the LP-relaxation of each node, we check whether there exist some $k \in I_N$ and $j \in \{k+1, \ldots, N\}$ such that $x_{jk} > x_{kk}$. Then, we add violated inequalities $x_{jk} \leq x_{kk}$ to the formulation until we end up the branch-and-bound procedure. Only for solving the root node, we solve the node iteratively until no violated inequalities are found. For other nodes, we add violated inequalities only for once. The overall branch-and-cut procedure is summarized in Algorithm 3.

We also add groups of valid inequalities of LM suggested by Lodi and Monaci [18] to reduce symmetry when solving the problem. In this thesis, computational experiments for ML are implemented using the branch-and-cut algorithm.

Algorithm 3 A branch-and-cut algorithm for the modified-level packing model.

Input: An instance of the 2TDK
Solve the LP-relaxation of LM in each node and get the optimal solution x^*;
if the node is the root node **then**
  while x^* does not violate any of inequalities (2.21) **do**
    For all $k \in I_N$ and $j \in \{k+1, \ldots, N\}$, search all x^*_{jk} such that $x_{jk} > x_{kk}$;
    Add corresponding inequalities $x_{jk} \leq x_{kk}$ to the formulation;
    Repeat solving the root node;
  end while
else
  if there remains unsolved nodes **or** x^* is not integral **then**
    For all $k \in I_N$ and $j \in \{k+1, \ldots, N\}$, search all x^*_{jk} such that $x_{jk} > x_{kk}$;
    Add corresponding inequalities $x_{jk} \leq x_{kk}$ to the formulation;
    Branch the node by the fractional variable if possible;
    Solve the next node.
  else
    Terminate the branch-and-bound procedure.
  end if
end if
return the optimal objective value;

Chapter 5

Computational Experiments

We conduct computational experiments to observe the undiscovered tendency of each model in its real usage. In this thesis, we conduct experiments using the solvers offered by Xpress 8.9 [9] with Intel(R) Core(TM) i7-4770 CPU @ 3.10GHz and 16GB of RAM. The running time for solving an instance was limited to 600 s.

5.1 Instances

We divide the well-known instance set proposed by Hifi and Roucairol [15] into two groups: a group of (relatively) small instances and a group of large instances. A group of small instances consists of 16 instances, and a group of large instances is composed of 20 instances. Each of the group is summarized in Table 5.1 and 5.2, respectively. In Table 5.1 and 5.2, the headings $w_{\min}$ and $w_{\max}$ are defined as $\min_{i \in I_n} w_i$ and $\max_{i \in I_n} w_i$, respectively. $h_{\min}$ ($h_{\max}$) and $d_{\min}$ ($d_{\max}$) are defined in the same way. The known optimal objective value of each instance is presented with the heading **OPT**. Optimal objective values of instances that we failed to solve to optimality in this thesis are obtained from the computational result of Alvarez-Valdes et al. [1].

Table 5.1: Summary of small instances.

Name	n	W	H	$w_{\min}$	$w_{\max}$	$h_{\min}$	$h_{\max}$	$d_{\min}$	$d_{\max}$	OPT
2	10	40	70	9	31	7	35	1	3	2,535
2s	10	40	70	9	31	7	35	1	3	2,430
3	20	40	70	9	33	11	43	1	4	1,720
3s	20	40	70	9	33	11	43	1	4	2,599
A1s	20	50	60	9	33	11	43	1	4	2,950
A2s	20	60	60	12	33	14	42	1	4	3,423
A3	20	70	80	15	35	14	43	1	4	5,380
A4	20	90	70	9	33	11	43	1	3	5,885
A5	20	132	100	13	69	12	63	1	5	12,553
CHL1	30	132	100	13	69	12	63	1	5	8,360
CHL1s	30	132	100	13	69	12	63	1	5	13,036
CHL2	10	62	55	11	31	9	31	1	3	2,235
CHL2s	10	62	55	11	31	9	31	1	3	3,162
CHL5	10	20	20	1	20	2	14	1	3	363
CHL6	30	130	130	18	69	12	63	1	5	16,572
CHL7	35	130	130	19	57	18	54	1	5	16,728

Table 5.2: Summary of large instances.

Name	n	W	H	$w_{\min}$	$w_{\max}$	$h_{\min}$	$h_{\max}$	$d_{\min}$	$d_{\max}$	OPT
ATP30	38	927	152	57	360	7	58	1	9	140,168
ATP31	51	856	964	44	331	50	380	1	9	820,260
ATP32	56	307	124	16	120	6	46	1	9	37,880
ATP33	44	241	983	15	90	52	390	1	9	235,580
ATP34	27	795	456	46	308	22	173	1	9	356,159
ATP35	29	960	649	50	363	34	248	1	9	614,429
ATP36	28	537	244	30	209	20	91	1	9	129,262
ATP37	43	440	881	23	175	51	350	1	9	384,478
ATP38	40	731	358	41	289	19	140	1	9	259,070
ATP39	33	538	501	28	214	48	192	1	9	266,135
ATP40	56	683	138	34	270	6	54	1	9	63,945
ATP41	36	837	367	43	326	32	144	1	9	202,305
ATP42	59	167	291	8	65	21	114	1	9	32,589
ATP43	49	362	917	19	143	46	362	1	9	208,998
ATP44	39	223	496	11	88	29	193	1	9	70,940
ATP45	33	188	578	9	74	49	228	1	9	74,205
ATP46	42	416	514	23	157	40	204	1	9	146,402
ATP47	43	393	554	25	156	32	215	1	9	144,317
ATP48	34	931	254	47	355	18	99	1	9	165,428
ATP49	25	759	449	42	301	23	157	1	9	206,965

One of the classes of artificial instances proposed by Berkey and Wang (1987) [3] is also used to check the performance of each model when $d_{\max}$ changes. We focus on Class 5 whose instance is defined as $(n, 100, 100, h, w, d, p)$ with n taking a value among $\{20, 40, 60, 80, 100\}$, $w_i \sim U(1, 100)$, $h_i \sim U(1, 100)$, $d_i = 1$, and $p_i = h_i * w_i$ for $i \in I_n$, where U describes the discrete uniform distribution. We generated 10 instances per each $n \in \{20, 40, 60, 80, 100\}$.

Then, we changed demands of instances in Class 5 and analyze the impact on the overall performance of each model. For a given instance, we set the demand for each item the value Δ (i.e., $d_i = \Delta$ for all $i \in I_n$). Four different values of Δ are used: 1, 3, 5, and 7. As other information such as heights and widths is unchanged, an increase in Δ affects solving the instance to optimality in two ways: the first effect is that the problem becomes easier in a sense that the problem is getting similar to the unconstrained case, and the second effect is that the problem gets difficult as there are much more options for items to choose.

5.2 Upper Bounds Comparison

The objectives of this section are to reveal the answers to the following questions:

(a) How much the LP-relaxation values of models follow the Theorem 3.8 in a real situation?

(b) How are the qualities of the LP-relaxation values for various models?

(c) How effective is the only polynomial-size model POLY in the aspect of finding a solution?

5.2.1 A Group of Small Instances

The LP-relaxation values and time (seconds) consumed for solving the LP-relaxation of models in Chapter 2 are reported in Table 5.3. The headings for the LP-relaxation values and time are **LP** and $t_{\mathbf{LP}}$, respectively. We also compute the LP gap defined as the follows:

$$\text{LP gap} = \frac{\text{(LP-relaxation value)} - \text{(Optimal objective value)}}{\text{(Optimal objective value)}} \times 100(\%).$$

Optimal objective values for a group of small instances are obtained from the result of subsection . We summarize LP gaps in Table 5.4 and Figure 5.1. We exclude the average LP gap of POLY in 5.1 since it shows the average LP gap of over 500 (%). In figures in this chapter, PM and SM stand for PM(d) and SM(d, d), respectively.

Table 5.3: Time costs and the LP-relaxation valuse for small instances.

	AF		POLY		LM		ML		PM(d)		SM(d,d)		SM-HA	
Instance	LP	t_{LP}	LP	t_{LP}	LP	t_{LP}	LP	t_{LP}	LP	t_{LP}	LP	t_{LP}	LP	t_{LP}
2	2,741.000	0.047	4,449.000	0.046	2,878.000	0.000	2,863.679	0.015	2,658.943	0.063	2,651.357	0.250	2,651.357	0.172
2s	2,633.000	0.063	4,344.000	0.062	2,800.000	0.000	2,770.601	0.031	2,553.943	0.047	2,546.357	0.187	2,546.357	0.188
3	1,822.000	0.110	21,020.000	4.734	2,005.342	0.015	2,005.342	0.047	1,893.600	0.062	1,814.286	0.640	1,814.286	0.594
3s	2,644.083	0.110	44,500.000	16.483	2,800.000	0.031	2,800.000	0.015	2,668.578	0.047	2,628.500	0.735	2,628.500	0.735
A1s	2,950.000	0.125	44,500.000	5.765	3,000.000	0.032	3,000.000	0.063	2,991.111	0.046	2,950.000	0.641	2,950.000	0.641
A2s	3,423.000	0.172	34,625.000	5.546	3,600.000	0.000	3,600.000	0.031	3,474.300	0.079	3,423.000	1.407	3,423.000	0.782
A3	5,411.357	0.343	34,754.000	4.702	5,600.000	0.016	5,600.000	0.032	5,595.471	0.078	5,380.000	1.140	5,380.000	0.890
A4	6,172.167	0.688	23,613.000	0.625	6,300.000	0.000	6,248.895	0.032	6,100.765	0.140	5,971.000	1.282	5,971.000	1.609
A5	12,793.667	0.875	47,557.000	3.250	13,200.000	0.000	13,200.000	0.063	13,182.839	0.250	12,553.000	2.750	12,553.000	1.343
CHL1	8,804.196	3.453	37,535.000	25.920	9,121.646	0.015	8,920.966	0.203	8,768.395	0.656	8,380.000	6.969	8,380.000	5.359
CHL1s	13,124.133	3.203	64,799.000	14.530	13,200.000	0.031	13,200.000	0.203	13,193.854	0.312	13,036.000	9.015	13,036.000	6.562
CHL2	2,384.750	0.079	4,769.000	0.031	2,473.652	0.000	2,405.185	0.016	2,272.176	0.031	2,237.500	0.218	2,237.500	0.188
CHL2s	3,351.333	0.094	7,117.000	0.032	3,410.000	0.000	3,399.915	0.016	3,306.882	0.015	3,279.000	0.219	3,279.000	0.250
CHL5	379.284	0.000	974.000	0.062	400.000	0.000	400.000	0.016	379.000	0.031	363.000	0.110	364.500	0.094
CHL6	16,792.544	3.281	80,103.000	16.983	16,900.000	0.016	16,900.000	0.250	16,897.000	0.359	16,652.667	8.968	16,652.667	5.937
CHL7	16,824.536	4.953	85,112.000	22.670	16,900.000	0.031	16,900.000	0.578	16,813.355	0.578	16,728.000	11.874	16,728.000	6.875
Average		1.100		7.590		0.012		0.101		0.175		2.900		2.014

Table 5.4: LP gaps for small instances.

Instance	AF	POLY	LM	ML	PM(d)	SM(d,d)	SM-HA
2	8.126	75.503	13.531	12.966	4.889	4.590	7.473
2s	8.354	78.765	15.226	14.016	5.101	4.788	6.409
3	5.930	1,122.093	16.590	16.590	10.093	5.482	7.993
3s	1.735	1,612.197	7.734	7.734	2.677	1.135	1.135
A1s	0.000	1,408.475	1.695	1.695	1.394	0.000	0.000
A2s	0.000	911.540	5.171	5.171	1.499	0.000	0.000
A3	0.583	545.985	4.089	4.089	4.005	0.000	0.000
A4	4.880	301.240	7.052	6.183	3.666	1.461	1.461
A5	1.917	278.850	5.154	5.154	5.017	0.000	0.000
CHL1	5.313	348.983	9.111	6.710	4.885	0.239	0.239
CHL1s	0.676	397.077	1.258	1.258	1.211	0.000	0.000
CHL2	6.700	113.378	10.678	7.615	1.663	0.112	0.112
CHL2s	5.988	125.079	7.843	7.524	4.582	3.700	4.527
CHL5	4.486	168.320	10.193	10.193	4.408	0.000	0.970
CHL6	1.331	383.364	1.979	1.979	1.961	0.487	0.487
CHL7	0.577	408.800	1.028	1.028	0.510	0.000	0.000
Average	3.537	517.478	7.396	6.869	3.598	1.375	1.925

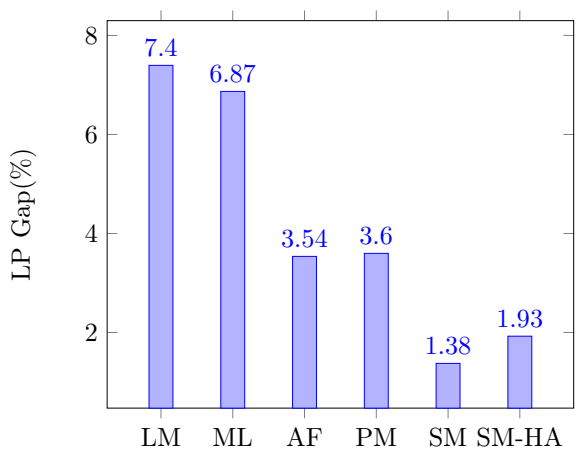


Figure 5.1: Average LP gaps for small instances.

Interestingly, $z_{\mathrm{LP}}^{\mathrm{AF}}$ is less than $z_{\mathrm{LP}}^{\mathrm{PM}(d)}$ in some instances such as 3, 3s, and A1s, while the relationship is reversed in other instances. Since $z_{\mathrm{LP}}^{\mathrm{AF}} = z_{\mathrm{LP}}^{\mathrm{SM}(\infty,\infty)}$, $z_{\mathrm{LP}}^{\mathrm{SM}(\infty,\infty)}$ and $z_{\mathrm{LP}}^{\mathrm{PM}(d)}$ are incomparable.

Although POLY costs the largest amount of time to compute the LP-relaxation values, its LP gaps are the worst. Then, AF seems to spend more time when H and W of an instance are relatively large. For level packing models, LM shows the fastest speed to compute the LP-relaxation values, and ML could enhance the LP-relaxation values with a small sacrifice on time. Pattern-based models generally require more time and level packing models for computing the LP-relaxation values, but LP gaps of pattern-based models are impressive. Especially, staged-pattern models sometimes offer the exact solution by solving only the root node.

To verify sizes of formulations, we summarize the numbers of variables and constraints of AF, POLY, and level packing models in Table 5.5. For pattern-based models, as we implement them by column generation, numbers of generated width patterns and height patterns ((**WP_Root** for width patterns and **HP_Root** for height patterns) are reported in Table 5.6. As shown in Table 5.5, AF and POLY show relatively large sizes of formulations. The compactness of level packing models is remarkable since POLY is a polynomial-size model, whereas level packing models are not.

Table 5.5: The number of variables and constraints for small instances.

	AF		POLY		LM		ML	
Instance	Variables	Constraints	Variables	Constraints	Variables	Constraints	Variables	Constraints
2	2,349	501	3,999	8,740	276	64	276	84
2s	2,349	501	3,906	8,537	276	64	276	86
3	5,765	931	24,444	59,157	1,953	189	1,953	191
3s	5,765	931	25,026	60,565	1,953	189	1,953	189
A1s	7,855	1,121	24,395	59,096	1,953	189	1,953	197
A2s	10,115	1,321	24,395	59,096	1,431	155	1,431	192
A3	12,593	1,541	25,026	60,565	1,081	127	1,081	156
A4	17,265	1,931	10,595	23,616	630	87	630	109
A5	23,782	2,801	25,317	61,269	1,035	126	1,035	187
CHL1	50,954	4,151	48,836	118,815	2,016	172	2,016	208
CHL1s	50,954	4,151	49,257	119,839	2,016	172	2,016	225
CHL2	3,228	706	3,780	8,285	190	48	190	58
CHL2s	3,228	706	3,780	8,285	190	48	190	58
CHL5	902	251	3,306	7,269	171	45	171	50
CHL6	50,877	4,121	50,150	121,925	2,145	179	2,145	246
CHL7	68,910	4,786	65,170	158,705	2,850	205	2,850	296

Table 5.6: The number of generated patterns for small instances.

	PM(d)	SM(d, d)		SM-HA	
Instance	WP_Root	WP_Root	HP_Root	WP_Root	HP_Root
2	20	27	23	21	16
2s	19	25	19	20	16
3	34	39	47	40	45
3s	32	33	47	33	48
A1s	40	41	44	41	43
A2s	40	46	49	43	38
A3	39	51	46	54	38
A4	40	57	37	54	40
A5	42	52	52	48	30
CHL1	66	96	76	95	58
CHL1s	60	83	80	85	60
CHL2	20	22	20	22	20
CHL2s	20	23	21	24	23
CHL5	15	19	18	16	17
CHL6	60	96	76	79	51
CHL7	71	104	92	99	55

Also, except for the instance A4, staged-pattern models generate more width patterns than a strip packing model. Because staged-pattern models can generate at most one height pattern at each iteration, it is plausible that they need much more time to compute the LP-relaxation values than the strip packing model.

Lastly, we compare the ratio of the LP-relaxation values of ML, PM(d), and SM(d, d). In Chapter 3, we theoretically proved that the ratio of $z_{\mathrm{LP}}^{\mathrm{ML}}/z_{\mathrm{LP}}^{\mathrm{PM}(d)}$ ($\frac{\mathrm{ML}}{\mathrm{PM}(d)}$) and $z_{\mathrm{LP}}^{\mathrm{ML}}/z_{\mathrm{LP}}^{\mathrm{SM}(d,d)}$ ($\frac{\mathrm{ML}}{\mathrm{SM}(d,d)}$) can be at most 2 and 4, respectively. For each instance in this set of instances, we summarize the ratios in Table 5.7. We verify that the ratios are far from the theoretical result in Theorem 3.8. Therefore, the LP-relaxations values of ML, PM(d), and SM(d, d) are much closer than we expected theoretically.

Table 5.7: The LP-relaxation values ratio for small instances.

Instance	$\frac{\text{ML}}{\text{PM}(d)}$	$\frac{\text{ML}}{\text{SM}(d,d)}$
2	1.077	1.080
2s	1.085	1.088
3	1.059	1.105
3s	1.049	1.065
A1s	1.003	1.017
A2s	1.036	1.052
A3	1.001	1.041
A4	1.024	1.047
A5	1.001	1.052
CHL1	1.017	1.065
CHL1s	1.000	1.013
CHL2	1.059	1.075
CHL2s	1.028	1.037
CHL5	1.055	1.102
CHL6	1.000	1.015
CHL7	1.005	1.010
Average	1.031	1.054

5.2.2 A Group of Large Instances

As in the previous subsection, we report the LP-relaxation values, time consumed for solving the LP-relaxation of each model, and LP gaps. Using a simplex method to solve the LP-relaxation of POLY did not succeed in any case within 600 s. On the other hand, solving the LP-relaxation of AF worked for some instances. Since the sizes of both LP relaxations of AF and POLY are too large and sparse, we applied the barrier method to solve them. The overall result of AF and POLY is reported in Table 5.8. We denote the case that exceeds the time limit by **TL**.

Table 5.8: Summary of the result of AF and POLY for large instances.

	AF						POLY					
Instance	Variables	Constraints	t_{LP}	t_{LP} (Barrier)	LP	LP Gap	Variables	Constraints	t_{LP} (Barrier)	LP	LP Gap
ATP30	571,441	35,493	TL	19.561	140,303.0	0.096	397,604	155,770	132.819	1,097,239.0	682.803
ATP31	966,732	44,774	TL	38.169	821,800.0	0.188	692,927	271,125	25.074	1,076,106.0	31.191
ATP32	411,922	17,485	274.919	8.640	37,935.1	0.145	786,634	307,395	179.535	444,878.0	1,074.440
ATP33	226,525	11,720	459.718	4.453	235,604.0	0.010	526,912	206,358	19.686	2,722,275.0	1,055.563
ATP34	269,632	22,003	212.798	7.375	358,889.0	0.767	222,542	87,472	20.421	1,822,958.0	411.838
ATP35	359,183	28,577	394.801	12.874	618,867.0	0.722	256,454	100,776	56.621	4,322,082.0	603.431
ATP36	191,779	15,365	114.508	4.874	129,617.0	0.275	231,725	91,008	75.073	921,161.0	612.631
ATP37	369,543	19,931	321.229	9.265	386,248.0	0.460	508,188	199,056	20.061	4,092,713.0	964.486
ATP38	508,271	29,719	592.772	15.514	260,039.0	0.374	442,496	173,355	19.780	2,528,339.0	875.929
ATP39	264,049	18,355	167.551	6.234	267,364.0	0.462	315,831	123,914	21.014	2,862,785.0	975.689
ATP40	896,738	38,555	TL	21.702	64,424.0	0.749	799,033	312,320	551.556	698,768.0	992.764
ATP41	462,969	30,608	TL	13.437	204,520.0	1.095	365,471	143,290	22.123	1,577,986.0	680.003
ATP42	262,778	10,322	126.976	4.109	33,175.2	1.799	883,120	345,184	434.736	331,165.0	916.186
ATP43	399,878	18,803	340.477	8.999	213,754.0	2.276	639,227	250,173	21.748	1,745,393.0	735.124
ATP44	159,780	9,311	71.963	2.281	73,119.0	3.072	418,675	164,052	87.853	442,101.0	523.204
ATP45	104,916	6,882	9.765	1.297	74,444.8	0.323	315,932	124,002	109.883	379,496.0	411.416
ATP46	338,875	18,113	291.464	6.953	146,955.0	0.378	485,010	190,008	251.170	1,091,678.0	645.672
ATP47	325,184	17,583	220.672	6.250	147,977.0	2.536	505,745	198,099	168.785	1,132,311.0	684.600
ATP48	476,796	32,011	TL	17.499	166,680.0	0.757	326,712	128,102	63.777	971,498.0	487.263
ATP49	224,229	19,500	195.174	5.890	210,560.0	1.737	194,073	76,342	28.310	804,135.0	288.537
Average				10.769		0.911			115.501		682.638

Since the number of variables and constraints of POLY is plentiful, even solving the LP-relaxation of POLY by the barrier method took a large amount of time. Also, the upper bounds provided by the LP-relaxation of POLY is meaningless. Therefore, albeit its theoretical polynomial-size formulation, solving problems using POLY is not preferable. On the other hand, AF shows a better result when using the barrier method. Although the number of variables is large, the sparsity of the overall structure may be favorable to the barrier method.

The general performance of each model is reported in Table 5.9, and the average LP gaps are illustrated in Figure 5.2. In the case of level packing models, sizes of their formulations are much smaller than them of AF and POLY, which leads to fewer time costs for solving their LP relaxations. The number of variables and constraints of level packing models are summarized in Table 5.10. Solving the LP relaxation of LM was the fastest on average, and its compact formulation may contribute to this result. Besides, although ML could provide a better quality of lower bounds, overhead from adding violated inequalities seems not negligible. As shown in Table 5.10, hundreds of cuts were found and added to its formulation.

Generally, ML was dominated by the outcome of PM(d) since the average time cost and LP gap of PM(d) outweigh them of ML. PM(d) shows the fastest computational cost among pattern-based models, but staged-pattern models are more favorable for their strengths of upper bounds as they sometimes guarantee the optimal objective values. Also, SM-HA proved to improve the column generation process for staged-pattern models with little sacrifice on the LP-relaxation values.

Table 5.9: Time costs, LP-relaxation values, and LP gaps for large instances.

	LM			ML			PM(d)			SM(d, d)			SM-HA		
Instance	LP	t_{LP}	LP Gap	LP	t_{LP}	LP Gap	LP	t_{LP}	LP Gap	LP	t_{LP}	LP Gap	LP	t_{LP}	LP Gap
ATP30	140,904.0	0.109	0.525	140,904.0	2.609	0.525	140,814.7	1.140	0.461	140,207.0	23.045	0.028	140,209.5	20.343	0.030
ATP31	825,184.0	0.250	0.600	825,184.0	6.009	0.600	824,220.0	1.141	0.483	820,868.5	74.495	0.074	820,868.5	59.308	0.074
ATP32	38,068.0	0.141	0.496	38,068.0	2.453	0.496	37,910.1	1.266	0.079	37,889.5	112.460	0.025	37,889.5	51.012	0.025
ATP33	236,903.0	0.141	0.562	236,903.0	4.812	0.562	235,734.0	6.765	0.065	235,580.0	35.279	0.000	235,580.0	35.872	0.000
ATP34	362,520.0	0.062	1.786	362,520.0	1.265	1.786	357,477.1	0.453	0.370	356,931.1	10.062	0.217	357,323.7	12.374	0.327
ATP35	623,040.0	0.093	1.401	623,040.0	1.641	1.401	617,352.8	0.531	0.476	616,651.4	12.953	0.362	616,651.4	11.014	0.362
ATP36	131,028.0	0.063	1.366	131,028.0	2.187	1.366	130,136.4	0.578	0.676	129,486.8	11.296	0.174	129,486.8	10.109	0.174
ATP37	387,640.0	0.157	0.822	387,640.0	1.875	0.822	385,900.0	1.422	0.370	384,665.3	55.996	0.049	384,919.9	38.935	0.115
ATP38	261,698.0	0.109	1.014	261,698.0	2.562	1.014	259,434.5	0.781	0.141	259,329.5	39.575	0.100	259,329.5	25.030	0.100
ATP39	269,538.0	0.047	1.279	269,538.0	1.187	1.279	268,668.0	0.531	0.952	266,585.5	19.561	0.169	266,585.5	12.811	0.169
ATP40	68,547.3	0.391	7.197	68,076.3	3.844	6.461	64,425.8	0.906	0.752	63,963.4	53.199	0.029	63,963.4	30.670	0.029
ATP41	215,993.0	0.125	6.766	213,954.8	0.907	5.759	205,389.2	0.453	1.525	202,305.0	17.874	0.000	202,305.0	18.655	0.000
ATP42	34,080.1	0.485	4.576	33,691.7	4.562	3.384	32,932.9	1.640	1.055	32,789.0	94.556	0.614	32,789.0	61.652	0.614
ATP43	222,175.7	0.250	6.305	221,279.0	1.312	5.876	214,503.6	1.110	2.634	212,093.3	64.198	1.481	212,093.3	46.419	1.481
ATP44	77,453.5	0.140	9.182	77,082.9	1.344	8.659	74,652.5	0.375	5.233	72,658.4	15.202	2.422	72,658.4	16.108	2.422
ATP45	77,892.4	0.110	4.969	77,484.8	1.547	4.420	74,324.9	0.266	0.162	74,205.0	8.843	0.000	74,205.0	19.467	0.000
ATP46	154,646.5	0.140	5.631	154,646.5	1.249	5.631	148,735.2	0.531	1.594	146,402.0	35.544	0.000	146,402.0	25.576	0.000
ATP47	157,521.8	0.156	9.150	157,160.3	0.734	8.899	150,603.0	0.453	4.356	144,526.5	23.749	0.145	144,526.5	23.499	0.145
ATP48	173,553.0	0.110	4.911	173,504.7	1.078	4.882	166,929.8	0.672	0.908	165,944.5	17.874	0.312	165,944.5	13.921	0.312
ATP49	226,610.4	0.063	9.492	224,695.2	0.546	8.567	210,651.6	0.437	1.781	208,511.5	7.297	0.747	208,511.5	8.671	0.747
Average		0.157	3.902		2.186	3.620		1.073	1.204		36.653	0.347		27.072	0.356

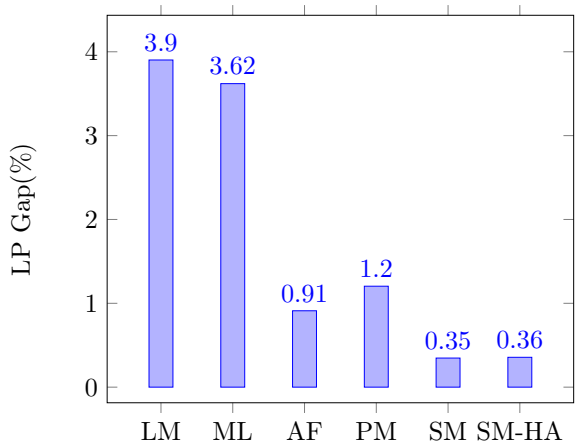


Figure 5.2: Average LP gaps for large instances.

Table 5.10: The number of variables and constraints of level packing models for large instances.

	LM		ML	
Instance	Variables	Constraints	Variables	Constraints
ATP30	18,528	655	18,528	821
ATP31	33,411	883	33,411	1,086
ATP32	31,125	834	31,125	1,002
ATP33	25,200	767	25,200	944
ATP34	8,515	442	8,515	592
ATP35	11,781	527	11,781	688
ATP36	11,781	529	11,781	692
ATP37	24,753	761	24,753	887
ATP38	20,503	689	20,503	836
ATP39	13,366	556	13,366	636
ATP40	42,195	994	42,195	1,044
ATP41	15,753	602	15,753	654
ATP42	52,975	1,127	52,975	1,242
ATP43	33,670	892	33,670	923
ATP44	19,306	674	19,306	718
ATP45	12,246	527	12,246	625
ATP46	19,503	664	19,503	702
ATP47	20,910	693	20,910	728
ATP48	14,028	568	14,028	621
ATP49	7,140	403	7,140	459

Then, we compare the numbers of generated patterns when solving the LP-relaxations of pattern-based models in Table 5.11. SM(d, d) generated 45 (%) more width patterns than PM(d) in average. SM-HA could reduce its computational time by less generating height patterns. As LP gaps of SM and SM-HA are almost the same, SM-HA seems to produce height patterns more efficiently.

Table 5.11: The number of generated patterns for large instances.

	PM(d)	SM(d, d)		SM-HA	
Instance	WP_Root	WP_Root	HP_Root	WP_Root	HP_Root
ATP30	78	111	113	89	93
ATP31	102	136	159	134	142
ATP32	111	172	234	148	119
ATP33	88	102	143	106	143
ATP34	55	83	79	77	89
ATP35	59	80	85	79	76
ATP36	56	64	71	69	61
ATP37	86	119	152	116	117
ATP38	80	116	139	109	95
ATP39	65	81	100	72	78
ATP40	110	167	179	157	106
ATP41	73	92	128	92	124
ATP42	121	201	257	175	168
ATP43	98	148	184	143	156
ATP44	78	107	112	107	112
ATP45	65	101	87	102	135
ATP46	83	155	163	155	125
ATP47	86	141	134	129	131
ATP48	68	95	107	94	81
ATP49	52	70	67	71	74

Lastly, we compute the relative ratios of the LP-relaxation values between ML, PM(d), and SM(d, d). The result is in Table 5.12. As shown in Table 5.7 and 5.12, the result from Theorem 3.8 seems rather theoretical.

Table 5.12: The LP-relaxation values ratio for large instances.

Instance	$\frac{\text{ML}}{\text{PM}(d)}$	$\frac{\text{ML}}{\text{SM}(d,d)}$
ATP30	1.001	1.005
ATP31	1.001	1.005
ATP32	1.004	1.005
ATP33	1.005	1.006
ATP34	1.014	1.016
ATP35	1.009	1.010
ATP36	1.007	1.012
ATP37	1.005	1.008
ATP38	1.009	1.009
ATP39	1.003	1.011
ATP40	1.057	1.064
ATP41	1.042	1.058
ATP42	1.023	1.028
ATP43	1.032	1.043
ATP44	1.033	1.061
ATP45	1.043	1.044
ATP46	1.040	1.056
ATP47	1.044	1.087
ATP48	1.039	1.046
ATP49	1.067	1.078
Average	1.024	1.033

To conclude, level packing models can solve their LP-relaxation models quickly, but the LP-relaxation values are more credible in the case of pattern-based models. More elaborate algorithms should be constructed in order to utilize AF or POLY computationally.

5.2.3 Class 5 Instances

In this subsection, we analyze how the performances of various models would change when the overall demand changes. Since we could get all the optimal values of this set of instances using SM-HA, LP gaps could be computed. The average time and the average LP gap are reported in Table 5.13.

Table 5.13: Average time costs and LP gaps for Class 5.

		AF		LM		ML		PM(d)		SM(d, d)		SM-HA	
Δ	n	t_{LP}	LP Gap	t_{LP}	LP Gap	t_{LP}	LP Gap	t_{LP}	LP Gap	t_{LP}	LP Gap	t_{LP}	LP Gap
1	20	0.428	3.144	0.003	7.352	0.011	7.275	0.056	3.022	0.487	0.026	0.428	0.052
1	40	3.280	2.173	0.002	3.117	0.023	3.107	0.138	1.655	2.836	0.074	2.186	0.111
1	60	19.186	1.348	0.003	1.943	0.027	1.943	0.250	1.359	8.257	0.023	6.521	0.161
1	80	48.662	0.889	0.025	1.158	0.097	1.158	0.342	0.778	16.408	0.000	12.655	0.069
1	100	106.067	0.518	0.038	0.667	0.129	0.667	0.540	0.520	33.910	0.006	25.131	0.010
3	20	0.442	0.803	0.008	3.717	0.139	3.717	0.060	2.434	1.009	0.085	0.851	0.112
3	40	3.011	0.287	0.045	0.764	0.245	0.764	0.161	0.670	6.923	0.044	5.098	0.048
3	60	18.856	0.271	0.080	0.635	0.416	0.635	0.250	0.635	23.645	0.033	15.803	0.054
3	80	47.209	0.133	0.291	0.277	0.728	0.277	0.364	0.251	37.138	0.036	24.039	0.036
3	100	102.963	0.191	0.469	0.229	0.908	0.229	0.611	0.229	82.555	0.024	46.019	0.042
5	20	0.444	0.692	0.030	3.114	0.202	3.114	0.083	2.525	1.631	0.230	1.206	0.249
5	40	3.075	0.195	0.095	0.568	0.711	0.568	0.225	0.507	10.731	0.052	7.114	0.052
5	60	19.086	0.167	0.192	0.451	2.185	0.451	0.336	0.451	25.478	0.037	17.302	0.043
5	80	45.423	0.064	0.419	0.173	2.849	0.173	0.483	0.173	49.020	0.014	28.614	0.014
5	100	105.352	0.069	0.688	0.086	7.447	0.086	0.786	0.086	92.762	0.020	57.188	0.021
7	20	0.427	0.808	0.039	2.940	0.322	2.940	0.066	2.383	1.267	0.270	1.089	0.290
7	40	2.984	0.191	0.177	0.520	1.518	0.520	0.183	0.459	8.135	0.015	5.798	0.021
7	60	18.094	0.184	0.436	0.421	4.237	0.421	0.270	0.421	21.116	0.045	14.841	0.045
7	80	46.023	0.056	1.030	0.153	13.709	0.153	0.345	0.153	39.214	0.022	24.903	0.022
7	100	103.285	0.067	1.822	0.078	30.157	0.078	0.617	0.078	70.995	0.016	44.209	0.016

In this set of instances, because the profit of each item is set to be the area of the item, we get the trivial upper bound $W * H = 10000$. It is interesting that except for the case of $\Delta = 1$ and $n = 20$, all the LP-relaxation values of LM were equal to 10000. LP gaps of level packing models are relatively high, and the effect of additional inequalities (2.21) was minute. Also, we were able to solve the LP-relaxation of AF by the simplex method due to small H and W, and the result shows that LP-relaxation values of AF and pattern-based models are much credible than level packing models. Especially, in the case of $\Delta = 1$ and $n = 80$, the LP-relaxation values of SM(d, d) were all as same as optimal objective values.

Table 5.14: The number of variables and constraints for Class 5.

		AF		LM		ML	
Δ	n	Variables	Constraints	Variables	Constraints	Variables	Constraints
1	20	14,228	2,161	210	39	210	39
1	40	48,061	4,221	820	79	820	79
1	60	100,613	6,281	1,830	119	1,830	119
1	80	168,856	8,341	3,240	159	3,240	159
1	100	271,346	10,401	5,050	199	5,050	199
3	20	14,053	2,161	1,830	119	1,830	119
3	40	48,061	4,221	7,260	239	7,260	239
3	60	100,613	6,281	16,290	359	16,290	359
3	80	168,856	8,341	28,920	479	28,920	479
3	100	271,346	10,401	45,150	599	45,150	599
5	20	14,053	2,161	5,050	199	5,050	199
5	40	48,061	4,221	20,100	399	20,100	399
5	60	100,613	6,281	45,150	599	45,150	599
5	80	168,856	8,341	80,200	799	80,200	799
5	100	271,346	10,401	125,250	999	125,250	999
7	20	14,053	2,161	9,870	279	9,870	279
7	40	48,061	4,221	39,340	559	39,340	559
7	60	100,613	6,281	88,410	839	88,410	839
7	80	168,856	8,341	157,080	1,119	157,080	1,119
7	100	271,346	10,401	245,350	1,399	245,350	1,399

Even though AF does not use a column generation method, the amount of time

to obtain the LP-relaxation values of AF was most demanding. In order to delineate this phenomenon, we suspect that the size of AF may yield this result. This doubt turns out to be valid, as shown in Table 5.14.

Level packing models seem very compact when Δ is low, but the number of their variables becomes closer to the number of variables in AF when Δ increases. Therefore, the time cost of level packing models cannot but exceed the time cost for the strip packing model for large Δ. On the other hand, the time costs for pattern-based models show dependency on n, rather than Δ. To bolster this opinion, we summarized the number of patterns generated at the root node of pattern-based models at Table 5.15. Note that the numbers of generated patterns are also dependent on n.

Table 5.15: The number of patterns generated at the root node for Class 5.

		PM(d)	SM(d, d)		SM-HA	
Δ	n	WP_Root	WP_Root	HP_Root	WP_Root	HP_Root
1	20	27.9	35.2	35.4	33.7	31.2
1	40	55.4	77.0	74.7	71.0	59.1
1	60	68.0	107.5	115.0	102.6	89.4
1	80	93.8	140.1	157.9	128.9	117.3
1	100	112.2	174.6	206.9	169.2	147.4
3	20	25.3	34.0	42.9	33.7	36.9
3	40	49.4	67.8	102.8	66.5	79.8
3	60	68.2	100.0	177.3	93.0	126.8
3	80	93.2	127.8	218.7	121.7	147.1
3	100	114.4	160.0	285.9	149.2	172.3
5	20	25.9	33.2	48.3	32.3	40.3
5	40	50.5	66.1	113.6	62.6	79.7
5	60	69.9	91.2	162.4	89.5	118.0
5	80	92.5	122.6	211.1	115.2	133.1
5	100	113.0	153.1	266.0	148.4	169.0
7	20	26.4	31.8	46.8	31.7	40.4
7	40	50.9	63.3	105.5	61.5	79.3
7	60	70.4	88.8	162.7	86.5	120.0
7	80	92.1	119.1	217.5	112.3	144.3
7	100	113.6	147.6	259.1	140.3	165.2

5.3 Solving Instances to Optimality

The purpose of this section is to address issues when solving problems using commercial solvers and to verify the effectiveness of exact methods provided in Chapter 4. The running time for each instance was limited to 600 s per instance.

5.3.1 A Group of Small Instances

The best lower bound obtained by the feasible solution within the time limit is recorded as **Sol**. Time (seconds) consumed for solving the instance exactly (**Time**), and the number of nodes in a branching tree are reported for each model in Table 5.16 and 5.17. We denote the case that exceeds the time limit by **TL**. For pattern-based models, the numbers of generated width patterns and height patterns are reported as **WP** and **HP**, respectively.

Although adding (2.21) yields the tighter LP-relaxation bounds, the total running time of it does not prove effectiveness. The overhead of adding inequalities seems to outweigh its gain on exactitude. It is also interesting that sometimes ML requires a larger number of nodes than LM in spite of its additional inequalities. However, both level packing models show excellence solving problems of small sizes.

AF seems to be unstable in respect of time costs and nodes so that it is difficult to certain whether it will solve a problem quickly. Fortunately, the best lower bounds of AF are all as same as the optimal objective values, whereas the best lower bounds of PM(d) are not always equivalent to the optimal objective values.

A trial to solve problems using POLY is not recommendable since it failed to solve any small instance to optimality within 600 s. Even with computing lots of nodes, its best lower bounds are approximately 10 percent lower than optimal objective

values.

For pattern-based models, it was not sufficient for the height-aggregated scheme to prove its efficiency in time costs and nodes. Instead, in the aspect of the numbers of generated height patterns and width patterns, SM-HA shows outstanding performance. Generally, staged-pattern models produced smaller numbers of width patterns than the strip packing model, which is opposite to the result at the root node.

Table 5.16: Summary of the result of non-pattern-based models for small instances.

	AF			POLY			LM			ML		
Instance	Sol	Time	Nodes	Sol	Time	Nodes	Sol	Time	Nodes	Sol	Time	Nodes
2	2,535	TL	2,619,049	2,431	TL	757,348	2,535	0.110	1,021	2,535	0.110	809
2s	2,430	TL	3,047,367	2,328	TL	796,473	2,430	0.156	3,020	2,430	0.187	1,883
3	1,720	11.665	33,150	1,480	TL	29,958	1,720	0.172	993	1,720	0.312	759
3s	2,599	0.641	47	2,391	TL	37,617	2,599	0.172	903	2,599	0.328	849
A1s	2,950	0.531	1	2,950	TL	32,725	2,950	0.125	587	2,950	0.156	301
A2s	3,423	0.609	1	3,412	TL	24,427	3,423	0.219	2,453	3,423	0.984	4,947
A3	5,380	0.984	19	4,738	TL	38,361	5,380	0.250	4,093	5,380	0.500	3,109
A4	5,885	7.144	1,983	5,885	TL	146,420	5,885	0.141	1,979	5,885	0.406	4,102
A5	12,553	23.067	8,671	12,404	TL	29,960	12,553	0.359	5,885	12,553	1.609	9,781
CHL1	8,360	208.861	58,847	4,560	TL	134,937	8,360	1.125	18,155	8,360	9.015	23,347
CHL1s	13,036	5.537	12	11,894	TL	11,091	13,036	0.516	6,537	13,036	2.984	9,929
CHL2	2,235	0.705	507	2,235	TL	778,125	2,235	0.063	177	2,235	0.062	115
CHL2s	3,162	1.953	3,657	2,978	TL	754,314	3,162	0.078	235	3,162	0.047	327
CHL5	363	0.640	2,655	363	TL	761,786	363	0.031	47	363	0.016	19
CHL6	16,572	11.942	545	10,332	TL	9,898	16,572	6.359	88,498	16,572	20.139	57,339
CHL7	16,728	17.817	877	12,953	TL	21,427	16,728	17.249	140,705	16,728	67.964	140,483

Table 5.17: Summary of the result of pattern-based models for small instances.

	PM(d)				SM(d,d)					SM-HA				
Instance	Sol	Time	Nodes	WP	Sol	Time	Nodes	WP	HP	Sol	Time	Nodes	WP	HP
2	2,535	73.003	4,087	134	2,535	58.449	2,631	122	139	2,535	96.884	3,727	130	40
2s	2,430	42.388	2,199	118	2,430	59.542	2,613	139	136	2,430	106.695	4,347	158	40
3	1,720	9.202	983	90	1,720	6.031	293	69	100	1,720	6.124	301	69	98
3s	2,599	2.000	161	64	2,599	1.406	15	38	55	2,599	1.344	15	37	52
A1s	2,950	0.281	11	41	2,950	0.672	1	41	44	2,950	0.656	1	41	43
A2s	3,423	126.554	769	390	3,423	1.406	1	46	49	3,423	0.781	1	43	38
A3	5,380	187.768	2,603	439	5,380	1.125	1	51	46	5,380	0.906	1	54	38
A4	5,885	TL	2,159	549	5,885	6.578	41	79	49	5,885	7.046	41	79	49
A5	12,553	TL	2,291	514	12,553	2.734	1	52	52	12,553	1.406	1	48	30
CHL1	8,360	TL	3,475	487	8,360	8.312	3	102	82	8,360	7.499	3	101	69
CHL1s	13,036	TL	1,027	525	13,036	9.015	1	83	80	13,036	6.546	1	85	60
CHL2	2,235	10.124	273	145	2,235	0.250	3	23	21	2,235	0.266	3	23	21
CHL2s	3,162	14.592	361	150	3,162	1.438	73	41	39	3,162	1.515	73	43	35
CHL5	363	1.469	125	60	363	0.094	1	19	18	363	0.140	3	17	18
CHL6	16,572	TL	1,205	485	16,572	25.358	51	114	115	16,572	16.467	49	92	61
CHL7	16,728	TL	809	446	16,728	11.827	1	104	92	16,728	6.921	1	99	55
Average				290				70	70				70	47

5.3.2 A Group of Large Instances

As Table 5.8 indicates that solving instances to optimality using AF and POLY seems reckless for large instances, we only provide results of level packing models and pattern-based models in Table 5.18 and 5.19, respectively.

Almost all of the instances were not solved to optimality by level packing models within 600 s. Comparing the number of nodes solved by LM and ML, ML failed to solve as many nodes as LM did since the overhead for checking violated inequalities was not minute.

Table 5.18: Summary of the result of level packing models for large instances.

	LM			ML		
Instance	Sol	Time	Nodes	Sol	Time	Nodes
ATP30	139,022	TL	791,397	136,502	TL	110,824
ATP31	817,795	TL	473,289	818,512	TL	54,174
ATP32	37,302	TL	583,166	37,141	TL	57,298
ATP33	233,855	TL	761,325	230,520	TL	69,519
ATP34	356,159	TL	2,047,102	355,451	TL	292,519
ATP35	612,904	TL	1,375,546	610,494	TL	196,026
ATP36	129,020	TL	1,503,956	128,651	TL	184,037
ATP37	384,266	TL	646,773	381,205	TL	81,408
ATP38	256,316	TL	758,317	254,329	TL	98,429
ATP39	265,853	TL	1,280,827	265,306	TL	155,431
ATP40	63,945	TL	461,083	63,686	TL	30,128
ATP41	202,305	25.983	47,365	202,305	186.409	28,070
ATP42	32,589	TL	311,425	32,589	TL	24,900
ATP43	208,998	TL	364,905	208,998	TL	35,623
ATP44	70,901	TL	856,106	70,541	TL	87,400
ATP45	74,205	TL	1,322,409	74,205	TL	220,750
ATP46	146,402	208.547	283,797	146,402	TL	87,531
ATP47	144,317	TL	935,946	144,317	TL	82,320
ATP48	165,428	TL	1,212,566	165,428	TL	121,872
ATP49	206,965	TL	1,898,748	206,884	TL	273,430

Table 5.19: Summary of the result of pattern-based models for large instances.

	PM(d)				SM(d, d)					SM-HA				
Instance	Sol	Time	Nodes	WP	Sol	Time	Nodes	WP	HP	Sol	Time	Nodes	WP	HP
ATP30	140,168	TL	857	399	140,168	51.793	11	111	199	140,168	29.701	11	91	115
ATP31	820,260	TL	602	179	820,260	178.159	41	147	282	820,260	136.475	41	147	214
ATP32	37,880	219.515	287	208	37,880	232.264	17	175	407	37,880	93.103	17	150	171
ATP33	235,580	134.897	227	133	235,580	35.310	1	102	143	235,580	35.997	1	106	143
ATP34	356,159	TL	1,148	223	356,159	TL	2,047	467	394	356,159	TL	2,001	337	271
ATP35	614,429	TL	1,112	335	614,429	TL	1,689	351	559	614,429	TL	1,769	360	435
ATP36	129,262	TL	1,335	504	129,262	79.619	75	96	171	129,262	65.620	75	95	124
ATP37	384,478	TL	699	340	384,478	82.556	7	119	192	384,478	111.664	9	118	239
ATP38	259,070	10.640	29	86	259,070	48.513	9	118	152	259,070	31.795	9	111	102
ATP39	266,135	TL	1,265	584	266,135	37.904	21	86	136	266,135	23.827	21	77	93
ATP40	63,945	TL	639	390	63,945	73.526	5	168	218	63,945	42.215	5	158	126
ATP41	202,305	TL	639	388	202,305	17.889	1	92	128	202,305	18.686	1	92	124
ATP42	32,589	TL	554	376	32,589	TL	374	339	703	32,589	TL	539	354	445
ATP43	208,998	TL	963	420	208,998	TL	416	371	463	208,998	TL	446	380	354
ATP44	70,901	TL	1,074	514	70,901	TL	980	400	623	70,901	TL	978	397	652
ATP45	74,205	TL	843	488	74,205	8.905	1	101	87	74,205	19.404	1	102	135
ATP46	146,402	TL	561	356	146,402	35.920	1	155	163	146,402	25.545	1	155	125
ATP47	144,317	TL	678	407	144,317	41.184	9	148	190	144,317	42.247	9	135	194
ATP48	165,428	74.745	211	131	165,428	20.436	5	96	113	165,428	17.577	5	95	91
ATP49	206,965	TL	3,065	271	206,965	93.556	447	136	140	206,965	95.602	447	132	151
Average		109.949		337		69.169		189	273		52.631		180	215

As shown in Table 5.19, staged-pattern models seem to be the most eminent approach to solve large instances than any other models. However, there exist some instances that even staged-pattern models failed to solve to optimality. Within staged-pattern models, SM-HA is superior than SM(d, d) in the aspect of time costs. SM-HA can solve 30 percent more quickly than SM(d, d) for instances that were solved to optimality. The smaller number of generated height patterns may contribute to this result.

In the respect of height patterns, the number of width patterns generated in the strip packing model outweighed it in the staged-pattern models. Although the performance of PM(d) is dominated by the performance of staged-pattern models, it is comparable to the performance of level packing models. We could conclude that there is a certain type of instance that is favorable to level packing models but not to the strip packing model since instances which LM solved to optimality were different from instances which PM(d) solved to optimality.

To compare the quality of the best solution found within the time limit, we define the IP gap as follows:

$$\text{IP gap} = \frac{\text{(Optimal objective value)} - \text{(Best Lower Bound)}}{\text{(Optimal objective value)}} \times 100(\%).$$

IP gaps for each model are summarized in Table 5.20. Although LM and ML seem to guarantee the decent quality of best solutions found, almost all of the best solutions found by pattern-based models offered the same values as the optimal objective values. This outcome shows the strength of pattern-based models in solving large instances.

Table 5.20: IP gaps for large instances.

Instance	LM	ML	PM(d)	SM(d, d)	SM-HA
ATP30	0.818	2.615	0.000	0.000	0.000
ATP31	0.301	0.213	0.000	0.000	0.000
ATP32	1.526	1.951	0.000	0.000	0.000
ATP33	0.732	2.148	0.000	0.000	0.000
ATP34	0.000	0.199	0.000	0.000	0.000
ATP35	0.248	0.640	0.000	0.000	0.000
ATP36	0.187	0.473	0.000	0.000	0.000
ATP37	0.055	0.851	0.000	0.000	0.000
ATP38	1.063	1.830	0.000	0.000	0.000
ATP39	0.106	0.311	0.000	0.000	0.000
ATP40	0.000	0.405	0.000	0.000	0.000
ATP41	0.000	0.000	0.000	0.000	0.000
ATP42	0.000	0.000	0.000	0.000	0.000
ATP43	0.000	0.000	0.000	0.000	0.000
ATP44	0.055	0.562	0.055	0.055	0.055
ATP45	0.000	0.000	0.000	0.000	0.000
ATP46	0.000	0.000	0.000	0.000	0.000
ATP47	0.000	0.000	0.000	0.000	0.000
ATP48	0.000	0.000	0.000	0.000	0.000
ATP49	0.000	0.039	0.000	0.000	0.000
Average	0.255	0.612	0.003	0.003	0.003

5.3.3 Class 5 Instances

The number of instances solved to optimality by level packing models within the time limit of 600 s are recorded in Figure 5.3 and 5.4. As N increases, the number of solved instances by level packing models decreases. Although ML requires some overheads to solve the problem, sometimes it could prove optimality better than LM. However, both level packing models seem to be substantially affected by the increase of Δ.

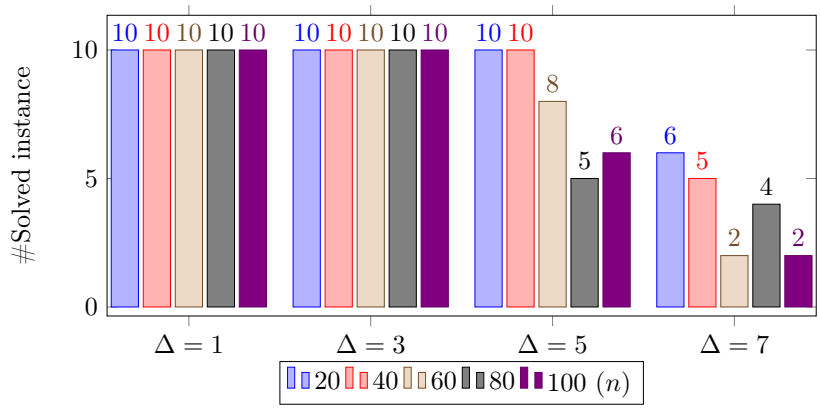


Figure 5.3: The number of solved instances by LM for Class 5.

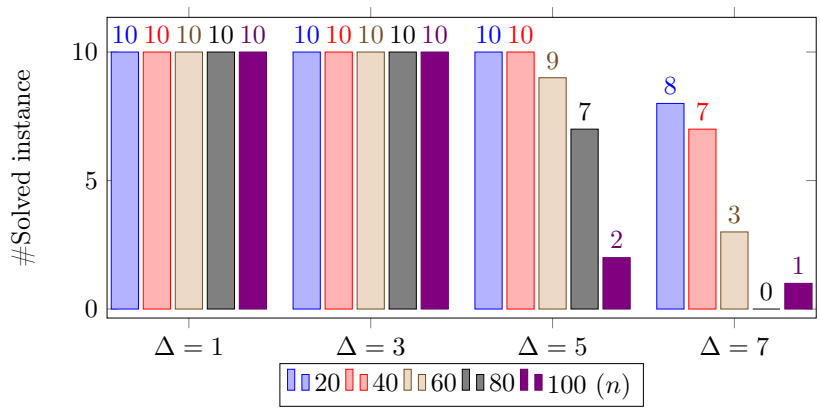


Figure 5.4: The number of solved instances by ML for Class 5.

Then, we report the result of AF and PM(d) in Figure 5.5 and 5.6, respectively. The result of AF was unexpected since the time cost for solving its LP-relaxation for other benchmark instances used to be relatively large. Even without an elaborate variable reduction technique or the branching strategy, AF could prove its effectiveness in solving a fixed, small size of instances. However, PM(d), which requires the cheapest time cost for solving the LP-relaxation among pattern-based models, sometimes failed to solve instances to optimality even for small N. Although PM(d)

did not completely solve all instances to optimality, its performance was robust to Δ. Instead, PM(d) shows more dependence on n. Another characteristic of the result of PM(d) is that the graph seems to have a V-formation pattern for fixed Δ. One hypothesis to explain this tendency is that PM(d) may be effective in solving the unconstrained instances that resemble instances with large Δ. On the other hand, no such tendency has been found in Figure 5.3.

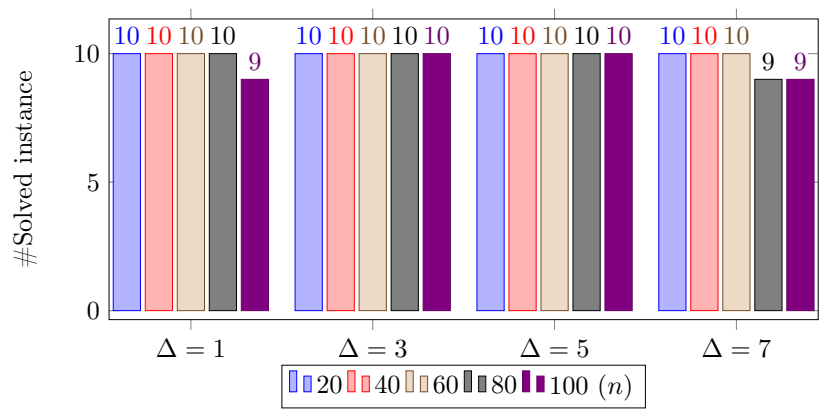


Figure 5.5: The number of solved instances by AF for Class 5.

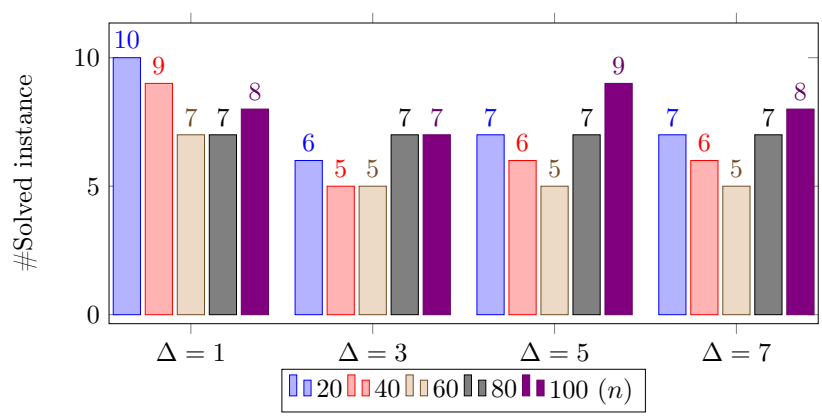


Figure 5.6: The number of solved instances by PM(d) for Class 5.

In the aspect of staged-pattern models, they were able to solve all instances

within 600 s. With analyzing the average time for computation illustrated in Figure 5.7 and 5.8, staged-pattern models seem to need more time to solve problems to optimality when n increases. Besides, higher Δ does not always lead to more computational burdensome.

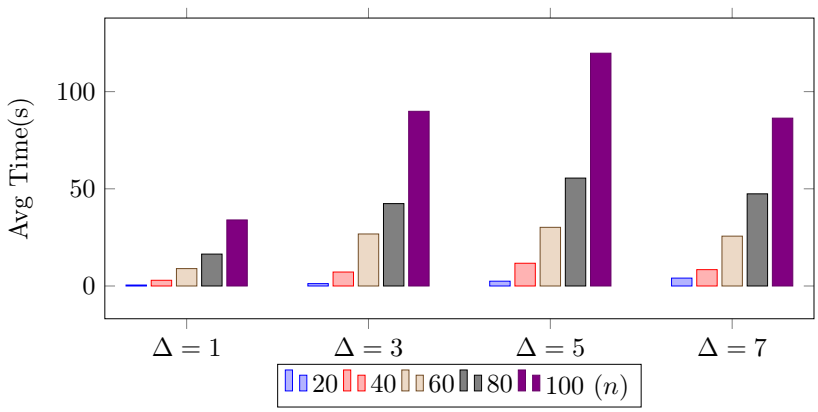


Figure 5.7: Average time costs of SM(d, d) for Class 5.

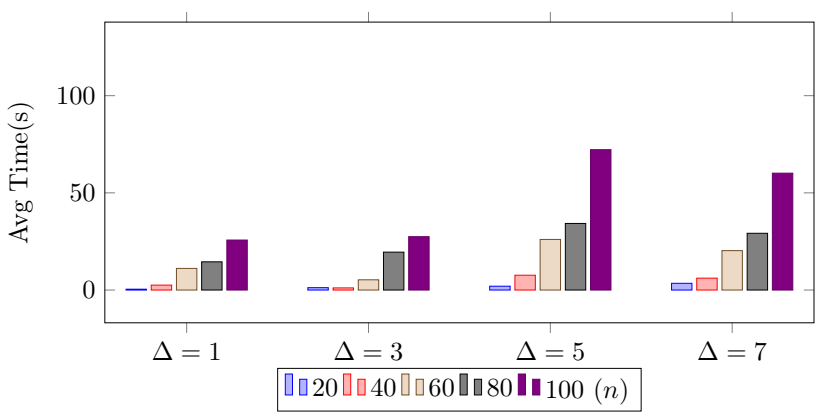


Figure 5.8: Average time costs of SM-HA for Class 5.

SM-HA proved to be the most eminent model, especially when the number of item types is substantial. When n is above a certain threshold, m will not change a lot as more items will share the same height. Therefore, it shows a more robust

graph than SM(d, d) shows.

To conclude, all models seem to be affected by the change of Δ, but the impacts were different from each other. Although level packing models offer very compact formulations for small n and Δ, their effectiveness as an exact method was countered when Δ increased. Also, we find that AF could perform well when W and H are relatively small. In the aspect of pattern-based models, their performances were rather robust than those of level packing models. The strip packing model shows a unique characteristic that favors instances with relatively large n. Other staged-pattern models show outstanding effectiveness in solving problems to optimality, which seems to depend on n, rather than Δ.

Chapter 6

Conclusion

In this thesis, we propose several integer linear programming models based on the previous studies on the 2TDK and the 2D2SP. One of the models, POLY, is the unique polynomial-size model for the 2TDK so far. We also suggest valid inequalities for the well-known level packing model, which not only enhance its LP-relaxation value but also make its structure easier to be analyzed. Then, this study establishes a nontrivial theoretical hierarchy between the modified level packing model and the pattern-based models in the aspect of their LP-relaxation values.

Utilizing these formulations to solve problems to optimality requires some techniques such as branch-and-price and branch-and-cut algorithms. For the modified level packing model, we construct a branch-and-cut algorithm. It checks whether the current solution satisfies all valid inequalities at every node. As pattern-based models have exponentially many width patterns or height patterns, several branch-and-price algorithms are devised.

To illustrate the properties of these models in their real usage, this thesis involves computational experiments of both well-known instances and artificial instances. Generally, a gap between the LP-relaxation values of level packing models and pattern-based models was far less than we had expected theoretically.

We then checked the efficiency of each exact method. For relatively small instances, level packing models outperformed other models. However, in solving larger instances, staged-pattern models proved their effectiveness. Although all pattern-based models generally provided almost near-optimal objective values within the time limit, solving instances to optimality was better achieved by staged-pattern models. Tightened upper bounds by height patterns may contribute to this result. In the aspect of demand sensitivity, level packing models had difficulty in solving instances with many duplicated items. The performance of AF and POLY in general was so poor that they may need more elaborate modifications for their real usage.

To conclude, this study extends the options for solving the 2TDK based on analysis in various situations. Theoretically, we develop an arc-flow model, staged-pattern models, and a novel polynomial-size model and prove that the upper bound provided by the staged-pattern model is nearest to the optimal objective value. Computationally, we verify that staged-pattern models implemented with branch-and-price algorithms are eminent.

For future works, the theoretical relationship between AF and ML is still unknown. In addition, as we focus on revealing some unknown properties of proposed models, more elaborate algorithms and heuristics can be devised based on our results. For example, tightening upper bounds in the branch-and-price procedure for pattern-based models may improve its performance dramatically. A direct relationship between the obtained upper bound and the objective value or approximation ratios of algorithms may also be established as Steinberg [24] did for the two-dimensional packing problem. Lastly, finding a polynomial-size formulation with a decent quality of upper bounds provided by its LP-relaxation is an intriguing issue.

Bibliography

[1] R. Alvarez-Valdes, R. Martí, J. M. Tamarit, and A. Parajón, *Grasp and path relinking for the two-dimensional two-stage cutting-stock problem*, INFORMS Journal on Computing, 19 (2007), pp. 261–272.

[2] G. Belov and G. Scheithauer, *A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting*, European Journal of Operational Research, 171 (2006), pp. 85–106.

[3] J. O. Berkey and P. Y. Wang, *Two-dimensional finite bin-packing algorithms*, Journal of the Operational Research Society, 38 (1987), pp. 423–429.

[4] V. M. Bezerra, A. A. Leao, J. F. Oliveira, and M. O. Santos, *Models for the two-dimensional level strip packing problem–a review and a computational evaluation*, Journal of the Operational Research Society, 71 (2020), pp. 606–627.

[5] A. Caprara and M. Monaci, *On the two-dimensional knapsack problem*, Operations Research Letters, 32 (2004), pp. 5–14.

[6] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer programming*, vol. 271, Springer, 2014.

[7] M. Dolatabadi, A. Lodi, and M. Monaci, *Exact algorithms for the two-dimensional guillotine knapsack*, Computers & Operations Research, 39 (2012), pp. 48–53.

[8] F. Eisenbrand and G. Shmonin, *Carathéodory bounds for integer cones*, Operations Research Letters, 34 (2006), pp. 564–568.

[9] FICO® Xpress Optimization, 2020. [Online]. `https://www.fico.com/`.

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

[11] M. R. Garey and D. S. Johnson, *Computers and intractability*, vol. 174, Freeman San Francisco, 1979.

[12] P. Gilmore and R. E. Gomory, *Multistage cutting stock problems of two and more dimensions*, Operations Research, 13 (1965), pp. 94–120.

[13] M. Hifi, *Exact algorithms for large-scale unconstrained two and three staged cutting problems*, Computational Optimization and Applications, 18 (2001), pp. 63–88.

[14] M. Hifi and R. M'Hallah, *Strip generation algorithms for constrained two-dimensional two-staged cutting problems*, European Journal of Operational Research, 172 (2006), pp. 515–527.

[15] M. Hifi and C. Roucairol, *Approximate and exact algorithms for constrained (un) weighted two-dimensional two-staged cutting stock problems*, Journal of Combinatorial Optimization, 5 (2001), pp. 465–494.

[16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*, Springer, Berlin, Germany, 2004.

[17] S.-J. Kwon, S. Joung, and K. Lee, *Comparative analysis of pattern-based models for the two-dimensional two-stage guillotine cutting stock problem*, Computers & Operations Research, 109 (2019), pp. 159–169.

[18] A. Lodi and M. Monaci, *Integer linear programming models for 2-staged two-dimensional knapsack problems*, Mathematical Programming, 94 (2003), pp. 257–278.

[19] M. E. Lübbecke and J. Desrosiers, *Selected topics in column generation*, Operations Research, 53 (2005), pp. 1007–1023.

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

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

[22] M. Mrad, I. Meftahi, and M. Haouari, *A branch-and-price algorithm for the two-stage guillotine cutting stock problem*, Journal of the Operational Research Society, 64 (2013), pp. 629–637.

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

[24] A. Steinberg, *A strip-packing algorithm with absolute performance bound 2*, SIAM Journal on Computing, 26 (1997), pp. 401–409.

[25] G. Wäscher, H. Haussner, and H. Schumann, *An improved typology of cutting and packing problems*, European Journal of Operational Research, 183 (2007), pp. 1109–1130.

국문초록

본 논문은 2단계 길로틴 절단(two-staged guillotine cut)을 사용하여 이윤을 최대화하는 2차원 2단계 배낭 문제(two-dimensional two-staged knapsack problem: 이하 2TDK)에 대한 정수최적화 모형과 최적해법을 다룬다. 우선, 본 연구에서는 스트립패킹모형, 단계패턴모형, 레벨패킹모형, 그리고 호-흐름모형과 같은 정수최적화 모형들을 소개한다. 그 뒤, 각각의 모형의 선형계획완화문제에 대해 상한강도를 이론적으로 분석하여 상한강도 관점에서 모형들 간 위계를 정립한다. 또한, 본 연구에서는 2TDK의 다항크기(polynomial-size) 모형의 존재성을 처음으로 증명한다. 다음으로 본 연구는 2TDK의 최적해를 구하는 알고리즘으로써 패턴기반모형들에 대한 분지평가 알고리즘과 레벨패킹모형을 기반으로 한 분지절단 알고리즘을 제안한다. 단계패턴모형이 이론적으로도 가장 좋은 상한강도를 보장할 뿐만 아니라, 계산 분석을 통해 단계패턴모형을 기반으로 한 분지평가 알고리즘이 제한된 시간 내 좋은 품질의 가능해를 찾음을 확인하였다.

주요어: 정수최적화 모형, 2차원 2단계 배낭 문제, 분지평가 알고리즘, 분지절단 알고리즘, 비교분석

학번: 2019-26644