Exact Approaches for Higher-Dimensional Orthogonal Packing and Related Problems

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doctor of Philosophy
(Ph.D.)

vorgelegt

der Fakultät Mathematik und Naturwissenschaften
der Technischen Universität Dresden

von

Diplom-Mathematiker, Marat A. Mesyagutov

geboren am 16. Februar 1985 in Ufa

Eingereicht am 23. September 2013.

Die Dissertation wurde in der Zeit von 02/2010 bis 08/2013
im Institut für Numerische Mathematik angefertigt.

Exact Approaches for Higher-Dimensional Orthogonal Packing and Related Problems

by

Marat A. Mesyagutov

Diploma in Mathematics

Submitted to the Faculty of Mathematics and Natural Sciences in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mathematics

at the

Dresden University of Technology

September 23, 2013

Exact Approaches for Higher-Dimensional Orthogonal Packing and Related Problems

by

Marat A. Mesyagutov

Submitted to the Faculty of Mathematics and Natural Sciences at the Dresden University of Technology on September 23, 2013 in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Mathematics

ABSTRACT

$\mathcal{NP}$-hard problems of higher-dimensional orthogonal packing are considered. We look closer at their logical structure and show that they can be decomposed into problems of a smaller dimension with a special contiguous structure. This decomposition influences the modeling of the packing process, which results in three new solution approaches.

Keeping this decomposition in mind, we model the smaller-dimensional problems in a single *position-indexed* formulation with non-overlapping inequalities serving as binding constraints. Thus, we come up with a new integer linear programming model, which we subject to polyhedral analysis. Furthermore, we establish *general non-overlapping* and *density* inequalities and prove under appropriate assumptions their facet-defining property for the convex hull of the integer solutions. Based on the proposed model and the strong inequalities, we develop a new *branch-and-cut* algorithm.

Being a relaxation of the higher-dimensional problem, each of the smaller-dimensional problems is also relevant for different areas, e.g. for scheduling. To tackle any of these smaller-dimensional problems, we use a Gilmore-Gomory model, which is a Dantzig-Wolfe decomposition of the position-indexed formulation. In order to obtain a contiguous structure for the optimal solution, its basis matrix must have a *consecutive 1's* property. For construction of such matrices, we develop new *branch-and-price* algorithms which are distinguished by various strategies for the enumeration of partial solutions. We also prove some characteristics of partial solutions, which tighten the slave problem of *column generation*.

For a nonlinear modeling of the higher-dimensional packing problems, we investigate state-of-the-art constraint programming approaches, modify them, and propose new *dichotomy* and *intersection* branching strategies. To tighten the constraint propagation, we introduce new *pruning* rules. For that, we apply 1D relaxation with *intervals* and *forbidden pairs*, an *advanced bar* relaxation, 2D *slice* relaxation, and 1D *slice-bar* relaxation with forbidden pairs. The new rules are based on the relaxation by the smaller-dimensional problems which, in turn, are replaced by a linear programming relaxation of the Gilmore-Gomory model.

We conclude with a discussion of implementation issues and numerical studies of all proposed approaches.

Thesis Supervisors:
Dr. rer. nat. Guntram Scheithauer
Prof. Dr. rer. nat. Andreas Fischer

Zugänge für die exakte Lösung höherdimensionaler orthogonaler Packungsprobleme und verwandter Aufgaben

von

Marat A. Mesyagutov

Eingereicht an der Fakultät Mathematik und Naturwissenschaften an der Technischen Universität Dresden am 23. September 2013 zur Erlangung des akademischen Grades Doctor of Philosophy in der Mathematik

ABSTRAKT

Es werden $\mathcal{NP}$-schwere höherdimensionale orthogonale Packungsprobleme betrachtet. Wir untersuchen ihre logische Struktur genauer und zeigen, dass sie sich in Probleme kleinerer Dimension mit einer speziellen Nachbarschaftsstruktur zerlegen lassen. Dies beeinflusst die Modellierung des Packungsprozesses, die ihreseits zu drei neuen Lösungsansätzen führt.

Unter Beachtung dieser Zerlegung modellieren wir die Probleme kleinerer Dimension in einer einzigen *positionsindizierten* Formulierung mit Nichtüberlappungsungleichungen, die als Bindungsbedingungen dienen. Damit entwickeln wir ein neues Modell der ganzzahligen linearen Optimierung und unterziehen dies einer Polyederanalyse. Weiterhin geben wir *allgemeine Nichtüberlappungs-* und *Dichtheitsungleichungen* an und beweisen unter geeigneten Annahmen ihre facettendefinierende Eigenschaft für die konvexe Hülle der ganzzahligen Lösungen. Basierend auf dem vorgeschlagenen Modell und den starken Ungleichungen entwickeln wir einen neuen *Branch-and-Cut-Algorithmus*.

Jedes Problem kleinerer Dimension ist eine Relaxation des höherdimensionalen Problems. Darüber hinaus besitzt es Anwendungen in verschiedenen Bereichen, wie zum Beispiel im Scheduling. Für die Behandlung der Probleme kleinerer Dimension setzen wir das Gilmore-Gomory-Modell ein, das eine Dantzig-Wolfe-Dekomposition der positionsindizierten Formulierung ist. Um eine Nachbarschaftsstruktur zu erhalten, muss die Basismatrix der optimalen Lösung die *consecutive-1's*-Eigenschaft erfüllen. Für die Konstruktion solcher Matrizen entwickeln wir neue *Branch-and-Price-Algorithmen*, die sich durch Strategien zur Enumeration von partiellen Lösungen unterscheiden. Wir beweisen auch einige Charakteristiken von partiellen Lösungen, die das Hilfsproblem der *Spaltengenerierung* verschärfen.

Für die nichtlineare Modellierung der höherdimensionalen Packungsprobleme untersuchen wir moderne Ansätze des Constraint Programming, modifizieren diese und schlagen neue *Dichotomie-* und *Überschneidungsstrategien* für die Verzweigung vor. Für die Verstärkung der Constraint Propagation stellen wir neue *Ablehnungskriterien* vor. Wir nutzen dabei 1D Relaxationen mit *Intervallen* und *verbotenen Paaren*, *erweiterte* Streifen-Relaxation, 2D *Scheiben-Relaxation* und 1D *Scheiben-Streifen-Relaxation* mit verbotenen Paaren. Alle vorgestellten Kriterien basieren auf Relaxationen durch Probleme kleinerer Dimension, die wir weiter durch die LP-Relaxation des Gilmore-Gomory-Modells abschwächen.

Wir schließen mit Umsetzungsfragen und numerischen Experimenten aller vorgeschlagenen Ansätze.

Betreuer:
Dr. rer. nat. Guntram Scheithauer
Prof. Dr. rer. nat. Andreas Fischer

Some parts of the research work presented in this thesis have been published or submitted and is currently under the reviewing process in the following:

1. Peer-reviewed publications:

   [MMBS11] M. Mesyagutov, E. Mukhacheva, G. Belov, and G. Scheithauer. Packing of one-dimensional bins with contiguous selection of identical items: An exact method of optimal solution. *Automation and Remote Control*, 72:141–159, 2011.
   [Chapter 1]

   [MSB12a] M. Mesyagutov, G. Scheithauer, and G. Belov. LP bounds in various constraint programming approaches for orthogonal packing. *Computers and Operations Research*, 39(10):2425–2438, 2012.
   [Chapter 3]

2. Research papers (preprints):

   [MSB12b] M. Mesyagutov, G. Scheithauer, and G. Belov. New constraint programming approaches for the 3D orthogonal packing. Technical report, Preprint MATH-NM-01-2012, Technische Universität Dresden, 2012.
   To be submitted.
   [Chapter 4]

   [MSB13a] M. Mesyagutov, G. Scheithauer, and G. Belov. A new branch-and-cut method for the strip packing problem. Technical report, Preprint MATH-NM-04-2013, Technische Universität Dresden, 2013.
   To be submitted.
   [Chapter 1]

   [MSB13b] M. Mesyagutov, G. Scheithauer, and G. Belov. New branch-and-price methods for the 1D contiguous bin packing problem. Technical report, Preprint MATH-NM-06-2013, Technische Universität Dresden, 2013.
   To be submitted.
   [Chapter 2]

Acknowledgments

My first thanks go to my advisor, Guntram Scheithauer, for the support, motivation, and kindness. I met him firstly while I did my first internship within the university studies at the Dresden University of Technology at the well-known scientific school of cutting and packing. After the internship, I realized that I really want to learn the know-how's and gain the experience here – that was my motivation for the continuation. Our sometimes daily meetings always inspired me to work twice, thrice more effective. His scientific experience and intuition have always guided to a right direction. I value very much those numerous soft skills which I learned from him as, e.g. to be extremely precise in details and that not only in math and science.

I would like to thank my advisor, Andreas Fischer. The last phase of the work in particular could not be possible without his support. The project which I am grateful to be a part of helped me a lot and broadened my experience and outlook. I am thankful to him for the fruitful and valuable discussions. Being a type of persons who dig into details, he carries also about generality – the values I learned.

I am very thankful to my friend and scientific mentor, Gleb Belov. I still do not know a person who has so much practical and implementation experience in cutting and packing. Literally, it was sometimes like: "Hey, Gleb, I have a brilliant idea which can crack this problem". But after explaining him my idea, I found out that he already tried it and the idea did not work. Gleb is a hard working person. He taught me how to be goal focused. I would like to thank him for introducing me to the world of hiking. We spent a lot of time hiking across the beautiful country Germany. I still remember the great hike experience to the top of the Zugspitze and the Pico del Teide (Spain).

I want particularly thank Elita Mukhacheva, my first teacher and deliberate life mentor, who played a big role in my life. She introduced me to the problems of cutting and packing. While I studied at the university, she organized weekly seminars on combinatorial problems, which I attended with a great interest. At the same time, I became involved in an effective implementation of some algorithms where I gained the first experience in this area. Later on, I improved these algorithms and studied new ones. My first publication with her co-authorship she reiterated multiple times, I believe this number was between five and ten. She was always supportive and full of energy. I learned how to be absolutely devoted to any work I do, as she was. The fact that she was correcting the master thesis of a student of hers just a few hours before she passed away after a long-lasting illness, says a lot about her.

I am very grateful to Nafisa Yusupova. Being a dean of the faculty at the university, she held an exciting lecture on the information theory where I firstly met her. Later on, she initiated that internship where the relationship to the Dresden school has started. She is an intelligent and very energetic person. I value very much all efforts from her side to connect the scientists from Russia with other countries. She is one of the fewer people who do a real work in this direction.

I would like to thank Vadim Kartak for productive talks we had at numerous meetings. My special thanks go to François Clautiaux for fruitful discussions and experience sharing at many conferences.

I would like express my appreciation to the colleagues from the Institute of Numerical Mathematics at the university. They were always supportive and kind to me.

I would not be where I am without my smart, interesting and inspiring friends and those who cares about me and played an important role. I am thankful to everyone who supported me. Special thanks go to my best friend, Igor, who is always supportive, understanding and willing to help.

Most importantly, I would like truly to thank my beloved family for their love and unlimited support. My mother, Fatima, is a wonderful person, whom I love. She cares about all members of the family and works as a doctor. She has an amazing merit of treating and wining people's favour, which I have been learning from her so far. My grandfather, Buljak, is as my father. His is an example of a person for me who follows his line. Being a director of a small factory, he taught me how to achieve results. Despite his non-emotionality, he does everything for the family. My grandmother, Farida, has selflessly devoted herself to the family. She is a wise and very kind person. My beloved girlfriend, Liliya, is a marvelous person. She has an extraordinary combination of intelligence, wisdom, and beauty. I know that I can always rely on her. I feel lucky to have such an amazing family. I love every one of them very much and hope that I will always make them proud.

Finally, I would like to acknowledge all the sources of funding that made my research possible (in the order of granting): German Academic Exchange Service (DAAD research grant[1]*); the Saxon State Ministry of Science and the Arts (Georgius-Agricola scholarship*[2]*); the European Union (Erasmus-Mundus doctoral scholarship*[3]*); German Research Foundation (DFG*[4]*).*

Marat A. Mesyagutov
Dresden, Aug. 8, 2013

[1] 325 A/06/92734
[2] 518-VO
[3] EM ECW-L04 TUD 08-134
[4] Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing"

Contents

Introduction

Cutting and packing problems are $\mathcal{NP}$-hard combinatorial optimization problems, which arise in context of many real-world applications in industry as well as in service. Simple in verbal formulation and very hard to solve, cutting and packing problems have many interdisciplinary contributions from mathematics, management science, operations research, engineering, and computer science. All having a similar logical structure – large objects must be cut optimally into smaller pieces for cutting, and small items must be allocated optimally in larger objects for packing, they find their applications in supply chain management, finance, investment, and transportation. Cutting and packing problems are also of interest from the theoretical side, since the question $\mathcal{P} \neq \mathcal{NP}$ remains still open.

The central problems in the thesis are of two types, *optimization* and *decision*. The former is formulated in a way of finding an optimal configuration. The latter only asks whether a configuration exists satisfying all given requirements. More specifically, we consider the following decision and optimization problems. For d-dimensions, $d = 2, 3, \ldots$, consider a set of m d-dimensional items with sizes $(w_i^1, \ldots, w_i^d)$, $i \in I := \{1, \ldots, m\}$:

1. The *d-dimensional orthogonal packing feasibility* problem (OPP-d) asks whether all the items can be orthogonally packed into a given container $(W^1, \ldots, W^d) \in \mathbb{Z}_+^d$ where $w_i^k \in \{1, \ldots, W^k\}$ for $i \in I$ and $k \in \{1, \ldots, d\}$.

2. The *d-dimensional strip packing* problem (SPP-d) asks for a packing of all the items into a d-dimensional semi-infinite strip $(W^1, \ldots, W^{d-1}) \in \mathbb{Z}_+^{d-1}$ which occupies the minimal length in W^d where $w_i^k \in \{1, \ldots, W^k\}$, $w_i^d \in \mathbb{Z}_+$ for $i \in I$ and $k \in \{1, \ldots, d-1\}$.

3. The 1-*dimensional contiguous bin packing* problem (CBPP-1) asks for a packing of m types of 1D items with a length w_i^1 and a quantity w_i^2, $i \in I := \{1, \ldots, m\}$ into 1D bins with a length $W^1 \in \mathbb{Z}_+$ in a fixed order which uses the minimal number of the bins satisfying the following restrictions: 1) Each item of one type is packed at most once into a bin; 2) Each item of one type is packed contiguously into a sequence of the bins; where $w_i^1 \in \{1, \ldots, W^1\}$ for $i \in I$.

For the decision version of the optimization problems the suitable answer would be "no" if no configuration exists satisfying all the requirements or "yes" with the items positions in the container which fulfill all the requirements. The answer for the optimization problems would be items positions in the container which realize the optimal configuration.

With the "higher-dimensional" packing in the title of the theses we refer to the problems with $d \geq 2$. Note, the CBBP-1 is not purely 1-dimensional, since the "contiguity" constraint and a fixed order of the bins can be interpreted using an additional dimension.

The thesis consists of four chapters. Each chapter is a separate paper and represents an independent method for the problem to solve.

Chapter 1 deals with SPP-2. We develop an Integer Linear Programming (ILP) model and subject it to polyhedral analysis. We determine *non-overlapping* and *density* constraints and prove under appropriate assumptions their facet-defining property for the convex hull of the integer solutions. We also consider some valid linear and non-linear inequalities. For the non-linear ones we develop a proper linearization if possible. Based on the proposed formulation and strong inequalities, we develop a new *branch-and-cut* algorithm and study it numerically.

Chapter 2 deals with CBPP-1. We use a Gilmore-Gomory model for the 1D bin packing problem and use the known fact that, if a column set has a *consecutive 1's* property (C1P), then the corresponding matrix is total unimodular, and thus all the corners of the polyhedron of its linear programming (LP)-relaxation are integral for the integer input data. For the construction of such column sets, we develop branch-and-price algorithms with two strategies: *column-*, and *subcolumn*-based enumerations. We also prove propositions about some *characteristics* of a specific or an arbitrary column which breaks the C1P of a given column set. For each of these characteristics, we develop an algorithm, the output data of which is used in the slave problem of the *column generation* thus tightening the bound. Concluding the chapter, we report the results of a numerical study of the proposed algorithms.

Chapter 3 deals with OPP-2 and SPP-2. We investigate the state-of-the-art constraint programming (CP) approaches for OPP-2, in particular the *basic fixation* strategy (fix at the lower bound or increase the lower bound for the variable domain, also known as "schedule or postpone") and *disjunctive* branching strategy, and propose new *dichotomy* and *intersection* strategies. We also propose new pruning rules based on tightened 1D relaxations of various kinds, e.g. *intervals*, *forbidden pairs*, and *advanced bar* relaxations. The new pruning rules are adapted into the *constraint propagation* process of the CP. Using the *dichotomic search* procedure, the developed methods for OPP-2 are transformed for SPP-2. The input data for the bar relaxation is obtained from the local partial solution, the information from the constraint propagation procedure, and the relative positions of the items in the container. The numerical results demonstrate the efficiency of the proposed strategies and of the combination of the CP and the LP-based pruning rules.

Chapter 4 deals with OPP-3 and SPP-3. Inspired by the results from Chapter 3, we investigate, modify and transform them into solution methods for the 3D case. We discuss basics of the algorithm and propose some minor modifications as *preprocessing* and consideration of *raster points*. We compare the *basic branching* strategy with the most successful according to the one from Chapter 3, *disjunctive* strategy. We also propose new pruning rules based on the geometrical and LP-relaxations of four types: a *simple* 1D *bar* relaxation with different stock lengths, a 1D *bar* relaxation, a 2D *slice* relaxation and a 1D *slice-bar* relaxation with forbidden item pairs. The performed

numerical study shows high efficiency of the propagated on 3D approach and of the geometrical and further relaxed by LP pruning rules.

The summery and outlook follow and conclude the thesis.

Chapter 1

A Branch-and-Cut Method for the Strip Packing Problem

We consider the 2D strip packing problem (SPP-2). Given a set of rectangular items, SPP-2 is to find a packing of all items occupying the minimal height of the given semi-infinite strip. We develop an Integer Linear Programming (ILP) model and subject it to polyhedral analysis. We determine non-overlapping and density constraints and prove under appropriate assumptions their facet-defining property for the convex hull of the integer solutions. We also consider some valid linear and non-linear inequalities. For the non-linear ones we develop a proper linearization if possible. Based on the proposed formulation and facets we develop a new branch-and-cut algorithm. Numerical results are presented.
Keywords: linear programming, branch-and-cut, facet-defining inequalities

1.1 Introduction

Let us consider a set of m rectangular items (w_i, h_i), with $i \in I := \{1, \ldots, m\}$. The 2-*dimensional orthogonal strip packing problem* (SPP-2) [LMV02] asks for a packing of items (w_i, h_i) with $i \in I$ without rotations into the semi-infinite strip with width W which occupies the minimal height. The guillotine constraint [MAVdC10, CJM08] is not considered. All input data are positive integers, i.e., $W \in \mathbb{Z}_+$, and $w_i \in \{1, \ldots, W\}$ for $i \in I$, $h := \{h_1, \ldots, h_m\} \in \mathbb{Z}_+^m$. The problem can be formulated also for the 3D case. If dimension of a problem is not relevant then it will be referenced as SPP further in this paper.

1.1.1 Modeling and notations

Let H be a feasible upper bound on the optimal value of the height of the strip in SPP-2. In order to formulate a model we consider the following solution representation. To the strip we apply a 2-dimensional grid with step 1, each point of which (u, v), $u \in U := \{1, \ldots, H\}$, $v \in V := \{1, \ldots, W\}$ can be covered by an item $i \in I$.

Let us introduce binary variables α_i^u, $i \in I$, $u \in U$ which indicate in the case of $\alpha_i^u = 1$ the minimal coordinate u of item i in the packing over the $(0, H)$-axis. Analogously, we introduce variables β_i^v, $i \in I$, $v \in V$ for the packing over the $(0, W)$-axis. That means, if $\alpha_i^u = \beta_i^v = 1$ then item i covers the rectangular region $\{(x, y) : u - 1 < x \leq u - 1 + h_i,\ v - 1 < y \leq v - 1 + w_i\}$, and (u, v) is called the allocation point of item i or coordinates of item i. If one of the variables α_i^j, $j = u - h_i + 1, \ldots, u$ is not 0, then item $i \in I$ intersects the coordinate $u \in U$. To simplify the description we introduce the following sets:

$$H_i(u) := \{\max\{1, u - h_i + 1\}, \ldots, \min\{u, H - h_i + 1\}\}, \quad u \in U;$$
$$W_i(v) := \{\max\{1, v - w_i + 1\}, \ldots, \min\{v, W - w_i + 1\}\}, \quad v \in V.$$

Thus, if $\alpha_i^u = 1$ then item i intersects all coordinates $k \in \{u, \ldots, u + h_i - 1\}$. Let

$$\tilde{\alpha}_i^u = \sum_{k \in H_i(u)} \alpha_i^k; \quad \tilde{\beta}_i^v = \sum_{k \in W_i(v)} \beta_i^k; \quad u \in U,\ v \in V.$$

The set of item pairs which fit together in both directions is denoted by

$$\mathcal{P} := \{(p, q) \in I \times I :\ p < q \wedge w_p + w_q \leq W \wedge h_p + h_q \leq H\}.$$

Let $H = \sum_{i \in I} h_i$ and f be the value (height) of a solution. We obtain the following Integer Linear Programming (ILP) model of SPP-2:

$$f \to \min; \quad \text{s.t.} \tag{1.1}$$
$$\sum_{u \in U} \alpha_i^u = 1, \quad i \in I; \tag{1.2}$$
$$\sum_{v \in V} \beta_i^v = 1, \quad i \in I; \tag{1.3}$$
$$\sum_{i \in I} w_i \tilde{\alpha}_i^u - W \leq 0, \quad u \in U; \tag{1.4}$$
$$\sum_{i \in I} h_i \tilde{\beta}_i^v - f \leq 0, \quad v \in V; \tag{1.5}$$
$$\tilde{\alpha}_p^u + \tilde{\alpha}_q^u + \tilde{\beta}_p^v + \tilde{\beta}_q^v \leq 3, \quad u \in U,\ v \in V,\ (p, q) \in \mathcal{P}; \tag{1.6}$$
$$\alpha_i^u,\ \beta_i^v \in \{0, 1\}, \quad i \in I,\ u \in U,\ v \in V. \tag{1.7}$$

The above formulation has $O(m(W + H))$ variables and $O(m^2WH)$ constraints. Note the formulation (1.1)-(1.7) can be referred as to a *position-indexed* formulation of SPP-2, since we define 0-1 decision variables for every placement position. Such kind of models were firstly proposed by L.V. Kantorovich for different problems in production [Kan39].

Lemma 1.1. *Model* (1.1)-(1.7) *is an exact formulation of SPP-2.*

Proof. In both directions has to be proven: one-one mapping of a feasible solution and variable values; one-one mapping of variable values and a feasible solution.

From one hand, from a feasible packing, i.e., $\{(x_i, y_i) : \ i \in I\}$, where the x_i- and y_i-coordinates are minimal, we obtain the corresponding values of variables:

$$\alpha_i^j = \begin{cases} 1, \ j = x_i + 1; \\ 0, \ \text{otherwise}; \end{cases} \qquad \beta_i^j = \begin{cases} 1, \ j = y_i + 1; \\ 0, \ \text{otherwise}. \end{cases}$$

From the other hand, based on variables we can build two graphs $G_d = (I, E_d)$, $d = 1, 2$, where

$$E_1 = \{(i,j) \in I \times I : \ i \neq j, \ \exists u \in U, \ \tilde{\alpha}_i^u = \tilde{\alpha}_j^u = 1\};$$

$$E_2 = \{(i,j) \in I \times I : \ i \neq j, \ \exists v \in V, \ \tilde{\beta}_i^v = \tilde{\beta}_j^v = 1\}.$$

Each of these graphs have the following three properties:

1. G_1 and G_2 are interval graphs.

2. Each stable set of G_1 and G_2 are y and x-feasible, respectively.

3. $E_1 \cap E_2 = \emptyset$.

The first property holds by construction of graphs. The second holds because variables satisfy constraints (1.4)-(1.5). The last property holds because variables satisfy constraints (1.6). Since all properties hold then from the Theorem 1 [FSvdV07] follows the lemma. □

Modeling with rotations

We do not allow the rotation of items by 90°. But it is possible to model the rotations by introducing additional variables which duplicate α_i^u and β_i^v, variables a_i^u and b_i^v. Variables a_i^u, $i \in I$, $u \in U$, in contrast to α_i^u, indicate in case of $a_i^u = 1$ the minimal coordinate u of rotated by 90° item i in the packing over the $(0, H)$-axis. Analogously, we introduce variables b_i^v, $i \in I$, $v \in V$ for the packing of rotated by 90° item i over the $(0, W)$-axis. Similarly to $H_i(u)$ and $W_i(v)$ we introduce the following sets:

$$\overline{H}_i(u) := \{\max\{1, u - w_i + 1\}, \ldots, \min\{u, H - w_i + 1\}\}, \quad u \in U;$$
$$\overline{W}_i(v) := \{\max\{1, v - h_i + 1\}, \ldots, \min\{v, W - h_i + 1\}\}, \quad v \in V.$$

Thus, if $a_i^k = 1$ then item i intersects all coordinates $k \in \{u, \ldots, u + w_i - 1\}$. Let

$$\tilde{a}_i^u = \sum_{k \in \overline{H}_i(u)} a_i^k; \quad \tilde{b}_i^v = \sum_{k \in \overline{W}_i(v)} b_i^k; \quad u \in U, \ v \in V.$$

Assigning a value to variables α_i^u, a_i^u, β_i^v, b_i^u and f, we obtain the following ILP

model of SPP-2 with rotations by 90°:

$$f \to \min; \quad \text{s.t.} \tag{1.8}$$

$$\sum_{u \in U} \{\alpha_i^u + a_i^u\} = 1, \quad i \in I; \tag{1.9}$$

$$\sum_{v \in V} \{\beta_i^v + b_i^u\} = 1, \quad i \in I; \tag{1.10}$$

$$\sum_{i \in I} \{w_i \tilde{\alpha}_i^u + h_i \tilde{a}_i^u\} - W \le 0, \quad u \in U; \tag{1.11}$$

$$\sum_{i \in I} \{h_i \tilde{\beta}_i^v + w_i \tilde{b}_i^v\} - f \le 0, \quad v \in V; \tag{1.12}$$

$$\sum_{i \in \{p,q\}} \{\tilde{\alpha}_i^u + \tilde{a}_i^u + \tilde{\beta}_i^v + \tilde{b}_i^v\} \le 3, \quad (u,v) \in U \times V,\ (p,q) \in \mathcal{P}; \tag{1.13}$$

$$f \ge 0; \tag{1.14}$$

$$\alpha_i^u,\ \beta_i^v,\ a_i^u,\ b_i^v \in \{0,1\}, \quad i \in I,\ u \in U,\ v \in V. \tag{1.15}$$

Further in this paper we consider SPP-2 without rotations.

1.1.2 Overview of solution methods

There exist ILP models [Bea85, Pad00, BB07, BKRS09] for SPP-2 or similar problems based on different representations of a feasible solution. For some of them, the exact solution is difficult because of the weak LP bounds, i.e., [Pad00], quadratic number of intersection variables and/or pseudo-polynomial number of position-indexed variables, i.e., [Bea85, BB07].

In [HNS08], the problem is considered in the 1D case without connecting inequalities (1.6). The resulting facet-defining inequalities are subject to study in the 2D case, but this is not the aim of this paper. In [MMBS11], the 1D case is also considered and handled by a branch-and-bound with lower bounds based on the LP relaxation of the decomposed model of the 1D bin packing problem which is solved by column generation method.

The proposing formulation (1.1)-(1.7) has some of the discussed drawbacks but in the subsequent we consider some approaches in order to tackle them, see 1.1.4.

1.1.3 Polyhedra, faces and facets

Here we define briefly some notations which are used in the subsequent sections. Thereby we follow the terminology given in [NW88].

Let $\gamma \in \mathbb{R}^k$. A set of points $x^1, \ldots, x^k \in \mathbb{R}^n$ is *affinely independent* if the unique solution of $\gamma^T (x_1, \ldots, x_k)^T = 0$, $\gamma^T \mathbf{1} = 0$ is $\gamma = 0$.

A *polyhedron* $P \subset \mathbb{R}^n$ is the set of points that satisfy a finite number of linear inequalities, that is, $P = \{x \in \mathbb{R}^n : \ Ax \le b\}$ with an $m \times n$-matrix A. A polyhedron $P \subset \mathbb{R}^n$ is *bounded* if there exists a constant K such that $|x_i| < K\ \forall x \in P$, $i = 1, \ldots, n$. A bounded polyhedron is called a *polytope*.

A polyhedron P is of *dimension* k, denoted by $\dim(P) = k$, if the maximum number of affinely independent points in P is $k + 1$.

The set $\{x \in \mathbb{R}^n : \pi^T x = \pi_0\}$ is called a *hyperplane*. The set $\{x \in \mathbb{R}^n : \pi^T x \leq \pi_0\}$ is called a *half-space*. The inequality $\pi^T x \leq \pi_0$ is called a valid inequality for P if it is satisfied by all points in P, i.e. $\pi^T x \leq \pi_0 \ \forall x \in P$.

Let $\pi^T x \leq \pi_0$ be a valid inequality for P and let $F = \{x \in P : \pi^T x = \pi_0\}$. F is called a *face* of P, and we say that $\pi^T x \leq \pi_0$ *represents* F. A face F is said to be *proper* if $F \neq 0$ and $F \neq P$.

A face F of P with $\dim(F) = \dim(P) - 1$ is called a *facet* of P. If $\pi^T x \leq \pi_0$ represents a facet of P then $\pi^T x \leq \pi_0$ is called a *facet-defining inequality*.

A set $S \subseteq \mathbb{R}^n$ is *convex* if $\forall x, y \in S$, $\gamma \in [0, 1]$, we have $\gamma x + (1 - \gamma)y \in S$. Let $x^1, \ldots, x^k \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}^k$ be given such that $\gamma^T \mathbf{1} = 1$ then the vector $\sum_{i=1}^k \gamma_i x^i$ is said to be a *convex combination* of $x^1, \ldots, x^k$; the *convex hull* of $x^1, \ldots, x^k$ is the set of all convex combinations of these vectors.

If $\pi x \leq \pi_0$ and $\mu x \leq \mu_0$ are two valid inequalities for $P \subseteq \mathrm{R}^n_+$, $\pi x \leq \pi_0$ *dominates* $\mu x \leq \mu_0$ if there exists $s > 0$ such that $\pi \geq s\mu$ and $\pi_0 \leq s\mu_0$, and $(\pi, \pi_0) \neq (s\mu, s\mu_0)$.

1.1.4 Our contribution

In order to solve SPP-2 we investigate the ILP model which is proposed in Section 1.1. In Section 1.2 we look closer at the non-overlapping constraints, propose general non-overlapping and density inequalities and prove under appropriate assumptions their facet-defining property. Based on the proposed formulation and constraints we develop a new branch-and-cut algorithm, Section 1.3. In section 1.4 we consider valid inequalities which can be applied within the formulation in order to reduce the size of the branching tree and exclude equivalent solutions. In section 1.5 we consider feasible constraints which are nonlinear, but in some cases can be approximated by extra variables and linear constraints. The final part of the paper reports numerical results and conclusion.

1.2 Strong valid inequalities and facets

In this section we consider four classes of facet-defining inequalities: non-overlapping, cover and density inequalities. For some classes we show the proof of their facet-defining property.

Now we give some notations and preliminary information. For the case of SPP-2 in formulation (1.1)-(1.7) let in the following

$$P := \{(\alpha, \beta) \in \mathbb{R}_+^{m(W+H)} : \ (1.2) - (1.6) \text{ hold}\}$$

denote the polyhedron of the continuous relaxation of SPP-2. Furthermore, let

$$S := \{(\alpha, \beta) \in \{0, 1\}^{m(W+H)} : \ (1.2) - (1.6) \text{ hold}\}$$

be the set of feasible solutions, and let $\mathrm{conv}(S)$ be the convex hull of S.

In order to get a full description of $\mathrm{conv}(S)$ by linear inequalities it is sufficient to use the facet-defining inequalities. In the following we develop some classes of such inequalities.

In order to simplify the following description we introduce the following sets:

$$\bar{U}_i := \{H - h_i + 2, \dots, H\}, \quad \bar{V}_i := \{W - w_i + 2, \dots, W\}. \tag{1.16}$$

where $\bar{U}_i$ and $\bar{V}_i$ contain the coordinates of an item i where it cannot be allocated.

Lemma 1.2. *Let $H := 2H_0 + h_{\max} - 1$ where H_0 is the value of an optimal solution. In case of SPP-2 in formulation* (1.1)-(1.7) *we have*

$$\dim(\mathrm{conv}(S)) = m(W + H) - \sum_{i \in I}[w_i + h_i].$$

Proof. Any solution $(\alpha, \beta) \in \{0,1\}^{m(W+H)}$ fulfills the $2m$ equalities (1.2)-(1.3). That means, $\dim(\mathrm{conv}(S)) \leq m(W + H) - 2m$. Last allocation points for an item i are $W - w_i + 1$ and $H - h_i + 1$, respectively, which equivalent to equations $\alpha_i^u = 0$, $u \in \bar{U}_i$, and $\beta_i^v = 0$, $v \in \bar{V}_i$ in the formulation. Herewith, $\dim(\mathrm{conv}(S)) \leq m(W + H) - 2m - \sum_{i \in I}[|U_i| + |V_i|]$ or $\dim(\mathrm{conv}(S)) \leq m(W + H) - 2m - \sum_{i \in I}[h_i - 1 + w_i - 1]$, hence $\dim(\mathrm{conv}(S)) \leq m(W + H) - \sum_{i \in I}[w_i + h_i]$.

Further we construct $1 + m(W + H) - \sum_{i \in I}[w_i + h_i]$ affinely independent points. Let $(\alpha, \beta)_0 := (\alpha_1^1, \dots, \alpha_1^H, \dots, \alpha_m^1, \dots, \alpha_m^H, \beta_1^1, \dots, \beta_1^W, \dots, \beta_m^1, \dots, \beta_m^W)$ be an optimal solution with $f := H_0$ and $\alpha_i^{k_i} := 1$, $\beta_i^{l_i} := 1$, and $\alpha_i^k := 0$ for $k \in U \backslash \{k_i\}$, $\beta_i^l := 0$ for $l \in V \backslash \{l_i\}$, where k_i and l_i are the positions of items in the optimal solution in $(0, H)$ and $(0, W)$, respectively. For the graphical representation, please refer to Fig. 1.1a.

Let $\bar{k} := H_0 + h_{\max} - 1$. Now we construct the first $mH - \sum_{i \in I} h_i$ points by shifting of each item over the $(0, H)$-axis. The shifting is divided into two phases: the first, we shift an item i along coordinates 1 up to H_0 skipping its original k_i; the second, we shift it along coordinates $H_0 + 1$ up to $H - h_i + 1$. For the first phase we put the rest of the items after the coordinate $\bar{k}$, and at the origin for the second phase, Fig. 1.1c-1.1d. For each $s \in I$:

$$f := 2H_0 + h_{\max} - 1,$$
$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k \in \{1, \dots, H_0\} \setminus \{k_i\}; \\ 1, \ i \in I \backslash \{s\}, \ k = \bar{k} + k_i; \\ 0, \ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1, \ l = l_i; \\ 0, \ \text{otherwise}. \end{cases} \tag{1.17}$$

For each $s \in I$:

$$f := 2H_0 + h_{\max} - 1,$$
$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k \in \{H_0 + 1, \dots, H\} \setminus \bar{U}_i; \\ 1, \ i \in I \backslash \{s\}, \ k = k_i; \\ 0, \ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1, \ l = l_i; \\ 0, \ \text{otherwise}. \end{cases} \tag{1.18}$$

where U_i is defined as (1.16). The number of points (1.17) is $\sum_{i \in I}[H_0 - 1] = m(H_0 - 1)$. The number of points (1.18) is $\sum_{i \in I}[H - H_0 - |\bar{U}_i|] = \sum_{i \in I}[H - H_0 - (h_i - 1)] = m(H - H_0 + 1) - \sum_{i \in I} h_i$. So, the number of points which arise from shifting of items over the $(0, H)$-axis is $mH - \sum_{i \in I} h_i$.

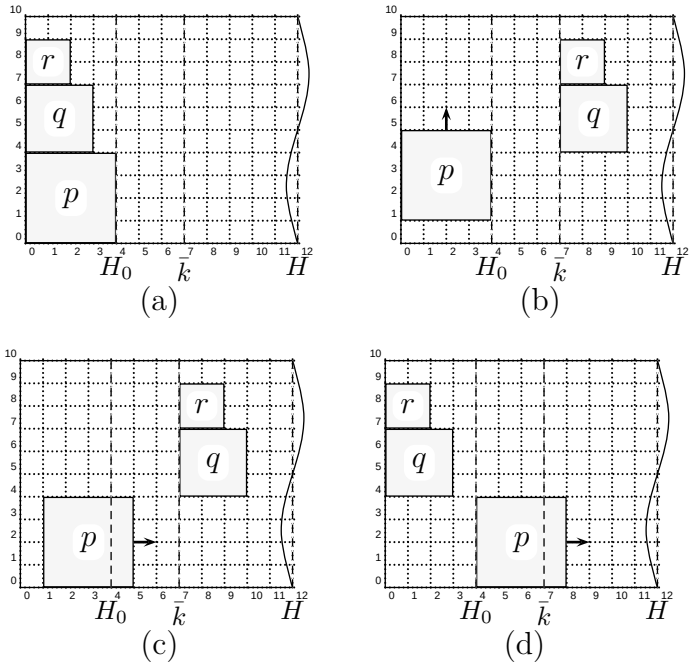


Figure 1.1: Feasible shifting of items for construction of affinely independent points in conv(S): (a) – An optimal solution; (b) – Shifting of item p over the $(0, W)$-axis; (c)-(d) – Shifting of item p over the $(0, H)$-axis. Note for all figures here and further we draw the strip rotated by 90° clockwise.

Now we construct the last $mW - \sum_{i\in I} w_i$ points by shifting of each item over the $(0, W)$-axis. While we shift an item i along coordinates 1 up to $W - w_i + 1$ skipping the origin l_i, we put the rest of the items after the coordinate $\bar{k}$ over $(0, H)$-axis, Fig. 1.1b. For each $s \in I$:

$$f := 2H_0 + h_{\max} - 1,$$
$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k = k_i; \\ 1,\ i \in I\backslash\{s\},\ k = \bar{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = s,\ k \in V\backslash(\bar{V}_i \cup \{l_i\}); \\ 1,\ i \in I\backslash\{s\},\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.19}$$

where V_i is defined as (1.16). The number of points (1.19) is $\sum_{i\in I}[W - 1 - |\bar{V}_i|] = \sum_{i\in I}[W - 1 - (w_i - 1)] = mW - \sum_{i\in I} w_i$.

Now we show the affinely independence of the introduced points. Firstly, we simplify the notation. Let $\hat{\imath}^k \in \{0, 1\}^H$ and $\hat{\jmath}^l \in \{0, 1\}^W$ be the k- and l-th unit vectors, respectively. Hence, we can rewrite each subvector in the initial solution, $(\alpha_i^1, \ldots, \alpha_i^H)$ as $\hat{\imath}^{k_i}$ and $(\beta_i^1, \ldots, \beta_i^W)$ as $\hat{\jmath}^{l_i}$. So the initial solution $(\alpha, \beta)_0$ is equal to $(\hat{\imath}^{k_1}, \ldots, \hat{\imath}^{k_m}, \hat{\jmath}^{l_1}, \ldots, \hat{\jmath}^{l_m})$ and the introduced points are as follows, Table 1.1 (note the vectors are written row by row).

The last $mW - \sum_{i\in I} w_i$ of points are linearly independent from the other points, since there is only a single $\hat{\jmath}^l$ with $l \in V\backslash\{l_i\}$ in each column. The second part of the points are linearly independent from the other points, since there is only a single $\hat{\imath}^k$

Table 1.1: $m(W+H)-\sum_{i\in I}[w_i+h_i]+1$ affinely independent points in conv(S): The table consist of four vertical parts: the first points; the points from consecutively shifting the values of α variables, (1.17), (1.18); the points from consecutively shifting the values of β variables. The first part has one point. The second and the third part has $mH-\sum_{i\in I}h_i$ points. The fourth part has $mW-\sum_{i\in I}w_i$. The points are written row by row.

$\begin{bmatrix}\alpha_1^1\\ \vdots\\ \alpha_1^H\end{bmatrix}$	$\begin{bmatrix}\alpha_2^1\\ \vdots\\ \alpha_2^H\end{bmatrix}$	$\cdots$	$\begin{bmatrix}\alpha_m^1\\ \vdots\\ \alpha_m^H\end{bmatrix}$	$\begin{bmatrix}\beta_1^1\\ \vdots\\ \beta_1^W\end{bmatrix}$	$\begin{bmatrix}\beta_2^1\\ \vdots\\ \beta_2^W\end{bmatrix}$	$\cdots$	$\begin{bmatrix}\beta_m^1\\ \vdots\\ \beta_m^W\end{bmatrix}$	
$\hat{i}^{k_1}$	$\hat{i}^{k_2}$	$\cdots$	$\hat{i}^{k_m}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	
$\hat{i}^{k}$	$\hat{i}^{k_2+k}$	$\cdots$	$\hat{i}^{k_m+k}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$k\in\{1,\dots,H_0\}\backslash\{k_1\}$
$\hat{i}^{k_1+\bar{k}}$	$\hat{i}^{k}$	$\cdots$	$\hat{i}^{k_m+\bar{k}}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$k\in\{1,\dots,H_0\}\backslash\{k_2\}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\hat{i}^{k_1+\bar{k}}$	$\hat{i}^{k_2+\bar{k}}$	$\cdots$	$\hat{i}^{k}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$k\in\{1,\dots,H_0\}\backslash\{k_m\}$
$\hat{i}^{k}$	$\hat{i}^{k_2}$	$\cdots$	$\hat{i}^{k_m}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$k\in\{H_0+1,\dots,H\}\backslash U_1$
$\hat{i}^{k_1}$	$\hat{i}^{k}$	$\cdots$	$\hat{i}^{k_m}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$k\in\{H_0+1,\dots,H\}\backslash\bar{U}_2$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\hat{i}^{k_1}$	$\hat{i}^{k_2}$	$\cdots$	$\hat{i}^{k}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$k\in\{H_0+1,\dots,H\}\backslash\bar{U}_m$
$\hat{i}^{k_1}$	$\hat{i}^{k_2+k}$	$\cdots$	$\hat{i}^{k_m+k}$	$\hat{j}^{l}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l_m}$	$l\in V\backslash(\bar{V}_1\cup\{l_1\})$
$\hat{i}^{k_1+\bar{k}}$	$\hat{i}^{k_2}$	$\cdots$	$\hat{i}^{k_m+\bar{k}}$	$\hat{j}^{l_1}$	$\hat{j}^{l}$	$\cdots$	$\hat{j}^{l_m+\bar{k}}$	$l\in V\backslash(\bar{V}_2\cup\{l_2\})$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\hat{i}^{k_1+\bar{k}}$	$\hat{i}^{k_2+\bar{k}}$	$\cdots$	$\hat{i}^{k_m}$	$\hat{j}^{l_1}$	$\hat{j}^{l_2}$	$\cdots$	$\hat{j}^{l}$	$l\in V\backslash(\bar{V}_m\cup\{l_m\})$

with $k\in\{1,\dots,H_0\}\backslash\{k_i\}$ in each column. For elements from the third part, in each column there is only a single $\hat{i}^k$ with $k\in\{H_0+1,\dots,H\}\backslash\bar{U}_i$ or it is equal at most to one element from the second part and/or at most to one element from the fourth part, which are linearly independent from all points. Thus, the points from the third part are linearly independent. Thus, the points are linearly and hence affinely independent.

Because constraints (1.2)-(1.6) hold for all points, we have $(\alpha_i^k,\dots,\beta_i^l,\dots)\in$ conv(S). Since the number of points in total is $m(W+H)-\sum_{i\in I}[w_i+h_i]+1$, the lemma follows. □

1.2.1 Non-overlapping inequalities

Constraint (1.6) handles the case of two items. Generally speaking the case of two items can be trivially extended on more items. Suppose we have three items $p,q,r\in I:\ p\neq q\neq r$, which fit into the strip in W-direction. Let us consider some allocations of these items against each other, see Fig. 1.2.

Obviously if projections of these three items overlap in one direction then they should not overlap in the other direction, otherwise there is an overlapping of items. Thus, for $u\in U$, $u_p\in H_p(u)$, $u_q\in H_q(u)$, $u_r\in H_r(u)$: $\alpha_p^{u_p}+\alpha_q^{u_q}+\alpha_r^{u_r}=3\ \Rightarrow\ \beta_p^{v_p}+$

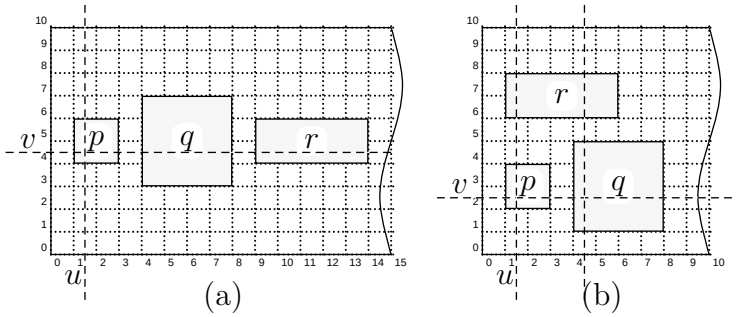


Figure 1.2: Allocations of three items p, q, r: (a) – Projections of $\{p, q, r\}$ overlap in $(0, W)$, but do not in H-direction; (b) – Projections of $\{p, q\}$ overlap in $(0, W)$, projections of $\{p, r\}$ and $\{q, r\}$ overlap in $(0, H)$.

$\beta_q^{v_q} + \beta_r^{v_r} \leq 1,\ \forall v \in V,\ v_p \in W_p(v),\ v_q \in W_q(v),\ v_r \in W_r(v)$. So, that follows

$$\alpha_p^{u_p} + \alpha_q^{u_q} + \alpha_r^{u_r} + \beta_p^{v_p} + \beta_q^{v_q} + \beta_r^{v_r} \leq 4, \quad u \in U,\ v \in V.$$

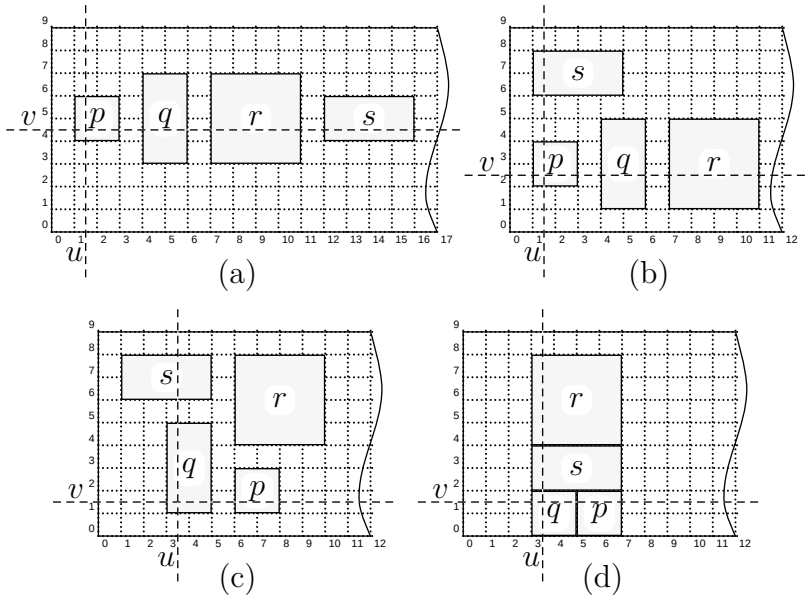


Figure 1.3: Allocations of four items p, q, r, s: (a) – Projections of $\{p, q, r, s\}$ overlap in $(0, W)$, but do not in $(0, H)$; (b) – Projections of $\{p, q, r\}$ overlap in $(0, W)$, projections of $\{p, s\}$ and $\{q, s\}$ overlap in $(0, H)$; (c) – Projections of $\{s, r\}$, $\{q, r\}$, $\{p, q\}$ overlap in $(0, W)$, projections of $\{q, s\}$ and $\{p, r\}$ overlap in $(0, H)$; (d) – Projections of $\{p, q\}$ overlap in $(0, W)$, projections of $\{r, s, p\}$ and $\{r, s, q\}$ overlap in $(0, H)$.

Analogously for 4 items, see Fig. 1.3. So,

$$\alpha_p^{u_p} + \alpha_q^{u_q} + \alpha_r^{u_r} + \alpha_s^{u_s} + \beta_p^{v_p} + \beta_q^{v_q} + \beta_r^{v_r} + \beta_s^{v_s} \leq 5, \quad u \in U,\ v \in V,$$

where $u_s \in H_s(u)$ and $v_s \in W_s(v)$.

The above idea can be extended for an arbitrary number of items. Let μ^W and μ^H be the maximal numbers of items, which fit in the W-, H-directions, respectively. It

means that for $\mu = \min\{\mu^W, \mu^H\}$ overlapping items' projections we should apply the overlapping constraints type (1.6). Let

$$C := \{S \subseteq I : \sum_{i \in S} w_i \leq W, \sum_{i \in S} h_i \leq H\}$$

be the set of subsets of I whose items fit into W- and H-directions. If for an $u \in U$ and $M \in C$

$$\sum_{i \in M} \alpha_i^u = |M| \;\Rightarrow\; \sum_{i \in M} \beta_i^v \leq 1, \quad v \in V,$$

which leads to

$$\sum_{i \in M} [\alpha_i^u + \beta_i^v] \leq |M| + 1, \quad u \in U,\ v \in V,\ M \in C. \tag{1.20}$$

In the same manner we get the similar inequality for $\tilde{\alpha}$ and $\tilde{\beta}$:

$$\sum_{i \in M} [\tilde{\alpha}_i^u + \tilde{\beta}_i^v] \leq |M| + 1, \quad u \in U,\ v \in V,\ M \in C. \tag{1.21}$$

Remark 1.1. *The number of* (1.20) *and* (1.21) *constraints is* $O(WH(2^m - m))$.

Theorem 1.3. *Inequalities* (1.20) *and* (1.21) *are valid for* S.

Proof. We prove the theorem using a contradiction. Let for an $u \in U$, $v \in V$, $M \in C$ the inequality (1.20) does not hold, i.e.:

$$\sum_{i \in M} [\alpha_i^u + \beta_i^v] \geq |M| + 2.$$

It means that there exist at most $2|M| - (|M| + 2)$ elements α^u and β^v that are zero. Therefore, there are at most $|M| - 2$ pairs of α_k^u and β_k^v, $k \in M$ where $\alpha_k^u = 0$ or $\beta_k^v = 0$. Thus, there exist $|M| - (|M| - 2) = 2$ pairs where both α^u and β^v are non-zero. Let $p, q \in M$, so that $p \neq q$ and $\alpha_p^u = \beta_p^v = \alpha_q^u = \beta_q^v = 1$. But this contradicts (1.6) by $\alpha_p^u + \beta_p^v + \alpha_q^u + \beta_q^v = 4$. Therefore, the assumption is incorrect.

Inequality (1.21) is proven in the similar manner. □

Lemma 1.4. *Let* $\bar{H}_i(u) := H_i(u) \setminus \{u\}$ *and* $\bar{W}_i(v) := W_i(v) \setminus \{v\}$. *Inequality* (1.21) *dominates* (1.20)*, if*

$$\sum_{i \in M} [\sum_{k \in \bar{H}_i(u)} \alpha_i^k + \sum_{k \in \bar{W}_i(v)} \beta_i^k] > 0.$$

Proof. Let us reformulate the inequality (1.21) as:

$$\sum_{i \in M} [\alpha_i^u + \beta_i^v] \leq |M| + 1 - \sum_{i \in M} [\sum_{k \in \bar{H}_i(u)} \alpha_i^k + \sum_{k \in \bar{W}_i(v)} \beta_i^k].$$

If $\Gamma := \sum_{i \in M} [\sum_{k \in \bar{H}_i(u)} \alpha_i^k + \sum_{k \in \bar{W}_i(v)} \beta_i^k] = 0$ then inequalities (1.21) and (1.20) are equal. If $\Gamma > 0$ then we have $|M| + 1 - \Gamma \leq u(|M| + 1)$. Let $s := 1 - \frac{\Gamma}{|M|+1}$. According to the definition of a dominating inequality, $(1, \ldots, 1)^T \geq s(1, \ldots, 1)^T$ should hold. Hence, $\Gamma > 0$ follows. □

Remark 1.2. *Let us look closer at the* (1.20) *constraints and discuss their facet-defining property. Let* $m = 2$, $W = H := 20$, $w_i = h_i := 5$, $u = v := 10$, *and let* (1.20) *be satisfied at equality, i.e.,* $\alpha_1^{10} + \beta_1^{10} + \alpha_2^{10} + \beta_2^{10} = 3$. *But this means that items 1 and 2 will never be allocated at positions 11,...,14 in both* $(0, H)$ *and* $(0, W)$. *The latter means equations in the formulation* $\alpha_1^u = 0$, $\beta_1^v = 0$ *for* $u, v = 11, \ldots, 14$ *which reduces the dimension of the face* $\{(\alpha, \beta) \in \text{conv}(S) : \ \alpha_1^{10} + \beta_1^{10} + \alpha_2^{10} + \beta_2^{10} = 3\}$. *Hence, inequality* (1.20) *is not a facet-defining unless* $w_i = h_i := 1$, $i \in I$.

Since (1.21) dominates (1.20) and the latter is not a facet defining inequality in the general case, further we consider only (1.21) and prove that (1.21) is a facet-defining inequality for $\text{conv}(S)$.

Theorem 1.5. *Let* $u \in U$, $v \in V$, $M \subseteq I$, $H := u + 2(H_0 + \sum_{i \in M} h_i) + 3h_{\max} - 3$, *where* H_0 *is the value of an optimal solution of a problem with items* $I \backslash M$ *and:*

1. $\exists p, q \in M, \ p \neq q$:

$$h_p \leq u, \ h_q \leq u, \ w_p \leq v, \ w_q \leq v, \ v + w_p - 1 \leq W, \ v + w_q - 1 \leq W. \quad (1.22)$$

2. $\forall i \in M \setminus \{p, q\}$:

$$v - 1 \geq w_i, \ W - v \geq w_i. \quad (1.23)$$

3. $\forall i \in I \setminus M$:

$$w_i \leq v - 1 \ \text{or} \ w_i \leq W - v. \quad (1.24)$$

Inequality (1.21) *is facet-defining for* $\text{conv}(S)$, *i.e.,* $F := \{(\alpha, \beta) \in \text{conv}(S) : \ \sum_{i \in M} [\tilde{\alpha}_i^u + \tilde{\beta}_i^v] = |M| + 1\}$ *is a facet of* $\text{conv}(S)$.

Proof. The proof is done by constructing of $m(W + H) - \sum_{i \in I} [w_i + h_i]$ affinely independent points of $\text{conv}(S)$ which fulfill the condition $\sum_{i \in M} [\tilde{\alpha}_i^u + \tilde{\beta}_i^v] = |M| + 1$.

Without loss of generality we assume $p := 1$, $q := 2$, and $M := \{1, 2, \ldots, |M|\}$. Let us split H at three points:

$$\bar{k} := u + \sum_{i \in M} h_i + h_{\max} - 1;$$
$$\tilde{k} := \bar{k} + \sum_{i \in M} h_i + h_{\max} - 1;$$
$$\hat{k} := \tilde{k} + H_0 + h_{\max} - 1.$$

Note that term $h_{\max}$ in the expressions for $\bar{k}$, $\tilde{k}$, and $\hat{k}$ is needed for the feasibility of step 11 (see below). Let $H_1 := \sum_{i \in M} h_i$.

Further we construct points from a feasible solution which fulfills $\sum_{i \in M} [\tilde{\alpha}_i^u + \tilde{\beta}_i^v] = |M| + 1$. This feasible solution we construct as follows, see Fig. 1.4. Firstly, we allocate item 1 at position $k_1 := u - h_1 + 1$, $l_1 = v - w_1 + 1$. The other items $i \in M \setminus \{1\}$ we allocate at position v in $(0, W)$ and one after another in $(0, H)$: $k_i = u + h_1 + \sum_{2 \leq j < i} h_j + 1$, $l_i = v$. Items from $I \backslash M$ we allocate optimally after $\tilde{k}$ at positions

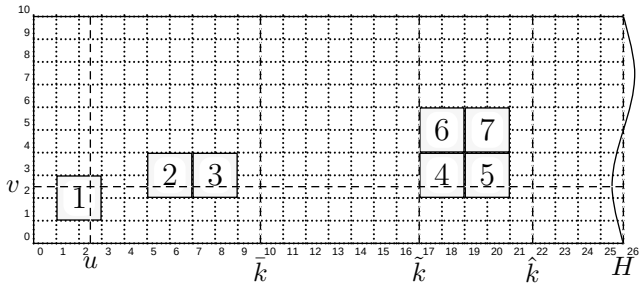


Figure 1.4: Feasible initial solution for the construction of linear independent points in $\{(\alpha,\beta)\in \mathrm{conv}(S):\ \sum_{i\in M}[\tilde{\alpha}_i^u+\tilde{\beta}_i^v]=|M|+1\}$. $M=\{1,2,3\}$, $I=M\cup\{4,\dots,7\}$, $u=v=3$.

$\tilde{k}+k_i$ and l_i where k_i and l_i are their positions in the optimal solution. Let $(\alpha,\beta)_0:=(\alpha_1^1,\dots,\alpha_1^H,\dots,\alpha_m^1,\dots,\alpha_m^H,\beta_1^1,\dots,\beta_1^W,\dots,\beta_m^1,\dots,\beta_m^W)$ be this feasible solution with $f:=H$ and $\alpha_i^{k_i}:=1$, $\beta_i^{l_i}:=1$, and $\alpha_i^k:=0$ for $k\in U\backslash\{k_i\}$, $\beta_i^l:=0$ for $l\in V\backslash\{l_i\}$.

Now we construct $m(W+H)-\sum_{i\in I}[w_i+h_i]$ points by shifting of items over the $(0,H)$- and $(0,W)$-axis, respectively, always avoiding their original location. Every shifting should be processed while $\sum_{i\in M}[\tilde{\alpha}_i^u+\tilde{\beta}_i^v]=|M|+1$ holds. Shiftings over the $(0,H)$-axis are divided in five phases as H is divided in five parts by u, $\bar{k}$, $\tilde{k}$, $\hat{k}$. Shiftings over the $(0,W)$-axis are divided into two phases, since W is divided in two parts by v. Here are the following fourteen steps:

1. Shift the items $i\in M\setminus\{1\}$ over the $(0,H)$-axis after the coordinate u skipping the origin for every i, see Fig. 1.5.

   Let $\alpha_1^{u-h_1+1}:=1$, $\beta_1^{v-w_1+1}:=1$, $f:=H$, and:

$$\beta_i^l:=\begin{cases}1,\ i\in M\setminus\{1\},\ l=v;\\ 1,\ i\in I\setminus M,\ l=l_i;\\ 0,\ \text{otherwise.}\end{cases}$$

   This shifting is divided into four phases, see Fig. 1.5a-1.5d:

   (a) For each $s\in M\setminus\{1\}$:

$$\alpha_i^k:=\begin{cases}1,\ i=s,\ k\in\{u+1,\dots,\bar{k}-h_i+1\}\setminus\{u+k_i\};\\ 1,\ i\in M\backslash\{1,s\},\ k=\bar{k}+k_i;\\ 1,\ i\in I\backslash M,\ k=\tilde{k}+k_i;\\ 0,\ \text{otherwise;}\end{cases}\tag{1.25}$$

   (b) For each $s\in M\setminus\{1\}$:

$$\alpha_i^k:=\begin{cases}1,\ i=s,\ k\in\{\bar{k}-h_i+2,\dots,\tilde{k}-h_i+1\};\\ 1,\ i\in M\backslash\{1,s\},\ k=u+k_i;\\ 1,\ i\in I\backslash M,\ k=\tilde{k}+k_i;\\ 0,\ \text{otherwise;}\end{cases}\tag{1.26}$$

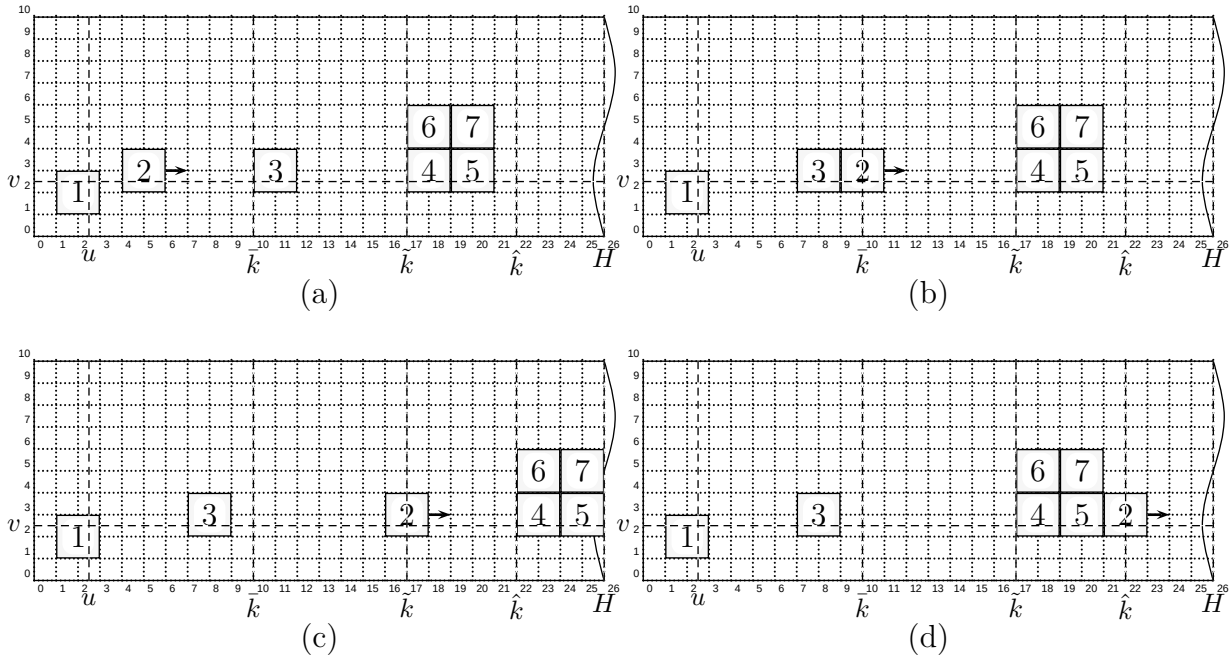


Figure 1.5: Feasible shifting of items from $M \setminus \{1\}$ over the $(0, H)$-axis while $\sum_{i \in M}[\tilde{\alpha}_i^u + \tilde{\beta}_i^v] = |M| + 1$ holds. $M = \{1, 2, 3\}$, $I = M \cup \{4, \ldots, 7\}$, $u = v = 3$. The shifting of item 2 after u is divided in four parts: (a) – First part, item 3 is allocated after $\bar{k}$; (b) – Second part, item 3 is allocated at its origin; (c) – Third part, items 4,. . . ,7 are allocated after $\hat{k}$; (d) – Fourth part, items 4,. . . ,7 are allocated at their origin.

(c) For each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k \in \{\tilde{k} - h_i + 2, \ldots, \hat{k} - h_i + 1\}; \\ 1, \ i \in M \backslash \{1, s\}, \ k = u + k_i; \\ 1, \ i \in I \backslash M, \ k = \hat{k} + k_i; \\ 0, \ \text{otherwise}; \end{cases} \tag{1.27}$$

(d) For each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k \in \{\hat{k} - h_i + 2, \ldots, H\} \setminus \bar{U}_i; \\ 1, \ i \in M \backslash \{1, s\}, \ k = u + k_i; \\ 1, \ i \in I \backslash M, \ k = \tilde{k} + k_i; \\ 0, \ \text{otherwise}; \end{cases} \tag{1.28}$$

The number of points which arise through the shiftings (1.25)-(1.28) is $\sum_{i \in M \setminus \{1\}}[H - u - 1 - |\bar{U}_i|] = \sum_{i \in M \setminus \{1\}}[H - u - 1 - (h_i - 1)] = (|M| - 1)(H - u) - \sum_{i \in M \setminus \{1\}} h_i$.

2. Shift the items $i \in M \setminus \{1\}$ over the $(0, H)$-axis before the coordinate u, see Fig. 1.6.

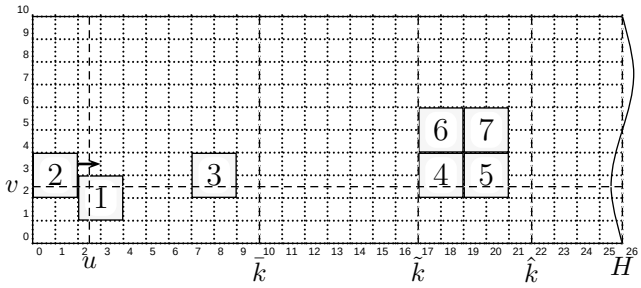


Figure 1.6: Feasible shifting of items from $M \setminus \{1\}$ over the $(0, H)$-axis before u while $\sum_{i \in M}[\tilde{\alpha}_i^u + \tilde{\beta}_i^v] = |M| + 1$ holds. $M = \{1, 2, 3\}$, $I = M \cup \{4, \ldots, 7\}$, $u = v = 3$.

Note that here we change the position of item 1 in $(0, H)$. Let $\alpha_1^u := 1$, $\beta_1^{v-w_1+1} := 1$, $f := H$, and for each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k \in \{1, \ldots, u - h_i\}; \\ 1, \ i \in M \setminus \{1, s\}, \ k = u + k_i; \\ 1, \ i \in I \setminus M, \ k = \tilde{k} + k_i; \\ 0, \ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1, \ i \in M \setminus \{1\}, \ l = v; \\ 1, \ i \in I \setminus M, \ l = l_i; \\ 0, \ \text{otherwise}. \end{cases} \tag{1.29}$$

The number of points which arise through the shifting (1.29) is $\sum_{i \in M \setminus \{1\}}[u - h_i] = (|M| - 1)u - \sum_{i \in M \setminus \{1\}} h_i$.

3. Shift the items $i \in M \setminus \{1\}$ over the $(0, W)$-axis after the coordinate v, see Fig. 1.7a.

   Let $\alpha_1^{u-h_1+1} := 1$, $\beta_1^{v-w_1+1} := 1$, $f := H$, and for each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k = u; \\ 1, \ i \in M \setminus \{1, s\}, \ k = \bar{k} + k_i; \\ 1, \ i \in I \setminus M, \ k = \tilde{k} + k_i; \\ 0, \ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1, \ i = s, \ l \in \{v + 1, \ldots, W - w_i + 1\} \\ 1, \ i \in M \setminus \{1\}, \ l = v; \\ 1, \ i \in I \setminus M, \ l = l_i; \\ 0, \ \text{otherwise}. \end{cases} \tag{1.30}$$

   The number of points which arise through the shifting (1.30) is $\sum_{i \in M \setminus \{1\}}[W - v - |\bar{V}_i|] = \sum_{i \in M \setminus \{1\}}[W - v - (w_i - 1)] = (|M| - 1)(W - v + 1) - \sum_{i \in M \setminus \{1\}} w_i$.

4. Shift the items $i \in M \setminus \{1\}$ over the $(0, W)$-axis before the coordinate v, see Fig. 1.7b.

   Note that here we change the position of item 1 in $(0, W)$. Let $\alpha_1^{u-h_1+1} := 1$, $\beta_1^v := 1$, $f := H$, and for each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1, \ i = s, \ k = u; \\ 1, \ i \in M \setminus \{1, s\}, \ k = \bar{k} + k_i; \\ 1, \ i \in I \setminus M, \ k = \tilde{k} + k_i; \\ 0, \ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1, \ i = s, \ l \in \{1, \ldots, v - w_i\} \\ 1, \ i \in M \setminus \{1\}, \ l = v; \\ 1, \ i \in I \setminus M, \ l = l_i; \\ 0, \ \text{otherwise}. \end{cases}$$

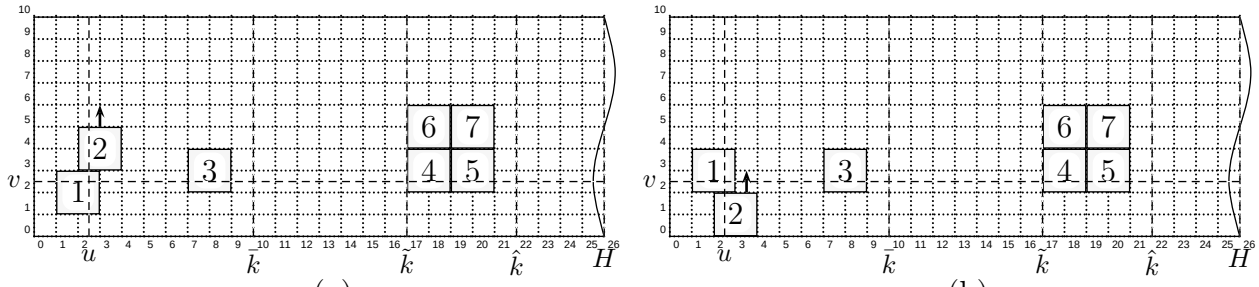


Figure 1.7: Feasible shifting of items from $M \setminus \{1\}$ over the $(0, W)$-axis while $\sum_{i \in M}[\tilde{\alpha}_i^u + \tilde{\beta}_i^v] = |M| + 1$ holds. $M = \{1, 2, 3\}$. $I = M \cup \{4, 5\}$. $u = v = 3$. Description: (a) – Shifting of item 2 after v; (b) – Shifting of item 2 before v.

(1.31)

The number of points which arise through the shifting (1.31) is $\sum_{i \in M \setminus \{1\}}[v - w_i] = (|M| - 1)v - \sum_{i \in M \setminus \{1\}} w_i$.

5. Shift the item 1 over the $(0, H)$-axis after the coordinate u.

   In this case item 2 covers position u and v. Let $\alpha_2^{u-h_2+1} := 1$, $\beta_2^{v-w_2+1} := 1$, $f := H$, and:

$$\beta_i^l := \begin{cases} 1, \ i \in M \setminus \{2\}, \ l = v; \\ 1, \ i \in I \setminus M, \ l = l_i; \\ 0, \ \text{otherwise.} \end{cases}$$

   This shifting is divided into four phases:

   (a)

$$\alpha_i^k := \begin{cases} 1, \ i = 1, \ k \in \{u + 1, \ldots, \bar{k} - h_1 + 1\}; \\ 1, \ i \in M \setminus \{1, 2\}, \ k = \bar{k} + k_i; \\ 1, \ i \in I \setminus M, \ k = \tilde{k} + k_i; \\ 0, \ \text{otherwise;} \end{cases} \tag{1.32}$$

   (b)

$$\alpha_i^k := \begin{cases} 1, \ i = 1, \ k \in \{\bar{k} - h_1 + 2, \ldots, \tilde{k} - h_1 + 1\}; \\ 1, \ i \in M \setminus \{1, 2\}, \ k = u + k_i; \\ 1, \ i \in I \setminus M, \ k = \tilde{k} + k_i; \\ 0, \ \text{otherwise;} \end{cases} \tag{1.33}$$

(c)

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k \in \{\tilde{k} - h_1 + 2, \ldots, \hat{k} - h_1 + 1\}; \\ 1,\ i \in M \backslash \{1,2\},\ k = u + k_i; \\ 1,\ i \in I \backslash M,\ k = \hat{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \tag{1.34}$$

(d)

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k \in \{\hat{k} - h_1 + 2, \ldots, H\} \setminus \bar{U}_1; \\ 1,\ i \in M \backslash \{1,2\},\ k = u + k_i; \\ 1,\ i \in I \backslash M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \tag{1.35}$$

The number of points which arise through the shifting (1.32)-(1.35) is $H - u - |\bar{U}_1| = H - u - (h_1 - 1) = H - u + 1 - h_1$.

6. Shift the item 1 over the $(0, H)$-axis before the coordinate u:
   Let $\alpha_2^u := 1$, $\beta_2^{v-w_2+1} := 1$, $f := H$, and:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k \in \{1, \ldots, u - h_1\}; \\ 1,\ i \in M \backslash \{1,2\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \backslash M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i \in M \setminus \{2\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.36}$$

   The number of points which arise through the shifting (1.36) is $u - h_1$.

7. Shift the item 1 over the $(0, W)$-axis after the coordinate v.
   Let $\alpha_2^{u-h_2+1} := 1$, $\beta_2^{v-w_2+1} := 1$, $f := H$, and:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k = u; \\ 1,\ i \in M \backslash \{1,2\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \backslash M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = 1,\ l \in \{v + 1, \ldots, W - w_i + 1\} \\ 1,\ i \in M \setminus \{1,2\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.37}$$

   The number of points which arise through the shifting (1.37) is $W - v - |\bar{V}_1| = W - v - (w_1 - 1) = W - v + 1 - w_1$.

8. Shift the item 1 over the $(0, W)$-axis before the coordinate v.
   Let $\alpha_2^{u-h_2+1} := 1$, $\beta_2^v := 1$, $f := H$, and:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k = u; \\ 1,\ i \in M \backslash \{1,2\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \backslash M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = 1,\ l \in \{1, \ldots, v - w_1\} \\ 1,\ i \in M \setminus \{1,2\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases}$$

(1.38)

The number of points which arise through the shifting (1.38) is $v - w_1$.

9. Shift item 1 over the $(0, H)$-axis within coordinates $H_1(u)$ skipping its origin in the initial solution $(\alpha, \beta)_0$.

   Let $f := H$, and:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k \in H_1(u) \setminus \{u - h_1 + 1\}; \\ 1,\ i \in M \setminus \{1\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \setminus M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = 1,\ l = v - w_1 + 1; \\ 1,\ i \in M \setminus \{1\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.39}$$

   The number of points which arise through the shifting (1.29) is $|H_1(u)| - 1 = h_1 - 1$.

10. Shift item 1 over the $(0, W)$-axis within coordinates $W_1(v)$ skipping its origin in the initial solution $(\alpha, \beta)_0$.

    Let $f := H$, and:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k = u - h_1 + 1; \\ 1,\ i \in M \setminus \{1\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \setminus M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = 1,\ l \in W_1(v) \setminus \{v - w_1 + 1\}; \\ 1,\ i \in M \setminus \{1\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.40}$$

    The number of points which arise through the shifting (1.40) is $|W_1(v)| - 1 = w_1 - 1$.

11. Shift every item $i \in M \setminus \{1\}$ over the $(0, H)$-axis within coordinates $H_i(u)$. This differs from the previous step, since items $M \setminus \{1\}$ have never been allocated at positions $H_i(u)$.

    Let $f := H$ then for each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k = \bar{k} + 1; \\ 1,\ i = s,\ k \in H_i(u); \\ 1,\ i \in M \setminus \{1, s\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \setminus M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = 1,\ l = v - w_1 + 1; \\ 1,\ i \in M \setminus \{1\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.41}$$

    The number of points which arise through the shifting (1.41) is $\sum_{i \in M \setminus \{1\}} |H_i(u)| = \sum_{i \in M \setminus \{1\}} h_i$.

12. Shift every item $i \in M \setminus \{1\}$ over the $(0, W)$-axis within coordinates $W_i(v)$ skipping its origin v. This differs from the step prior to the previous, since items $M \setminus \{1\}$ have already been allocated at position v in the initial solution $(\alpha, \beta)_0$.

    Let $f := H$ then for each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1,\ i = 1,\ k = \bar{k} + 1; \\ 1,\ i = s,\ k = u - h_i + 1; \\ 1,\ i \in M \setminus \{1, s\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \setminus M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = 1,\ l = v; \\ 1,\ i = s,\ l \in W_i(v) \setminus \{v\}; \\ 1,\ i \in M \setminus \{1, s\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.42}$$

    The number of points which arise through the shifting (1.42) is $\sum_{i \in M \setminus \{1\}} |W_i(v)| - 1 = -(|M| - 1) + \sum_{i \in M \setminus \{1\}} w_i$.

13. Shift items from $I \setminus M$ over the $(0, H)$-axis skipping their origin in the initial solution $(\alpha, \beta)_0$.

    Depending on which part of the condition (1.23) is fulfilled, we let $\beta_i^{v - w_1 + 1} := 1$ or $\beta_i^v := 1$ for $i \in I \setminus M$. Suppose $w_i \leq v - 1$ for $i \in I \setminus M$, so $\beta_i^v := 1$ for $i \in M$. Let $\alpha_1^{u - h_1 + 1} := 1$, and $\beta_1^v := 1$ and $\alpha_1^k = \beta_1^l := 0$, otherwise; $f := H$:

$$\beta_i^l := \begin{cases} 1,\ i \in M \setminus \{1\},\ l = v; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 0,\ \text{otherwise}; \end{cases} \tag{1.43}$$

    We divide the shifting into five phases as in steps 1-2:

    (a) For each $s \in I \setminus M$:

$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k \in \{1, \ldots, u - h_i + 1\}; \\ 1,\ i \in M \setminus \{1\},\ k = u + k_i; \\ 1,\ i \in I \setminus M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}; \end{cases} \tag{1.44}$$

    This shifting is valid since we require (1.22)-(1.23).

    (b) For each $s \in I \setminus M$:

$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k \in \{u - h_i + 2, \ldots, \bar{k} - h_i + 1\}; \\ 1,\ i \in M \setminus \{1\},\ k = \bar{k} + k_i; \\ 1,\ i \in I \setminus M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise}. \end{cases} \tag{1.45}$$

(c) For each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k \in \{\bar{k} - h_i + 1, \ldots, \tilde{k} - h_i + 1\}; \\ 1,\ i \in M \backslash \{1, s\},\ k = u + k_i; \\ 1,\ i \in I \backslash M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise.} \end{cases} \tag{1.46}$$

(d) For each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k \in \{\tilde{k} - h_i + 2, \ldots, \hat{k} - h_i + 1\}; \\ 1,\ i \in M \backslash \{1, s\},\ k = u + k_i; \\ 1,\ i \in I \backslash M,\ k = \hat{k} + k_i; \\ 0,\ \text{otherwise.} \end{cases} \tag{1.47}$$

(e) For each $s \in M \setminus \{1\}$:

$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k \in \{\hat{k} - h_i + 2, \ldots, H\} \setminus \bar{U}_i; \\ 1,\ i \in M \backslash \{1, s\},\ k = u + k_i; \\ 1,\ i \in I \backslash M,\ k = \tilde{k} + k_i; \\ 0,\ \text{otherwise.} \end{cases} \tag{1.48}$$

The number of points which arise through the shiftings (1.44)-(1.48) is $\sum_{i \in I \setminus M}[H - 1 - |\bar{U}_i|] = \sum_{i \in I \setminus M}[H - 1 - (h_i - 1)] = |I \setminus M|H - \sum_{i \in I \setminus M} h_i$.

14. Shift items from $I \setminus M$ over the $(0, W)$-axis skipping their origin in the initial solution $(\alpha, \beta)_0$.

Let $\alpha_1^{u - h_1 + 1} := 1$, $\beta_1^{v - w_1 + 1} := 1$, $f := H$, and for each $s \in I \setminus M$:

$$\alpha_i^k := \begin{cases} 1,\ i = s,\ k = \tilde{k} + k_i; \\ 1,\ i \in (I \backslash M) \setminus \{s\},\ k = \hat{k} + k_i; \\ 1,\ i \in M,\ k = u + k_i; \\ 0,\ \text{otherwise;} \end{cases} \qquad \beta_i^l := \begin{cases} 1,\ i = s,\ l \in (\{1, \ldots, W\} \setminus \bar{V}_i(v)) \setminus \{v\}; \\ 1,\ i \in I \setminus M,\ l = l_i; \\ 1,\ i \in M,\ l = v; 0,\ \text{otherwise.} \end{cases} \tag{1.49}$$

The number of points which arise through the shifting (1.49) is $\sum_{i \in I \setminus M}[W - 1 - \bar{V}_i(v)] = \sum_{i \in I \setminus M}[W - 1 - (w_i - 1)] = |I \setminus M|W - \sum_{i \in I \setminus M} w_i$.

Now we calculate the number of points from steps 1-12:

$$\{(|M|-1)(H-u)-\sum_{i\in|M|\setminus\{1\}}h_i\}+\{(|M|-1)u-\sum_{i\in M\setminus\{1\}}h_i\}+\{(|M|-1)(W-$$
$$v+1)-\sum_{i\in M\setminus\{1\}}w_i\}+\{(m-1)v-\sum_{i\in M\setminus\{1\}}w_i\}+\{H-u+1-h_1\}+\{u-h_1\}+$$
$$\{W-v+1-w_1\}+\{v-w_1\}+\{h_1-1\}+\{w_1-1\}+\sum_{i\in M\setminus\{1\}}h_i-\{(|M|-1)-$$
$$\sum_{i\in M\setminus\{1\}}w_i\}=$$
$$|M|(H+W)+|M|-1-2\sum_{i\in I\setminus\{1\}}[w_i+h_i]-w_1-h_1-|M|+1+\sum_{i\in M\setminus\{1\}}[w_i+h_i]=$$
$$|M|(W+H)-\sum_{i\in M}[w_i+h_i].$$

Herewith, that gives together with steps 13-14 the number of points:

$$|M|(W+H)-\sum_{i\in M}[w_i+h_i]+|I\setminus M|(W+H)-\sum_{i\in I\setminus M}[w_i+h_i]=m(W+H)-\sum_{i\in I}[w_i+h_i].$$

The points are affinely independent for the same reason as the points in the proof of Lemma 1.2.

Because constraints (1.2)-(1.6) hold for all points, we have $(\alpha_i^k,\ldots,\beta_i^l,\ldots)\in\mathrm{conv}(S)$. Since the number of points in total is $m(W+H)-\sum_{i\in I}[w_i+h_i]$, the theorem follows. □

1.2.2 Cover inequalities

This section is partially based on [NW88]. In the following we consider cover inequalities and a *lifting procedure* for their strengthening. We also prove that the lifted cover inequalities are facet-defining for $\mathrm{conv}(S)$.

Definition 1.1. *A set $C\subseteq I$ for $u\in U$ is a cover if $\sum_{i\in C}w_i-W>0$. A cover is minimal if $C\setminus\{i\}$ is not a cover for any $i\in C$.*

Lemma 1.6. *If $C\subseteq I$ is a cover, the cover inequality*

$$\sum_{i\in C}\tilde{\alpha}_i^u\leq|C|-1 \tag{1.50}$$

is valid for S for any $u\in U$.

Proof. The proof is similar to Proposition 2.1 in [NW88], p.265. □

By Proposition 2.3 in [NW88], p.266, the cases are shown where the cover inequalities are facet-defining for the polyhedron of the 0-1 knapsack problem. For the general case, a *lifting procedure* is considered [NW88], p.269, and [Wol98], p.149, which tightens the cover inequalities so they become facet-defining for general case.

Strengthening cover inequalities

The idea of strengthening the cover inequalities is to add to the left-hand side of (1.50) as much as possible items from $I\backslash C$ which are not in the cover with the proper coefficient π_i, $i \in I\backslash C$, while the right-hand side of the inequality remains the same, i.e.:

$$\sum_{i\in C} \tilde{\alpha}_i^u + \sum_{i\in I\backslash C} \pi_i \tilde{\alpha}_i^u \leq |C| - 1 \tag{1.51}$$

is valid for S.

Now we describe the lifting procedure which returns in case of a minimal cover C a set of π_i values.

Algorithm 1.1 (LIFTING). *Determination of a lifted cover inequality for a minimal cover C.*
Input data: Minimal cover C.
Output data: (1.51).

(1) *Set $k = 0$. Let $R = I\backslash C =: \{r_1, \ldots, r_{|R|}\}$.*
(2) *Set $k = k+1$. Calculate the largest value for π_{r_k} for which the following inequality is valid:*

$$\pi_{r_k}\tilde{\alpha}_{r_k}^u + \sum_{i=1}^{k-1} \pi_{r_i}\tilde{\alpha}_{r_i}^u + \sum_{i\in C} \tilde{\alpha}_i^u \leq |C| - 1,$$

through the solution of the following 0-1 integer program:

$$\lambda_k = \max \sum_{i=1}^{k-1} \pi_{r_i}\tilde{\alpha}_{r_i}^u + \sum_{i\in C} \tilde{\alpha}_i^u; \quad \text{s.t.} \tag{1.52}$$

$$\sum_{i=1}^{k-1} \pi_{r_i}\tilde{\alpha}_{r_i}^u + \sum_{i\in C} \tilde{\alpha}_i^u \leq W - w_{r_k}; \tag{1.53}$$

$$\tilde{\alpha}_i^u = 0, \quad i = r_k, \ldots, r_{|R|}; \tag{1.54}$$

$$\tilde{\alpha}_i^u \in \{0, 1\}, \quad i \in I,\ u \in U. \tag{1.55}$$

(3) *Set $\pi_{r_k} = |C| - 1 - \lambda_k$.*
(4) *If $k = |R|$ then exit, else go to 1.*

1.2.3 Density inequalities

Here we consider only solutions which are in some sense dense. That means that for a solution which is feasible there exists no item which does not contact from the left-hand side neither another item nor $(0, W)$ side of the strip.

Let $I_u = \{i \in I :\ u - h_i \geq 1\}$ for $u \in U$ denote the set of items which can end right before position u and μ^W be the maximal number of items which fit in W. The

following inequality prevents allocation of items at a coordinate u if there does not exist at least one item i which allocation was at the coordinate $u - h_i$, if $m > \mu^W$:

$$\sum_{i \in I} \alpha_i^u \leq \mu^W \sum_{i \in I_u} \alpha_i^{u-h_i}, \quad u \in \{2, \ldots, H - \min_{i \in I}\{h_i\} + 1\}. \tag{1.56}$$

The same holds for the other direction. Let denote $I_v = \{i \in I : \ v - w_i \geq 1\}$, $v \in V$, the set of items which can end right before position v and μ^H be the maximal number of items which fit in H, if $m > \mu^H$:

$$\sum_{i \in I} \beta_i^v \leq \mu^H \sum_{i \in I_v} \beta_i^{v-w_i}, \quad v \in \{2, \ldots, W - \min_{i \in I}\{w_i\} + 1\}. \tag{1.57}$$

Remark 1.3. *Note if $m = \mu^W$ then inequalities* (1.56) *are incorrect, since from $\sum_{i \in I} \alpha_i^u = m$ should follow $\sum_{i \in I} \alpha_i^{u-h_i} = 0$ but it follows $\sum_{i \in I} \alpha_i^{u-h_i} \geq 1$.*

Remark 1.4. *Note the factors μ^W and μ^H before the second sum in* (1.56) *and* (1.57)*, respectively, are not reducible in the general case.*

Theorem 1.7. *Let $m \geq \mu^W + 2$. For an $u \in U$, if $\exists(\alpha, \beta) \in S$:*

$$\sum_{i \in K(j)} \alpha_i^u = \mu^W, \quad j \in I^u, \ K(j) \subseteq I \setminus \{j\}, \tag{1.58}$$

then (1.56) *for an $u \in \{2, \ldots, H - h_{\min} + 1\}$ is a facet-defining inequality for* $\mathrm{conv}(S)$*, i.e.,*

$$F := \{(\alpha, \beta) \in \mathrm{conv}(S) : \ \sum_{i \in I} \alpha_i^u = \mu^W \sum_{i \in I_u} \alpha_i^{u-h_i}\}$$

is a facet of $\mathrm{conv}(S)$*.*

Proof. Let us consider the following equation:

$$\sum_{i \in I} \alpha_i^u = \mu^W \sum_{i \in I_u} \alpha_i^{u-h_i}. \tag{1.59}$$

Since the left-hand side of the equation $\sum_{i \in I} \alpha_i^u \leq \mu^W$ due to the definition then there exist only two types of solutions, namely:

$$\sum_{i \in I} \alpha_i^u = 0, \quad \sum_{i \in I^u} \alpha_i^{u-h_i} = 0; \tag{1.60}$$

$$\sum_{i \in I} \alpha_i^u = \mu^W, \quad \sum_{i \in I^u} \alpha_i^{u-h_i} = 1. \tag{1.61}$$

The case (1.60) leads to the considerable reduction of the dimension. The case of (1.61) means that there exists a subset of μ^W items which are allocated at the coordinate u and at least one item i which is allocated at the coordinate $u - h_i$. So the number of the fixed items is $\mu^W + 1$ and the other items are free to fix. According to the condition of the theorem we can go through all combinations of items so that every item becomes free. Hence, analogues to the proof of Lemma 1.2 and Theorem 1.5 we construct $m(W + H) - \sum_{i \in I}[w_i + h_i]$ points under the condition (1.59).

□

Remark 1.5. *If* $m := \mu^W + 1$ *then obviously* (1.56) *is not a facet-defining inequality, unless* (1.58) *holds and* $h_i = 1$, $H = 2$.

Remark 1.6. *Note the inequality* (1.57) *is not facet-defining, since while minimization of the* (1.1) *the maximal value of the sum* $\sum_{i\in I}\beta_i^v$ *will be smaller then* μ^H, *and the equation* $\sum_{i\in I}\beta_i^v = \mu^H\sum_{i\in I_v}\beta_i^{v-w_i}$ *will have only one single solution, namely* $\beta_i^v = \beta_i^{v-w_i} = 0$. *This is equivalent to additional equation constrains in the formulation and hense reduction of the dimension of the face* $\{(\alpha,\beta)\in \mathrm{conv}(S) : \sum_{i\in I}\beta_i^v = \mu^H\sum_{i\in I_v}\beta_i^{v-w_i}\}$.

Finally for this section, we propose the following theorem.

Theorem 1.8. *The proposed in this paper facet-defining inequalities are not the full description of* $\mathrm{conv}(S)$.

Proof. We prove the theorem numerically. For the test we took the instance *ngcut04* from [MMV03] with the following data:

$$W = 10,\ H = 20,\ m = 7,\ w = (3,3,2,2,2,2,1),\ h = (8,7,15,15,12,12,9).$$

Then we generate all facet-defining inequalities from this section and them to the formulation of the problem (1.1)-(1.7). Then we have solved the root LP relaxation of the problem with the primal simplex method and get the following results:

1. The value of the optimal solution of the LP relaxation: 16.2.
2. The values of the variables α_i^u:

i \ u	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.67												0.33							
2	0.44												0.56							
3	1																			
4	1																			
5	0.33								0.67											
6	1																			
7									1											

$\tilde{\alpha}_i^u$ values:

i \ u	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7					0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
2	0.4	0.4	0.4	0.4	0.4	0.4	0.4						0.6	0.6	0.6	0.6	0.6	0.6	0.6	
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
5	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	1	1	1	1	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
6	1	1	1	1	1	1	1	1	1	1	1	1								
7									1	1	1	1	1	1	1	1	1			

3. The values of the variables β_i^v:

i \ v	1	2	3	4	5	6	7	8	9	10
1			0.29					0.71		
2	0.40			0.21			0.39			
3	0.54				0.46					
4	0.35		0.16			0.49				
5				0.46		0.01	0.50		0.03	
6			0.38					0.15	0.47	
7			0.46			0.03				0.50

$\tilde{\beta}_i^v$ values:

i \ v	1	2	3	4	5	6	7	8	9	10
1			0.3	0.3	0.3			0.7	0.7	0.7
2	0.4	0.4	0.4	0.2	0.2	0.2	0.4	0.4	0.4	
3	0.5	0.5			0.5	0.5				
4	0.4	0.4	0.2	0.2		0.5	0.5			
5				0.5	0.5		0.5	0.5		
6			0.4	0.4				0.2	0.6	0.5
7			0.5							0.5

The optimal solution of the LP relaxation in the root node is not integer within the margin of rounding error. Hence, the theorem follows.

□

1.3 The branch-and-cut algorithm

Here we describe the overall algorithm of finding an optimum of SPP-2. The algorithm is *branch-and-cut* which is branch-and-bound in which cutting planes are generated throughout the branching tree. The idea is the same as in branch-and-bound, we calculate lower bounds for each node and use it as a pruning criteria. In comparison to the branch-and-bound, in branch-and-cut we invest as much efforts as possible until the limits allow in order to strengthen the dual bound for the node.

In practice we search for a balance between the number of nodes to process and the number of cutting planes which are added at each node. From one hand, if we add to many cutting planes at each node, the optimization becomes slower. From the other hand, if we add proper cutting planes, we increase the dual lower bound hence reduce the number of descendant nodes. The other issue is the management of the branching tree. For the branch-and-bound we store only the bounds for each variable and processed to the descendants by adding one bound-constraint. For the branch-and-cut we use a *cut-pool* in which all generated cuts are stored. So, in addition to keeping the bounds and the indexes of variables in the basis it is also necessary to indicate which constraints are needed to reconstruct the formulation at the given node.

Algorithm 1.2 (BRANCH-AND-CUT)**.** *The overall optimization algorithm.*
Input data: W, m, $w = (w_1, \ldots, w_m)$, $h = (h_1, \ldots, h_m)$.
Output data: optimal solution $(\underline{\alpha}, \underline{\beta})^*$ *and its value* $\underline{\phi}$.

(*1*) Initialization*:*

- *node* j_0 *with formulation* $F^{j_0} := \{(1.2) - (1.6)\}$*;*
- $\phi = \min\{f : \ (\alpha, \beta) \in \mathbb{R}_+^{m(W+H)}, \ F^{j_0} \ holds\}$, $\underline{\phi} = \infty$*;*
- *incumbent* $(\underline{\alpha}, \underline{\beta})^* = \emptyset$*;*
- *node list* $N := \{j_0\}$.

(*2*) Node*: If* $N = \emptyset$*, go to* Exit*. Else choose* $j \in N$*, set* $N := N \backslash \{j\}$ *and go to* Restore.

(*3*) Restore *the formulation* F^j*. Set* $k := 1$ *and* $F^{j,1} = F^j$.

(*4*) LP relaxation*: Iteration* k*. Solve*

$$\phi^{j,k} = \min\{f : \ (\alpha, \beta) \in \mathbb{R}_+^{m(W+H)}, \ F^{j,k} \ holds\}.$$

If infeasible, prune and go to Node*. Else solution* $(\underline{\alpha}, \underline{\beta})^{j,k}$ *and go to* Cut.

(*5*) Cut*: Iteration* k*. In order to cut off* $(\underline{\alpha}, \underline{\beta})^{j,k}$ *solve separation problems:*

(*5.1*) NOV-SEPARATION*. If found a cut then add the cut to* $F^{j,k}$*:*

$$F^{j,k+1} := F^{j,k1} \cup \{\sum_{i \in M} [\tilde{\alpha}_i^u + \tilde{\beta}_i^v] \leq |M| + 1\}.$$

(*5.2*) CO-SEPARATION*. If found a cut then apply* LIFTING*. Add the cut to* $F^{j,k}$*:*

$$F^{j,k+1} := F^{j,k1} \cup \{\sum_{i \in C(u)} \tilde{\alpha}_i^u + \sum_{i \in I \backslash C(u)} \pi_i \tilde{\alpha}_i^u \leq |C(u)| - 1\}.$$

If $\phi^{j,k} \geq \underline{\phi}$*, go to* Node*. If no cuts found OR* $k > k_{\max}$ *then go to* Prune*. Else set* $k := k + 1$ *and go to* LP relaxation.

(*6*) Prune*: If* $(\underline{\alpha}, \underline{\beta})^{j,k} \in S$*, set* $\underline{\phi} = \phi^{j,k}$*, update the incumbent* $(\underline{\alpha}, \underline{\beta})^* := (\underline{\alpha}, \underline{\beta})^{j,k}$ *and go to* Node*, else go to* Branching.

(*7*) Branching*: Select a variable* α_i^u*:* $\epsilon < \underline{\alpha}_i^u < 1 - \epsilon$ *OR* β_i^v*:* $\epsilon < \underline{\beta}_i^v < 1 - \epsilon$ *in* $(\underline{\alpha}, \underline{\beta})^{j,k}$ *and create two descendants:*

(*7.1*) *node* j_1 *with formulation* $F^{j_1} := F^{j,k} \cup \{\alpha_i^u = 0 \ OR \ \beta_i^v = 0\}$,
(*7.2*) *node* j_2 *with formulation* $F^{j_2} := F^{j,k} \cup \{\alpha_i^u = 1 \ OR \ \beta_i^v = 1\}$.

Set $N := N \cup \{j_1, j_2\}$*. Go to* Node.

(*8*) Exit*: Return the incumbent* $(\underline{\alpha}, \underline{\beta})^*$ *and the optimal value* $\underline{\phi}$.

In BRANCH-AND-CUT for the case when there is no improvement of the LP bound after addition of cuts within $k_{\max}$ iterations, the algorithm proceeds to the prune and the branching procedures.

1.3.1 Separation for non-overlapping inequalities

Now let $\mathbb{F}$ be the family of non-overlapping inequalities (1.21) for S, and let us examine the non-overlapping separation problem for $\mathbb{F}$. Explicitly we are given a nonintegral point $(\underline{\alpha}, \underline{\beta})$, i.e., $\exists i \in I$, $\exists u \in U$ or $\exists v \in V$:

$$\epsilon < \underline{\alpha}_i^u < 1 - \epsilon \quad \text{or} \quad \epsilon < \underline{\beta}_i^v < 1 - \epsilon, \tag{1.62}$$

for a sufficient small $\epsilon > 0$. And now we wish to know whether this point satisfies all the non-overlapping inequalities (1.21) which we rewrite in the following form:

$$\sum_{i \in M} (\tilde{\alpha}_i^u + \tilde{\beta}_i^v - 1) \leq 1.$$

In order to answer the question we check whether there exists a set $M \subseteq I$ for which $\sum_{i \in M} (\tilde{\underline{\alpha}}_i^u + \tilde{\underline{\beta}}_i^v - 1) \geq 1 + \epsilon$. Since the set M in this case is unknown, we formulate the following problem:

$$M(u,v) := \{i \in I : \ \tilde{\underline{\alpha}}_i^u + \tilde{\underline{\beta}}_i^v - 1 \geq \epsilon\}, \quad \delta(u,v) = \sum_{i \in M(u,v)} [\tilde{\underline{\alpha}}_i^u + \tilde{\underline{\beta}}_i^v - 1]. \tag{1.63}$$

Theorem 1.9. *(a) If $\forall (u,v) \in U \times V$ $\delta(u,v) \leq 1$, then $(\underline{\alpha}, \underline{\beta})$ satisfies all the non-overlapping inequalities.*
(b) If $\exists (u,v) \in U \times V$ $\delta(u,v) \geq 1 + \epsilon$ then the inequality $\sum_{i \in M} (\tilde{\alpha}_i^u + \tilde{\beta}_i^v) \leq |M| + 1$ cuts off $(\underline{\alpha}, \underline{\beta})$ by an amount $\delta(u,v) - 1$.

Proof. [Wol98]. □

Algorithm 1.3 (NOV-SEPARATION)**.** *Determination of the most violated inequality of type* (1.21).
Input data: $(\underline{\alpha}, \underline{\beta})$.
Output data: Most violated inequality of type (1.21).

(*1*) *For $u \in U$ if $\tilde{\underline{\alpha}}^u \neq \tilde{\underline{\alpha}}^{u-1}$ ($\tilde{\underline{\alpha}}^1 \neq \tilde{\underline{\alpha}}^0$):*

  (*1.1*) *For $v \in V$ if $\tilde{\underline{\beta}}^v \neq \tilde{\underline{\beta}}^{v-1}$ ($\tilde{\underline{\beta}}^1 \neq \tilde{\underline{\beta}}^0$): Solve* (1.63) *and select (u,v) with the maximal $\delta(u,v)$.*

(*2*) *If $\delta(u,v) \geq 1 + \epsilon$ then return:*

$$\sum_{i \in M} (\tilde{\alpha}_i^u + \tilde{\beta}_i^v) \leq |M| + 1,$$

else return $\emptyset$.

Remark 1.7. *The complexity of* NOV-SEPARATION *is $O(mWH)$.*

1.3.2 Separation for cover inequalities

Now let $\mathbb{G}$ be the family of cover inequalities (1.50) for S, and let us examine the cover separation problem for $\mathbb{F}$. Suppose we are given a nonintegral point $(\underline{\alpha}, \underline{\beta})$ with (1.62) and we wish to know whether this point satisfies all the cover inequalities (1.50) which we rewrite in the following form:

$$\sum_{i \in C(u)} (1 - \tilde{\alpha}_i^u) \geq 1.$$

In order to answer the question we check whether there exists a set $C(u) \subseteq I$ with $\sum_{i \in C(u)} w_i > W$ for which $\sum_{i \in C(u)} (1 - \underline{\tilde{\alpha}}_i^u) \leq 1 - \epsilon$. Since the set $C(u)$ in this case is unknown, we formulate the following 0-1 integer program where the variable $\gamma_i = 1$ if $i \in C(u)$ and $\gamma_i = 0$ otherwise:

$$\rho(u) = \min \sum_{i \in I} (1 - \underline{\tilde{\alpha}}_i^u) \gamma_i; \quad \text{s.t.} \tag{1.64}$$

$$\sum_{i \in I} w_i \gamma_i > W; \tag{1.65}$$

$$\gamma_i \in \{0, 1\}, \quad i \in I. \tag{1.66}$$

Theorem 1.10. *(a) If $\forall u \in U$ $\rho(u) \geq 1$ then $(\underline{\alpha}, \underline{\beta})$ satisfies all the cover inequalities. (b) If $\exists u \in U$ $\rho(u) \leq 1 - \epsilon$ with optimal solution $\bar{\gamma}$ then $C = \{i \in I : \bar{\gamma}_i = 1\}$ is a cover and the cover inequality $\sum_{i \in C} \tilde{\alpha}_i^u \leq |C| - 1$ cuts off $(\underline{\alpha}, \underline{\beta})$ by an amount $\rho(u) - 1$.*

Proof. [Wol98]. □

Algorithm 1.4 (CO-Separation)**.** *Determination of the most violated cover inequality.*
Input data: $(\underline{\alpha}, \underline{\beta})$.
Output data: Most violated cover inequality.

(1) *For $u \in U$ solve* (1.64)-(1.66) *and select the cover $C = \{i \in I : \bar{\gamma} = 1\}$ for the maximal $\rho(u)$.*
(2) *If $\rho(u) \leq 1 - \epsilon$ return:*

$$\sum_{i \in C} \tilde{\alpha}_i^u \leq |C| - 1,$$

else return $\emptyset$.

Remark 1.8. *The complexity of* CO-Separation *is $O(mW^2)$.*

1.4 Valid linear inequalities

Here we describe three classes of valid linear inequalities which are valid for $\text{conv}(S)$. They are not as strong as facet-defining inequalities but they can remove some equivalent solutions in terms of SPP-2. Eventually, when we explore the branching tree we

want to find one optimal solution and prevent exploring all others which are in some sense equivalent.

All equations and inequalities are considered as *preprocessing* and applied in the root node of the branching tree, i.e., this is equivalent to the extension of the formulation (1.1)-(1.7) of SPP-2 with these constraints.

1.4.1 Raster points equations

Sometimes there is no use considering all values of U and V for possible allocation of items. The values which are of interest are called *raster points* [Sch08]. They are calculated as follows:

$$R_i^w := \{0 \le k \le W : \ k = \sum_{i \in I \setminus \{i\}} w_i \gamma_i, \ \gamma_i \in \{0,1\}, \ i \in I \setminus \{i\}\}, \quad i \in I,$$
$$R_i^h := \{0 \le k \le H : \ k = \sum_{i \in I \setminus \{i\}} h_i \gamma_i, \ \gamma_i \in \{0,1\}, \ i \in I \setminus \{i\}\}, \quad i \in I.$$

In the mentioned book, the author proposes an approach of a reduction of the number of raster points based on a fixed upper bound for the points by consideration of a *reduced set* of raster points. Since we minimize f-variable in the formulation (1.1)-(1.7) and possible allocation points over the $(0, H)$-axis depend on f then we can apply the reduction only for R_i^w:

$$\widetilde{R}_i^w := \{\max\{k \in R_i^w : \ k \le W - r\} : \ r \in R_i^w\}.$$

In order to exclude the non-raster points from the consideration we extend the formulation (1.1)-(1.7) by the following equations:

$$\alpha_i^u = 0, \quad i \in I, \ u \in U \setminus R_i^h, \tag{1.67}$$
$$\beta_i^v = 0, \quad i \in I, \ v \in V \setminus \widetilde{R}_i^w. \tag{1.68}$$

Remark 1.9. *Since we extend the formulation by equality constraints we reduce the dimension of* $\mathrm{conv}(S)$*. It is easy to show using points* (1.17)-(1.19) *that*

$$\dim(\mathrm{conv}(S)) = m(W+H) - \sum_{i \in I}[w_i + h_i] - \sum_{i \in I}[|V \setminus \widetilde{R}_i^w| + |H \setminus R_i^h|].$$

1.4.2 Symmetry elimination equations

The idea of this elimination is the following. Let $(\alpha, \beta)_0 = (\alpha_1^1, \ldots, \alpha_1^H, \ldots, \alpha_m^1, \ldots, \alpha_m^H, \beta_1^1, \ldots, \beta_1^W, \ldots, \beta_m^1, \ldots, \beta_m^W)$ be an optimal solution with value $f := H_0$ and $\alpha_i^{k_i} := 1$, $\beta_i^{l_i} := 1$, and $\alpha_i^k := 0$ for $k \in U \setminus \{k_i\}$, $\beta_i^l := 0$ for $l \in V \setminus \{l_i\}$, $i \in I$. Let us introduce the following two operations which transform $(\alpha, \beta)_0$:

1. $(0, H)$-symmetrical mapping:

$$\bar{\alpha}_i^k := \begin{cases} 1, \ k = H_0 - k_i; \\ 0, \ \text{otherwise}; \end{cases} \qquad \bar{\beta}_i^l := \begin{cases} 1, \ l = l_i; \\ 0, \ \text{otherwise}, \end{cases} \qquad \forall i \in I.$$

2. $(0, W)$-symmetrical mapping:

$$\bar{\alpha}_i^k := \begin{cases} 1, & k = k_i; \\ 0, & \text{otherwise}; \end{cases} \qquad \bar{\beta}_i^l := \begin{cases} 1, & l = W - l_i; \\ 0, & \text{otherwise}, \end{cases} \qquad \forall i \in I.$$

Both of the transformations yield solutions with the same $f = H_0$. Since it is enough to find at least one of them we are interested in forbidding the three others. This can be done by requiring of an item to be allocated in the first half of the packing over the $(0, H)$-, and $(0, W)$-axes, respectively. Numerically it is more effective to apply this for the largest item.

Let $I^w = \{i \in I : \ w_i = \max_{k \in I}\{w_k\}\}$ be the set of items with the largest width and $I^h = \{i \in I^w : \ h_i = \max_{k \in I^w}\{h_k\}\}$ be the set of those from I^w with the largest height. Then for $i^* := \text{argmin}\{I^h\}$ the following constraints are applied for the formulation:

$$\sum_{u \in U^*} \alpha_{i^*}^u = 1, \quad U^* := \{1, \ldots, \min\{\lfloor \frac{H - h_{i^*}}{2} \rfloor + 1, H - h_{i^*} + 1\}\}; \tag{1.69}$$

$$\sum_{v \in V^*} \beta_{i^*}^v = 1, \quad V^* := \{1, \ldots, \min\{\lfloor \frac{W - w_{i^*}}{2} \rfloor + 1, W - w_{i^*} + 1\}. \tag{1.70}$$

Remark 1.10. *Constraints* (1.69)-(1.70) *are equivalent to:*

$$\alpha_{i^*}^u = 0, \quad u = \min\{\lfloor \frac{H - h_{i^*}}{2} \rfloor + 1, H - h_{i^*} + 1\} + 1, \ldots, H - h_{i^*} + 1; \tag{1.71}$$

$$\beta_{i^*}^v = 0, \quad v = \min\{\lfloor \frac{W - w_{i^*}}{2} \rfloor + 1, W - w_{i^*} + 1\} + 1, \ldots, W - w_{i^*} + 1, \tag{1.72}$$

the formulation (1.1)-(1.7) *extended with. It is easy to show that the face* $F := \{(\alpha, \beta) \in \text{conv}(S) : \ (1.71) - (1.72)$ *hold*$\}$ *has dimension:*

$$\dim(\text{conv}(F)) = m(W + H) - \sum_{i \in I}[w_i + h_i] - |U \setminus U^*| - |V \setminus V^*|.$$

1.4.3 Combination inequalities

Here we use some basic facts for the allocation of items: items with the width laying in interval $(\frac{W}{2}, W]$ can not be allocated together at one position in $(0, H)$; at most two items with the width laying in interval $(\frac{W}{3}, \frac{W}{2}]$ can be allocated at one position in $(0, H)$.

Let for $p \in V$:

$$I_p^w := \{i \in I : \ \frac{W}{p+1} < w_i \leq \frac{W}{p}\}$$

be the set of items which sizes are the fractions of p. The following inequalities hold:

$$\sum_{i \in I_p^w} \tilde{\alpha}_i^u \leq p, \quad u \in U, \ p \in V, \ I_p^w \neq \emptyset; \tag{1.73}$$

From the other hand, the following inequalities are also valid:

$$\sum_{i \in I_p^*} \tilde{\alpha}_i^u \leq p, \quad I_p^* := \bigcup_{j=1}^{p} I_j^w, \ u \in U, \ p \in V, \ I_p^w \neq \emptyset. \tag{1.74}$$

Remark 1.11. *Let us discuss now the dimension of the face* $F := \{(\alpha, \beta) \in \mathrm{conv}(S)$*:* $\sum_{i \in I_1^w} \tilde{\alpha}_i^u = 1\}$ *based on the combination inequality* (1.73) *for* $p = 1$*,* $I_1^w \neq \emptyset$*, and an* $u \in U$*. Equation* $\sum_{i \in I_1^w} \tilde{\alpha}_i^u = 1$ *means that the following constraints hold in the formulation:*

1. $i^* := \underset{i \in I_1^w}{\mathrm{argmin}}\, h_i$ *and* $i \in (I_1^w \setminus \{i^*\}) \cup \{i \in I \setminus I_1^w : \; w_i > W - \min\limits_{i \in I_1^w} w_i\}$*:*

$$\alpha_i^k = 0, \quad k \in \{\max\{1, u - h_i\}, \ldots, u-1, u+1, \ldots, \min\{u + h_{i^*}, H - h_{i^*} + 1\}\}.$$

2. *For* $\bar{i^*} := \underset{i \in I_1^w \setminus \{i^*\}}{\mathrm{argmin}}\, h_i$ *and* i^**:*

$$\alpha_{i^*}^k = 0, \quad k \in \{\max\{1, u - h_{i^*}\}, \ldots, u-1, u+1, \ldots, \min\{u + h_{\bar{i^*}}, H - h_{\bar{i^*}} + 1\}\}.$$

So, in general case F *is not a facet.*

1.4.4 Dominating knapsack inequalities

Here we tighten the inequalities (1.4). Let us consider the following question: what is the maximal occupation of a knapsack with capacity W, if an item $i \in I$ is present? In order to answer to this question we formulate the following 0-1 integer program which is also known as a subset sum problem, $W^j := \max\{\sum_{i \in I} w_i \gamma_i : \; \sum_{i \in I} w_i \gamma_i \leq W, \; \gamma_j = 1, \; \gamma_i \in \{0, 1\}\}$ where W^j is maximal occupation.

If the maximal occupation W^j is less than W then we can tighten the knapsack condition by following inequality:

$$\sum_{i \in I \setminus \{j\}} w_i \tilde{\alpha}_i^u + (W - W^j)\tilde{\alpha}_j^u \leq W, \quad u \in U. \tag{1.75}$$

Lemma 1.11. *Inequality* (1.75) *dominates* (1.4) *if* $W^j < W - w_j$*.*

Proof. If $\alpha_j^u = 0$ then inequalities (1.75) and (1.4) are equal. If $\alpha_j^u = 1$ then we have $W^j \leq u(W - w_j)$. Let $s := \frac{W^j}{W - w_j}$. According to the definition of a dominating inequality, $(w_1, \ldots, w_{j-1}, w_{j+1}, \ldots, w_m)^T \geq s(w_1, \ldots, w_{j-1}, w_{j+1}, \ldots, w_m)^T$ should hold. Hence, $W^j < W - w_j$ follows. □

1.5 Valid nonlinear inequalities and linearization

In this section we give some notes for the nonlinear modeling of constraints for SPP-2. We linearize and apply some of the them within the formulation (1.1)-(1.7).

1.5.1 Non-overlapping inequalities

Here we consider different approaches for the non-overlapping constraints. These constraints are similar to (1.20).

Let $I_\alpha(u) := \{i \in I : \ \tilde{\alpha}_i^u > 0\}$ and $I_\beta(v) := \{i \in I : \ \tilde{\beta}_i^v > 0\}$ be the sets of items which are allocated at $u \in U$ and $v \in V$, respectively. The following inequalities are valid:

$$|I_\alpha(u) \cap I_\beta(v)| \leq 1, \quad u \in U, \ v \in V,$$

which leads to

$$\sum_{i \in I} [\tilde{\alpha}_i^u \tilde{\beta}_i^v] \leq 1, \quad u \in U, \ v \in V. \tag{1.76}$$

Non-overlapping inequalities in a rectangular area with minimal sizes

Based on the idea which was before constraints (1.20), the following inequalities hold:

$$\sum_{i \in I_\beta(v)} \tilde{\alpha}_i^u \leq 1, \quad \sum_{i \in I_\alpha(u)} \tilde{\beta}_i^v \leq 1, \quad u \in U, \ v \in V. \tag{1.77}$$

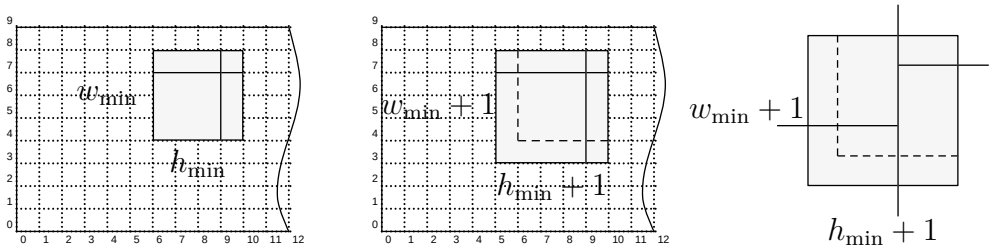


Figure 1.8: Rectangular areas with sizes: (a) – $w_{\min} \times h_{\min}$; (b) – $(w_{\min}+1) \times (h_{\min}+1)$. (c) – Items which fit into the area with sizes $(w_{\min}+1) \times (h_{\min}+1)$.

In order to simplify the following description let us define the following sets:

$$\tilde{I}_\alpha(u) = \{i \in I : \sum_{k \in \widetilde{H}_i(u)} \alpha_i^k > 0\}, \quad \tilde{I}_\beta(v) = \{i \in I : \sum_{k \in \widetilde{W}_i(v)} \beta_i^k > 0\},$$

where

$$\widetilde{H}_i(u) = \{\max\{1, u - h_{\min} - h_i + 1\}, \ldots, \min\{u, H - h_i + 1\}\},$$
$$\widetilde{W}_i(v) = \{\max\{1, v - w_{\min} - w_i + 1\}, \ldots, \min\{v, W - w_i + 1\}\}.$$

Suppose we distinguish a rectangular area with sizes $(w_{\min} \times h_{\min})$, see Fig. 1.8a. Obviously at most 4 items can fit into this area, see Fig. 1.8c. The same holds for rectangular area with sizes increased by 1, see Fig. 1.8b. Hereby the following constraints hold:

$$\sum_{i \in \widetilde{I}_\alpha(u)} \sum_{k \in \widetilde{W}_i(v)} \beta_i^k \leq 4, \qquad \sum_{i \in \widetilde{I}_\beta(v)} \sum_{k \in \widetilde{H}_i(u)} \alpha_i^k \leq 4, \qquad u \in U, \ v \in V. \tag{1.78}$$

Here is a another formulation of the above observation:

$$|\widetilde{I}_\alpha(u) \cap \widetilde{I}_\beta(v)| \leq 4. \tag{1.79}$$

Non-overlapping constraints in a rectangular area

Let $\widetilde{w} := \min_{i\in I}\{w_i : \ h_i = h_{\min}\}$ and $k_{\widetilde{w}} := \max\{\sum_{i\in I} a_i : \ \sum_{i\in I} w_i a_i \leq \widetilde{w} - 2, \ a_i \in \{0,1\}\}$. To simplyfy the following description let us introduce the following sets:

$$\begin{aligned}
&\hat{U}_i(u) := \{\max\{1, u - h_{\min} - h_i + 2\}, \ldots, \min\{u, H - h_i + 1\}\};\\
&\hat{V}_i(v) := \{\max\{1, v - \widetilde{w} - w_i + 2\}, \ldots, \min\{v, W - w_i + 1\}\};\\
&\hat{I}_W(v) = \{i \in I : \sum_{v\in\hat{V}_i(v)} \beta_i^v = 1\}.
\end{aligned}$$

For each rectangular area $\{(x,y) : \ u - h_{\min} + 1 \leq x \leq u, \ v - \widetilde{w} + 1 \leq y \leq v\}$ with $u \in U$, $v \in V$ the following inequality holds:

$$\sum_{i\in\hat{I}_W(v)} \sum_{u\in\hat{U}_i(u)} \alpha_i^u \leq 2k_{\widetilde{w}} + 4. \tag{1.80}$$

Remark 1.12. *Note if $h_{\min} = 1$ then the right-hand side of* (1.80) *can be replaced by $k_{\widetilde{w}}$ + 2.*

Remark 1.13. *Note that the idea can be generalized for any rectangular area. In this case the inequality would be weaker. If for $w_1 < w_2 \leq w_i$ with $i \in I$ then w_1 can be replaced by $w_1 + 1$ in the rectangular area; h_1 can be replaced with $h^* \geq h_1$ with $h^* = \max\{h : \ \mu^H = k_{\widetilde{h}}\} + 2$.*

1.5.2 Left-lowest allocation equation

Here without loss of generality we assume that one of the items is allocated at the left-lowest position. This results in $\alpha_i^1\beta_i^1 = 1$ for an $i \in I$:

$$\sum_{i\in I} \alpha_i^1\beta_i^1 = 1. \tag{1.81}$$

Since (1.81) is nonlinear, we consider the following linear approximation. Obviously (1.81) can be approximated by $\sum_{i\in I} \min\{\alpha_i^1, \beta_i^1\} = 1$. In order to linearize it we introduce new variables $\gamma_i = \min\{\alpha_i^1, \beta_i^1\}$. Herewith we get the following linearization for (1.81):

$$\begin{aligned}
&\sum_{i\in I}\gamma_i = 1;\\
&0 \leq \gamma_i \leq \alpha_i^1;\\
&0 \leq \gamma_i \leq \beta_i^1;\\
&\gamma_i \geq \alpha_i^1 + \beta_i^1 - 1.\\
&\gamma_i \in \{0,1\}, \quad i \in I.
\end{aligned}$$

1.5.3 Lowest allocation

Here without loss of generality we require that when an item has no bottom neighbor then it is adjacent to the bottom side of the strip.

Let $\mu(w) := \max\{|K| : K \subseteq I, \sum_{i\in K} w_i \leq w-1\}$ be the maximal number of items which fit in $w-1$. The following inequality holds:

$$\sum_{i\in I}[\sum_{u=1}^{h_{\min}} \alpha_i^u \sum_{v=1}^{w} \beta_i^v] \leq \mu(w)+1, \quad w \in V. \tag{1.82}$$

1.6 Numerical study

In this section we discuss numerical experiments for SPP-2 instances from different sources.

The algorithm was implemented as a multi-threaded application in C++ based on gcc 4.1.2, on an Intel Xeon X5670 (2.93 GHz) CPU. IBM ILOG CPLEX 12.5 was used as an LP solver. The test instances, detailed results and source code are available on the CaPaD website[1].

Separation problem CO-Separation from Section 1.3 is solved by the dynamic programming approach with strong bounds [MPT99], implementation of which was taken from the personal website[2] of D. Pisinger.

In Tables 1.2-1.3 the number of nodes and time are the mean values over the solved instances. From a rational number we take only the integer part without rounding.

Here are the following implementation issues to consider:

1. Time limit for each instance and method was set to 1800 seconds.

2. Per one iteration we add only one cut which is at most violated. Then the resolution of the LP relaxation follows.

3. The generated cuts are never deleted from the formulation, even when moving from one part of the branching tree to a completely other part.

4. We do not consider any methods of stabilization or optimization of the cuts' generation, since this is not a subject of the research of this paper. Considering of these steps may result in a better computational behavior of the branch-and-cut method.

1.7 Conclusions

Here we have proposed and studied a new formulation of the 2D strip packing problem and a new branch-and-bound method.

The main theoretical and experimental observations of the paper are the following:

[1] http://www.math.tu-dresden.de/∼capad
[2] http://www.diku.dk/∼pisinger

1. Under appropriate assumptions we proposed two classes of facet-defining inequalities: general non-overlapping and density.

2. Numerical testes show the better stability of the branch-and-cut algorithm. Numerical tests also show the numerical effectiveness of the found inequalities: the percentage of optimally solved instances from [CJM08] has increased from 85% to 95%, the solution time has declined by the factor 10. For the instances from [MMV03] and [Hop00, HT00], the number of optimally solved instances increased and solution time has declined by the factor 1,9.

3. The found facet-defining inequalities are still not enough to described the convex hull of feasible integer solutions.

4. The following issues are subject to further study: what is the best cuts' addition strategy; further valid and facet-defining inequalities; possibility of extension of the facet-defining inequalities from [HNS08] for the 2D strip packing problem; decomposition approach?

1.8 Acknowledgments

We thank David Pisinger for the provided code for the solution of 0-1 knapsack problems. We appreciate the Academic Initiative of IBM which enables many researchers all over the world to compare their methods using state-of-the-art IBM ILOG Optimization Software.

Table 1.2: Results of the 2D instances from [CJM08]: f – value of the goal function; n_1 – number of nodes for the incumbent; t_1 – time for the incumbent; n_2 – total number of nodes; t_2 – total time; cuts – is the number of added cuts. * The problem 00X23 was solved optimally by the branch-and-cut method with $\max_{M \subseteq I} |M| \leq 10$, $n_2 = 5407303$, $t_2 = 231$.

	CPLEX, pure (1.1)-(1.7)						branch-and-cut						
inst	opt	f	n_1	t_1	n_2	t_2	opt	f	n_1	t_1	n_2	t_2	cuts
Infeasible instances													
00N10	1	22	0	1	415	2	1	22	735	0	6304	0	402
00N15	1	21	388	49	466	53	1	21	10084	5	23611	6	73
00N23	0	21	973	865	-	-	0	21	35499	116	-	-	-
00X23	0	21	2732	1053	-	-	0	21	42222	154	-*	-*	-
02N20	0	22	0	98	-	-	1	22	5572	26	18988	22	284
03N10	1	21	0	1	0	1	1	21	215	0	216	0	36
03N15	1	21	494	72	8510	1752	1	21	7076	5	80592	10	599
03N16	0	21	936	117	-	-	1	21	56435	12	90487	17	1963
03N17	1	21	22	57	166	98	1	21	2318	7	101953	17	205
04N15	1	22	0	11	7648	493	1	22	8620	2	52069	5	36
04N17	0	21	981	116	-	-	1	21	7232	6	15214	9	39
04N18	0	21	900	160	-	-	1	21	1858	5	257619	23	173
05N15	1	21	409	92	4786	496	1	21	25470	6	293635	82	2363
05N17	1	21	288	209	10568	1231	1	21	300	4	17020	5	76
05X15	1	21	810	201	18430	1186	1	21	11492	10	82506	20	31
07N10	1	22	0	1	106	6	1	22	4582	1	17400	2	54
07N15	1	21	0	7	977	57	1	21	3423	1	8100	2	2184
07X15	1	21	558	103	987	158	1	21	8924	6	11964	7	75
08N15	1	21	0	6	41	14	1	21	1538	2	2135	2	2103
10N10	1	21	0	1	97	1	1	21	269	0	1835	0	247
10N15	1	22	0	3	6086	51	1	22	1379	1	485540	31	172
10X15	1	21	239	22	291	32	1	21	9108	2	10182	3	8
13N10	1	21	0	3	160	6	1	21	871	1	3820	1	1313
13N15	1	22	0	3	133	11	1	22	703	1	60417	6	282
13X15	1	22	19	13	10931	816	1	22	2757	3	10803	5	68
15N10	1	21	23	4	279	5	1	21	2407	1	5532	1	1188
15N15	1	22	0	4	894	28	1	22	698	1	14266	2	4
mean	21		362	121	3427	309	25		9325	14	66888	11	559
Feasible instances													
02F17	1	20	9745	1721	9745	1720	1	20	32846	9	32846	7	799
02F20	1	20	3828	747	3828	747	1	20	698383	353	949132	675	7631
02F22	1	20	1809	539	1809	539	1	20	7992	50	7992	14	5409
03X18	1	20	2103	232	2103	232	1	20	73488	21	73488	17	1253
04F15	1	20	9579	1798	9579	1797	1	20	25223	9	25223	8	709
04F17	1	20	1085	90	1085	90	1	20	51443	30	56412	39	6106
04F19	1	20	3046	1410	3046	1410	1	20	17222	30	773969	287	3535
04F20	1	20	590	294	590	294	1	20	183342	52	340128	156	5062
05F15	1	19	2350	114	2350	114	1	19	12670	8	12670	6	753
05F18	1	20	1860	343	1965	790	1	20	105855	31	105855	27	4734
05F20	1	20	0	14	3	116	1	20	2756	21	2756	6	3604
07F15	1	20	697	106	697	106	1	20	3644	2	3644	1	2362
08F15	1	20	1427	351	4769	1333	1	20	24580	8	28231	7	1327
20F15	1	17	592	63	613	92	1	17	7169	4	7766	2	1942
20X15	1	20	0	8	0	8	1	20	2036	4	2036	2	1727
mean	15		2581	522	2812	626	15		83243	42	161477	83	3130

Table 1.3: Results of the 2D instances from [MMV03] and [Hop00, HT00]: f – value of the goal function; n_1 – number of nodes for the incumbent; t_1 – time for the incumbent; n_2 – total number of nodes; t_2 – total time; cuts – is the number of added cuts.

			CPLEX, pure (1.1)-(1.7)						branch-and-cut						
inst	m	W	opt	f	n_1	t_1	n_2	t_2	opt	f	n_1	t_1	n_2	t_2	cuts
ht01	16	20	1	20	184	60	184	60	1	20	5127	4	5127	2	1610
ht02	17	20	1	20	839	321	839	320	1	20	10966	5	10966	2	95
ht03	16	20	1	20	270	132	270	132	1	20	2613	4	2613	2	686
cgcut01	16	10	1	23	381	22	381	22	1	23	7720	2	11560	4	3
ngcut01	10	10	1	23	0	0	0	0	1	23	0	0	1364	1	28
ngcut02	17	10	1	30	0	4	66	15	1	30	1268	2	2497	1	327
ngcut03	21	10	-	-	-	-	-	-	0	28	771072	88	-	-	-
ngcut04	7	10	1	20	0	0	0	0	1	20	0	0	422	0	3
ngcut05	14	10	1	36	0	2	0	2	1	36	2991	2	30684	8	142
ngcut06	15	10	1	31	608	40	790	45	1	31	4760	1	6538	1	621
ngcut07	8	20	1	20	0	0	0	0	1	20	0	0	0	0	0
ngcut08	13	20	0	33	838	146	-	-	1	33	49010	6	56893	6	1209
ngcut09	18	20	-	-	-	-	-	-	0	50	491526	66	-	-	-
ngcut10	13	30	1	80	0	36	1582	137	1	80	2364	2	31212	3	2
ngcut11	15	30	-	-	-	-	-	-	1	52	743638	299	1080259	602	5823
ngcut12	22	30	-	-	-	-	-	-	1	87	18828	83	2105293	681	3135
C1_1	16	20	1	20	184	238	184	238	1	20	5127	4	5127	2	1610
C1_2	17	20	1	20	839	882	839	882	1	20	10966	4	10966	2	95
C1_3	16	20	1	20	270	448	270	448	1	20	2613	4	2613	1	686
mean			14	0	294	155	386	164	17	0	112136	30	197890	78	946

Chapter 2

Branch-and-Price Methods for the 1D Contiguous Bin Packing Problem

We consider the 1D contiguous bin packing problem (CBPP-1). Given a set of 1D items, CBPP-1 is to find a packing of all items into the 1D bins in a fixed order which uses the minimal number of the bins satisfying restrictions of *heterogeneity* (each item of one type is packed at most ones into a bin) and *contiguity* (each item of one type is packed contiguously into a sequence of bins). We use a Gilmore-Gomory model for the 1D bin packing problem and expose additional restrictions due to contiguity. To be a feasible model, the basis matrix of the solution must have a *consecutive 1's* property (C1P). We use the known results of A. Tucker on matrices with the C1P and develop new branch-and-price algorithms for construction of such matrices. The algorithms are distinguished by various strategies for the enumeration of partial solutions. We also prove some characteristics of partial solutions, which tightens the slave problem of the column generation. We conclude with the discussion of the numerical results.
Keywords: column generation, branch-and-price, consecutive 1's property

2.1 Introduction

The 1-*dimensional contiguous bin packing* problem (CBPP-1) asks for a packing of m type of 1D items with a length w_i and a quantity h_i, $i \in I := \{1, \ldots, m\}$ into 1D bins with a length $W \in \mathbb{Z}_+$ in a fixed order which uses the minimal number of the bins satisfying the following restrictions:

1. Each item of one type is packed at most once into a bin.

2. Each item of one type is packed contiguously into a sequence of the bins.

All data is integer, i.e., $w_i \in \{1, \ldots, W\}$, $h_i \in \mathbb{Z}_+$ for $i \in I$.

Let us reinterpret CBPP-1. Suppose we have a computational resource with a capacity of W units; m jobs which consume w_i units of the resource for h_i units of time. We require that the jobs have to be calculated on the computational resource without interruption of the job execution. Herewith with obtain the *non-preemptive scheduling* problem [CMM03]. This problem is equivalent to CBPP-1.

2.1.1 Modeling and notations

Let us firstly consider the problem of 1D bin packing. A feasible packing of the items into a bin is represented by the following binary vector:

$$a := (a_1, \dots, a_m)^T \in \{0,1\}^m, \tag{2.1}$$

where $a_i = 1$ indicates the case when the item i is packed into the bin. Thus we do not consider a concrete position of each item i in a bin packing as we did in [MSB13a] for the 1D problems. To keep track of different packing variants of a bin we introduce an index j and a set J which describe all packings $a^j := (a_1^j, \dots, a_m^j)^T \in \{0,1\}^m$, $j \in J$. Note a vector a^j represents a feasible packing if:

$$\sum_{i \in I} w_i a_i^j \leq W. \tag{2.2}$$

For each packing variant a^j let us introduce a non-negative integer variable x_j which indicates how often a^j is applied. Herewith we obtain according to the [GG61, GG63] the following Gilmore-Gomory model of the 1D bin packing problem:

$$\sum_{j \in J} x_j \to \min; \quad \text{s.t.} \tag{2.3}$$

$$\sum_{j \in J} a_i^j x_j = h_i, \quad i \in I; \tag{2.4}$$

$$x_j \in \mathbb{Z}_+, \quad j \in J. \tag{2.5}$$

Through the constraints (2.4) we ensure that each type i of the items is packed exactly h_i times.

The linear programming (LP) relaxation of the problem (2.3)-(2.5) is defined as follows:

$$\sum_{j \in J} y_j \to \min; \quad \text{s.t.} \tag{2.6}$$

$$\sum_{j \in J} a_i^j y_j = h_i, \quad i \in I; \tag{2.7}$$

$$y_j \geq 0, \quad j \in J. \tag{2.8}$$

Since $|J|$ can be very large we use the Dantzig-Wolfe decomposition and solve the problem by the revised simplex method [DW60].

In [GG63] the solution method for (2.6)-(2.8) is referred as to a column-generation technique. The problem is split into two: the master and the slave problems. The master problem is the original problem (2.6)-(2.8) with only a subset of the variables

which are currently under consideration. The slave problem is the following 0-1 integer program also known as a binary knapsack problem:

$$\sum_{i\in I} d_i a_i \to \max; \quad \text{s.t.} \tag{2.9}$$

$$\sum_{i\in I} w_i a_i \le W; \tag{2.10}$$

$$a_i \in \{0,1\}, \quad i \in I, \tag{2.11}$$

where d_i is the dual simplex multiplier for the i-th constraint of (2.7) in the master problem. The slave problem (2.9)-(2.11) is created to identify a new variable which will be added into the master problem. The value of the objective function (2.9) is the reduced cost of the new variable with respect to the current dual variables, and the feasibility constraint (2.2).

The solution process for the (2.6)-(2.8) problem is as follows. The master problem is solved from a feasible solution. Herewith we obtain values of the dual simplex multipliers d_i for each constraint (2.7) in the master problem. Then we solve the slave problem with the newly obtained d_i and get a solution a^*. If $1-\sum_{i\in I} d_i a_i^* \le -\epsilon$ with a sufficiently small constant $\epsilon \ge 0$ then a variable with a negative reduced cost has been found. This variable is then added to the master problem and the master problem is resolved. After the resolution new values of the dual simplex multipliers are generated and the process is repeated until no variable of a negative reduced cost is found. If the slave problem returns a solution with a nonnegative reduced cost we conclude that the solution of the master problem is optimal.

Let $\bar{J}_x := \{j \in J : \ x_j > 0\}$ be the set of column indexes which corresponds to the solution of the integer problem (2.3)-(2.5), i.e., $\{a^j : \ j \in \bar{J}_x\}$. Analogously we define $\bar{J}_y := \{j \in J : \ y_j > \epsilon\}$ as a set of column indexes which corresponds to the solution of the LP relaxation (2.6)-(2.8), i.e., $\{a^j : \ j \in \bar{J}_y\}$.

Now let us go back to CBPP-1. For $\{a^j : \ j \in \bar{J}_x\}$ with $\{x_j : \ j \in \bar{J}_x\}$ to be a feasible solution of CBPP-1, both heterogeneity and contiguity constraints have to be fulfilled. The first restriction is fulfilled because of the definition (2.1) of a packing. For the contiguity restriction to be fulfilled, the column set $\{a^j : \ j \in \bar{J}_x\}$ must have a consecutive 1's property (C1P).

Definition 2.1. *A binary column set (or matrix) is said to have the C1P (for rows) if there exists a permutation of its columns that places the 1's consecutively in every row.*

Let us now consider the LP relaxation (2.9)-(2.11), its solution $\{a^j : \ j \in \bar{J}_y\}$, $\{x_j : \ j \in \bar{J}_y\}$ and the following fact.

Proposition 2.1. *A matrix with the C1P is totally unimodular.*

Proof. See the proof of Corollary 2.10 in [NW88], p.544. Note in the book the matrices with the C1P are called interval and this property appears in the columns. □

Thus, for the columns $\{a^j : \ j \in \bar{J}_y\}$ in the solution of the LP relaxation (2.9)-(2.11) with the C1P the corresponding $\{x_j : \ j \in \bar{J}_y\}$ is integral. So we restrict the consideration to the relaxation problem and the column sets with the C1P. Some characterizations of such column sets are given in Section 2.1.2.

2.1.2 Column sets and bipartite graphs

Let $C := \{c^j \in \{0,1\}^m : \ j \in J_C\}$ be a column set for an $J_C := \{1, \ldots, n\}$. The bipartite graph $B_C := (V_1^C, V_2^C, E_C)$ associated to the column set C is defined by $V_1^C := I$, $V_2^C := J_C$, and $E_C := \{(i,j) \in V_1^C \times V_2^C : \ c_i^j = 1\}$, see Fig. 2.1. A bipartite graph is convex if its half adjacency matrix[1] has the C1P. For the column set on Fig. 2.1a the corresponding bipartite graph, see Fig. 2.1b, is non-convex.

Let $(p \to q)$ for $p, q \in V_2^C$ be the set of vertexes from $V_1^C \cup V_2^C$, which are in a path from p to q. Hence, if there does not exist a path from p to q with $p, q \in V_2^C$ in B_C then $(p \to q) = \emptyset$.

Let $N(k) := \{i \in V_1^C : \ c_i^k = 1\}$ be the closed neighborhood of a node $k \in V_2^C$, see Fig. 2.1.

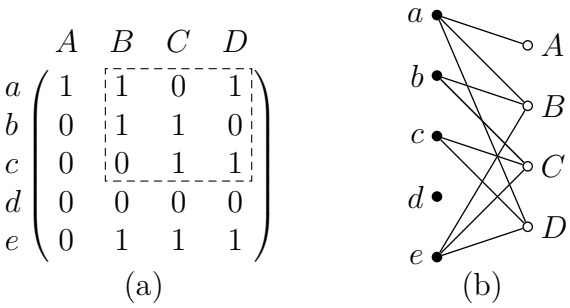


Figure 2.1: Column set without the C1P: (a) – A column set; (b) – The corresponding bipartite graph. The property is broken in the subcolumns of the columns B, C, D which are marked by the dashed box. Thus, the corresponding bipartite graph is non-convex. The closed neighborhoods of each vertex is $N(A) := \{a\}$, $N(B) := \{a, b, e\}$, $N(C) := \{b, c, e\}$, and $N(D) := \{a, c, e\}$. The bipartite graph has a single asteroidal triple, namely (B, C, D) because only for these vertexes there exist the paths $(B \to C)^D := \{B, b, C\}$, $(C \to D)^B := \{C, c, D\}$, and $(B \to D)^C := \{B, c, D\}$.

A characterization of the graphs corresponding to the column sets with the C1P was given by A. Tucker. The characterization in given in terms of so-called *asteroidal triples* (A-triples) [Tuc72] defined as follows.

Definition 2.2. *Let $B_C := (V_1^C, V_2^C, E_C)$ be a bipartite graph. Vertexes $p, q, r \in V_2^C$ with $p \neq q$, $q \neq r$, and $p \neq r$ form an A-triple, if $(p \to q) \neq \emptyset$, $(p \to r) \neq \emptyset$, $(q \to r) \neq \emptyset$, and:*

$$(p \to q) \cap N(r) = \emptyset \ \wedge \ (p \to r) \cap N(q) = \emptyset \ \wedge \ (q \to r) \cap N(p) = \emptyset.$$

On Fig. 2.1b there is only one A-triple, namely (B, C, D). Note in [Tuc72] a common graph is considered not the bipartite one. We deal with bipartite graphs, so we define A-triples only for them.

Let $(p \to q)^r$ be a path from p to q without the closed neighborhood of r: $(p \to q)^r \cap N(r) = \emptyset$.

The following theorem characterizes convex bipartite graphs.

[1]We deal with undirected bipartite graphs, therefore only one part of the adjacency matrix is enough to define the corresponding graph.

Theorem 2.2 ([Tuc72, Theorem 6, p.156]). *Let $B_C := (V_1^C, V_2^C, E_C)$ be the bipartite graph associated to a column set C. The set C has the C1P if and only if V_2^C contains no A-triple of B_C.*

The roles of V_1 and V_2 are interchanged here compared to [Tuc72]. A direct consequence of Theorem 2.2 are the following corollaries.

Corollary 2.3. *A column set $C := \{c^j \in \{0,1\}^m : \ j \in J_C\}$ has the C1P if and only if the associated bipartite graph $B_C := (V_1^C, V_2^C, E_C)$ with $V_1^C := I$, $V_2^C := J_C$ contains no A-triple.*

Corollary 2.4. *A column set $C := \{c^j \in \{0,1\}^m : \ j \in J_C\}$ does not have the C1P if and only if the associated bipartite graph $B_C := (V_1^C, V_2^C, E_C)$ with $V_1^C := I$, $V_2^C := J_C$ contains at least one A-triple.*

Another Tucker's characterization of graphs having no A-triple is given in the following theorem.

Theorem 2.5 ([Tuc72, Theorem 7, p.157]). *In a bipartite graph $B_C := (V_1^C, V_2^C, E_C)$ the vertex set V_2^C contains no A-triple if and only if B_C contains none of the graphs G_k^1, G_k^2, G_k^3 (with $k \geq 1$), G^4, and G^5 as shown in Fig. 2.2.*

A direct consequence of Theorem 2.5 is the following corollary.

Corollary 2.6. *A bipartite graph $B_C := (V_1^C, V_2^C, E_C)$ has at least one A-triple if and only if B_C contains one or more graphs G_k^1, G_k^2, G_k^3, G^4, and G^5 as a subgraph.*

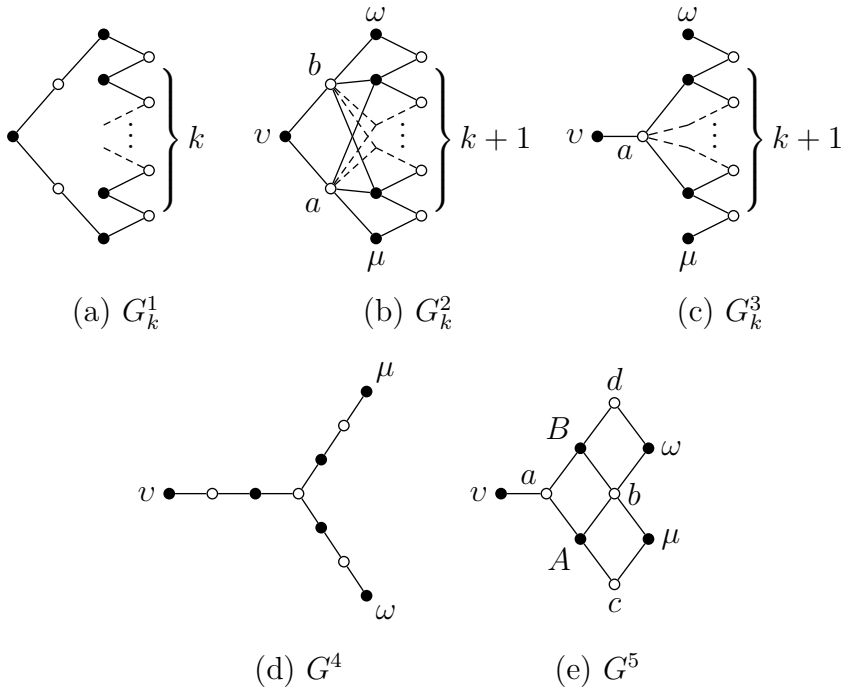
2.1.3 Overview of solution methods

A position-indexed formulation for CBPP-1 is considered in [HNS08] where a branch-and-cut algorithm based on some facet-defining inequalities is proposed. However, the proposed model has different drawbacks, the most crucial of which is the pseudo-polynomial number of position-indexed variables.

In [MMBS11], CBPP-1 is handled by branch-and-bound methods with the lower bounds based on the LP relaxation of the Dantzig-Wolfe decomposed of the 1D bin packing problem which is solved by the column generation method. The approach with the decomposed model is similar to those which are considered in this paper. The main difference is that here we propose branch-and-price algorithms in contrast to the combinatorial enumeration algorithms in [MMBS11]. Here we also tighten the column generation.

2.1.4 Our contribution

In order to solve CBPP-1 we use a Gilmore-Gomory model for the 1D bin packing problem which was discussed in Section 2.1. In this section we use the know fact that if a column set has the C1P then the corresponding matrix is total unimodular and thus all the corners of the polyhedron of its LP relaxation are integer for the integer



(a) G_k^1 (b) G_k^2 (c) G_k^3

(d) G^4 (e) G^5

Figure 2.2: Forbidden subgraphs. The vertex set V_2^C of a bipartite graph $B_C := (V_1^C, V_2^C, E_C)$ contains an A-triple if and only if B_C contains one of the displayed graphs as an induced subgraph where black vertexes correspond to vertexes in V_2^C. Numbers k and $k+1$ refer to the numbers of white vertexes in the right-hand parts of the first three graphs. In the case of the graph G_k^1, any three different black vertexes build an A-triple. In other cases there is only a single A-triple, namely (υ, μ, ω).

input data. For the construction of such column sets, in Section 2.2 we develop branch-and-price algorithms with two strategies: column-, and subcolumn-based enumerations. In Sections 2.3, 2.4 we prove propositions about some characteristics of a specific or an arbitrary column which breaks the C1P of a given column set. For each of these characteristics we develop an algorithm the output data of which is used in the slave problem of the column generation thus tightening the bound. The final part of the paper reports numerical results and conclusion.

2.2 The branch-and-price algorithms

Here we describe the overall algorithm of finding an optimum of CBPP-1. The algorithm is a hybrid of the branch-and-bound and the column generation methods, i.e., throughout the branching tree the column generation is applied for solving the LP relaxation. The idea is the same as in branch-and-bound, we calculate lower bounds for each node and use it as a pruning criteria.

In practice, for the branching-and-price, we store the columns which we generate throughout the optimization process in a column pool. Processing from one node to the descendants is done by adding a box-constraint for the corresponding variable. So storing the column indexes for each node is enough to reconstruct the formulation.

As we have shown in Section 2.1.1, the solution of the LP relaxation (2.9)-(2.11) with the C1P is integral. Thus, our goal now is to enumerate column sets with the C1P.

The idea of branchings consist in an enumeration of feasible 1D partial packings having the C1P. On each step all columns with indexes $J^+ \subseteq J$, i.e., $\{a^j : \ j \in J^+\}$, are fixed to appear in the packing while all columns with indexes $J^- \subseteq J$ must not appear, i.e., $\{a^j : \ j \in J^-\}$. The column set $\{a^j : \ j \in J^+\}$ is constructed such that the C1P arise.

Let V be the set of nodes of the branching tree $T := (V, E)$, which are connected by the edges from the set E. Hereby the sets $J^+(u)$ and $J^-(u)$ are associated with each node $u \in V$.

Given a node $u \in V$ of the branching tree and having the sets $J^+(u)$ and $J^-(u)$ in mind, the master problem of the column-generation method becomes the following form:

$$\mathrm{LP}(u): \ \phi(u) = \sum_{j \in J} y_j \to \min; \quad \text{s.t.} \tag{2.12}$$

$$\sum_{j \in J} a_i^j y_j = h_i, \quad i \in I; \tag{2.13}$$

$$y_j \geq 1, \quad j \in J^+(u); \tag{2.14}$$

$$y_j = 0, \quad j \in J^-(u); \tag{2.15}$$

$$y_j \geq 0, \quad j \in (J \setminus J^+(u)) \setminus J^-(u), \tag{2.16}$$

and the slave problem:

$$\mathrm{CG}(u): \ \sum_{i \in I} d_i a_i \to \max; \quad \text{s.t.} \tag{2.17}$$

$$\sum_{i \in I} w_i a_i \leq W; \tag{2.18}$$

$$\sum_{i \in I} [a_i(1 - a_i^j) + a_i^j(1 - a_i)] > 0, \quad j \in J^-(u); \tag{2.19}$$

$$a_i \in \{0, 1\}, \quad i \in I. \tag{2.20}$$

where the constraints (2.19) exclude the solutions from $J^-(u)$ from the feasible set of solutions.

Let $\bar{J}(u) := \{j \in J : \ y_j > \epsilon\}$ be the set of column indexes which are in the solution of LP(u). Some of the columns $\{a^j : \ j \in \bar{J}(u)\}$ build together with the columns with indexes from $J^+(u)$ a column set having the C1P. Let $\bar{J}^+(u) \subseteq \bar{J}(u)$ be such that $\forall j \in \bar{J}^+(u)$, $J^+(u) \cup \{a^j\}$ has the C1P, and $\bar{J}^-(u) := \bar{J}(u) \setminus \bar{J}^+(u)$. For testing whether a column set has the C1P we use the PQ-tree algorithm [BL76].

In the following we introduce two branch-and-price algorithms which are distinguished by the branching strategies.

2.2.1 A column-based branching strategy

Here we introduce a column-based branching strategy. The main idea is to construct the solution having the C1P from the columns of the LP relaxation.

Having the $J^+(u)$ and $J^-(u)$ sets we solve the LP relaxation and test the columns whether they build with the columns $\{a^j : j \in J^+(u)\}$ a column set with the C1P. If there exists one then we branch on this column. If it does not exist then we tighten the formulation of the LP relaxation and resolve until a column to branch is found, the node is pruned considering the bound or it is proven that no column to branch exists.

Algorithm 2.1 (Branch-and-price-cols)**.** *A branch-and-price algorithm with the column-based branching strategy.*
Input data: W, m, $w = (w_1, \ldots, w_m)$, $h = (h_1, \ldots, h_m)$.
Output data: optimal solution (A, Y) and its value H.

(*1*) Initialization*:*
- *node u_0, $J^+(u_0) = J^-(u_0) := \emptyset$.*
- *incumbent $A := \emptyset$, $Y := \emptyset$, $H := \infty$.*
- *node list $N := \{u_0\}$.*

(*2*) Node*: If $N = \emptyset$, go to* Exit*. Else choose $u \in N$, set $N := N \backslash \{u\}$ and go to* Restore.

(*3*) Restore *the sets $J^+(u)$ and $J^-(u)$.*

(*4*) LP relaxation*: Solve* LP(u) *considering $J^+(u)$ and $J^-(u)$ and obtain the solution and hence the set $\bar{J}(u)$. For all $j \in \bar{J}(u) \setminus J^+(u)$ test by the PQ-tree algorithm, if $J^+(u) \cup \{a^j\}$ has the C1P. If yes then set $\bar{J}^+(u) := \bar{J}^+(u) \cup \{j\}$, else set $\bar{J}^-(u) := \bar{J}^-(u) \cup \{j\}$.*

(*5*) Prune*: If $\phi(u) \geq H$, go to* Node*. If the solution $\{y_j : j \in \bar{J}(u)\}$ is integral then set $H := \phi(u)$, update the incumbent (A, Y) with $A := \{a^j : j \in \bar{J}(u)\}$, $Y := \{y_j : j \in \bar{J}(u)\}$ and go to* Node*, else go to* Branching.

(*6*) Branching*: Having $J^+(u)$ and $\bar{J}^+(u)$ do:*

(*6.1*) *If $\bar{J}^+(u) \setminus J^+(u) = \emptyset$ then create one node v with:*
- *$J^+(v) := J^+(u)$.*
- *$J^-(v) := J^-(u) \cup \bar{J}^-(u)$.*

Set $N := N \cup \{v\}$. Go to Node*.*

(*6.2*) *If $\bar{J}^+(u) \setminus J^+(u) \neq \emptyset$ then set $j^* := \operatorname{argmax}\{y_j : j \in \bar{J}^+(u) \setminus J^+(u)\}$, and create node v_1:*
- *$J^+(v_1) := J^+(u) \cup \{j^*\}$,*
- *$J^-(v_1) := J^-(u) \cup \bar{J}^-(u)$,*

and create node v_2:
- *$J^+(v_2) := J^+(u)$,*
- *$J^-(v_2) := J^-(u) \cup \bar{J}^-(u) \cup \{j^*\}$.*

Set $N := N \cup \{v_1, v_2\}$. Go to Node*.*

(*7*) Exit*: Return the incumbent (A, Y) and its value H.*

As we can see the main drawback of the described strategy is that we may create a lot of descendants while $\bar{J}^+(u) \setminus J^+(u) = \emptyset$, see step 6.1 in Algorithm 2.1. For

eliminating this drawback we tighten the column generation by forbidding not only the columns but subcolumns.

Let us consider the column set on Fig. 2.1 where four columns $A, \ldots, D$, and five rows $a, \ldots, e$ exist. Columns A, B, C together have the C1P. If we join them with the column D then the obtained column set will not have the C1P. The property is broken not in the whole matrix, but only in the part marked by the dashed box. Note if there is a such combination $a = 1$, $b = 0$, $c = 1$ of rows in a column D, the C1P would not arise despite the values of d and e. So, if we now consider only columns were there is no such a combination of rows, we reduce the columns which break a priori the C1P.

Let a subcolumn d^k be given by an index set $I_k \subset I$ and coefficients d_i^k, $i \in I_k$. For a given node $u \in V$ of the branching tree we consider a set of subcolumns $D(u) := \{d^k : \ k \in K(u)\}$ which breaks the C1P of the column set $\{a^j : \ j \in J^+(u)\}$. Hereby, if for some $j \in J$ and $k \in K(u)$: $a_i^j = d_i^k$, $\forall i \in I_k$ then $y_j = 0$ has to follow which is realized in the master problem of the column generation by:

$$\sum_{i \in I_k} |a_i^j - d_i^k| = 0 \ \Rightarrow \ y_j = 0, \quad j \in (J \backslash J^+(u)) \backslash J^-(u), \ k \in K(u).$$

The above constraint is nonlinear. For its tackling we remove it from the master problem and add it to the slave problem CG(u) of the column generation in the following form:

$$\sum_{i \in I_k} [a_i(1 - d_i^k) + d_i^k(1 - a_i)] > 0, \quad k \in K(u), \tag{2.21}$$

which is herewith tightened.

For construction of subcolumns for a given set of columns having the C1P we propose two approaches. Suppose for a given $u \in V$ we have $J^+(u)$ and the column set $\{a^j : \ j \in J^+(u)\}$. The first approach uses the latter columns having the C1P and construct the subcolumns which can potentially break the C1P. This approach is considered in Section 2.4.

Based on u, $J^+(u)$ and due to Algorithm 2.1, we solve LP(u) and obtain the set $\bar{J}(u)$. Afterward we apply the PQ-tree algorithm for each $j \in \bar{J}(u) \setminus J^+(u)$ and define $\bar{J}^-(u)$. Now from the columns $\{a^j : \ j \in \bar{J}^-(u)\}$ we find the subcolumns which break the C1P. This is the second approach which is considered in Section 2.3.

2.2.2 A subcolumn-based branching strategy

Here we introduce a subcolumn-based branching strategy. This branching strategy was firstly discussed in [Bel10]. The main idea of the branching strategy is to branch on the subcolumns which break the C1P of the current basis solution of the LP relaxation.

Having a basis solution of the LP relaxation we test it for the C1P by the PQ-tree algorithm [BL76]. If it does not have the C1P then the basis solution contained one or more submatrices of the types $M_I, \ldots, M_V$ in [Tuc72], Fig. 3, p.161. Note the corresponding matrix will contain one or more of the graphs from Fig. 2.2 as a subgraph. The submatrices of the types $M_I, \ldots, M_V$ can be found by the algorithms described in [Dom09], Chapter 3, p.63–78.

Algorithm 2.2 (Branch-and-price-subcols)**.** *A branch-and-price algorithm with the subcolumn-based branching strategy.*
Input data: W, m, $w = (w_1, \ldots, w_m)$, $h = (h_1, \ldots, h_m)$.
Output data: optimal solution (A, Y) and its value H.

(*1*) Initialization*:*

- *node u_0, $D(u_0) := \emptyset$.*
- *incumbent $A := \emptyset$, $Y := \emptyset$, $H := \infty$.*
- *node list $N := \{u_0\}$.*

(*2*) Node*: If $N = \emptyset$, go to* Exit*. Else choose $u \in N$, set $N := N \backslash \{u\}$ and go to* Restore*.*

(*3*) Restore *the set $D(u)$.*

(*4*) LP relaxation*: Let $J(u)$ be the set of columns from the column pool and let $J^d(u) := \{j \in J(u) : \ a_i^j(1 - d_i^k) + d_i^k(1 - a_i^j) = 0, \ d^k \in D(u)\}$ be the set of indexes of columns from $J(u)$ which contain a forbidden subcolumn from $D(u)$. Solve the following LP relaxation considering $D(u)$ and $J^d(u)$:*

$$\text{LP'}(u): \ \phi(u) = \sum_{j \in J} y_j \to \min; \quad \text{s.t.} \tag{2.22}$$

$$\sum_{j \in J} a_i^j y_j = h_i, \quad i \in I; \tag{2.23}$$

$$y_j = 0, \quad j \in J^d(u); \tag{2.24}$$

$$y_j \geq 0, \quad j \in J \setminus J^d(u), \tag{2.25}$$

and obtain the solution and hence the set $\bar{J}(u)$. By the PQ-tree algorithm test if $\{a^j : \ j \in \bar{J}(u)\}$ has the C1P. If yes then goto Prune*, else find a submatrix $F(u) := \{f_1, \ldots, f_{\sigma(u)}\}$ of the types $M_I, \ldots, M_V$ in $\{a^j : \ j \in \bar{J}(u)\}$.*

(*5*) Prune*: If $\phi(u) \geq H$, go to* Node*. If the solution $\{y_j : \ j \in \bar{J}(u)\}$ is integral then set $H := \phi(u)$, update the incumbent (A, Y) with $A := \{a^j : \ j \in \bar{J}(u)\}$, $Y := \{y_j : \ j \in \bar{J}(u)\}$ and go to* Node*, else go to* Branching*.*

(*6*) Branching*: Create nodes v_k with $k = 1, \ldots, \sigma(u)$: $D(v_i) := D(u) \cup \{f_k\}$, set $N := N \cup \{v_k\}$. Go to* Node*.*

(*7*) Exit*: Return the incumbent (A, Y) and its value H.*

2.3 Subcolumns breaking the C1P

Here we consider an approach of finding the subcolumns of a given column which breaks the C1P of a given set of columns having the C1P.

Now going back to a node $u \in V$ of the branching tree T we can conclude using Corollary 2.3 that since $\{a^j : \ j \in J^+(u)\}$ has the C1P then the corresponding bipartite graph has no A-triple. The following statement is true.

Statement 2.7. *Let $u \in V$ be a node of the branching tree T and let a^γ with $\gamma \in J \setminus J^+(u)$. The column set $C := \{a^\gamma\} \cup \{a^j : j \in J^+(u)\}$ does not have the C1P if and only if the associated bipartite graph $B_C := (I, \{1, \dots, |C|\}, E_C)$ contains at least one A-triple.*

Remark 2.1. *An A-triple which exists in B_C may contain the vertex γ and may not contain it as well.*

Hereby, if we join the columns $\{a^j : j \in J^+(u)\}$ with a column a^j with $j \in \bar{J}^-(u)$, the resulting set will not have the C1P and according to Statement 2.7 the corresponding bipartite graph will have at least one A-triple.

Let $J_C \subseteq J$ and $C := \{a^j : j \in J_C\}$ be a set of columns and let $C(I_r, I_c) := \{a^j_i : i \in I_r,\ j \in I_c\}$ be the set of the columns from $I_c \subset J_C$ and the rows from $I_r \subset I$. The following lemma is true.

Lemma 2.8. *If a column set C has the C1P then the set $C(I_r, I_c)$ will also have the C1P for any $I_r \subset I$ and $I_c \subset J_C$.*

Proof. Let us prove the lemma for elements $k \in I \setminus I_r$ and $d \in J_C \setminus I_c$. Let C be a set having the C1P. Due to Theorem 2.2, the corresponding bipartite graph B_C has no A-triple. Let us consider the set $C(I\setminus\{k\}, J_C\setminus\{d\})$ and assume that it does not have the C1P. Due to Corollary 2.4, the corresponding bipartite graph $B_{C(I\setminus\{k\}, J_C\setminus\{d\})}$ has at least one A-triple, e.g., (υ, μ, ω) with $\upsilon \neq \mu$, $\mu \neq \omega$, and $\upsilon \neq \omega$. It means that $\{k, d\} \cap [(\upsilon \to \mu)^\omega \cup (\upsilon \to \omega)^\mu \cup (\mu \to \omega)^\upsilon] = \emptyset$, while $(\upsilon \to \mu)^\omega \neq \emptyset$, $(\upsilon \to \omega)^\mu \neq \emptyset$, and $(\mu \to \omega)^\upsilon \neq \emptyset$. The latter follows that the paths $(\upsilon \to \mu)^\omega, (\upsilon \to \omega)^\mu, (\mu \to \omega)^\upsilon$ must exist also in the graph B_C. Hence, (υ, μ, ω) is an A-triple in B_C and due to Corollary 2.4 the set C does not have the C1P which contradicts the condition that C has the C1P. Hereby we set $I := I\setminus\{k\}$, $J_C := J_C\setminus\{d\}$ and repeat the above procedure until $I = I_r$ and $J_C = I_c$. □

Let d^* be a subcolumn which breaks the C1P of $\{a^j : j \in J^+(u)\}$ and be given by an index set $I_* \subset I$ and coefficients d^*_i with $i \in I_*$. If $C(I_*, J_C) \cup \{d^*\}$ does not have the C1P then d^* is called ineligible. An ineligible subcolumn is called minimal if $\nexists i^* \in I_*$:

$$\bar{d^*} = \begin{cases} 1 - d_i, & i = i^*, \\ d_i, & i \in I_*\setminus\{i^*\}, \end{cases}$$

so that $C(I_*, J_C) \cup \{\bar{d^*}\}$ does not have the C1P.

Theorem 2.9. *If d^* is a minimal ineligible subcolumn then $1 \leq \sum_{i \in I_*} d^*_i \leq 3$ is true.*

Proof. Since $C(I_*, J_C) \cup \{d^*\}$ does not have the C1P then due to Corollary 2.4 the corresponding bipartite graph contains at least one A-triple. Therefore, due to Corollary 2.6 the bipartite graph contains one or more graphs G^1_k, G^2_k, G^3_k, G^4, and G^5 as a subgraph. Since d^* corresponds to a vertex of the set V_2, the number of 1's in d^* is a degree of the corresponding vertex which can be any of V_2. Let us calculate minimal and maximal degrees of vertexes from V_2 of G^1_k, G^2_k, G^3_k, G^4, and G^5, respectively.

Let us consider the graph G_k^1. If we take any $\upsilon, \mu, \omega \in V_2^{G_k^1}$ with $\upsilon \neq \mu$, $\upsilon \neq \omega$, and $\mu \neq \omega$ then there would exist nonempty paths $(\upsilon \to \mu)^\omega$, $(\upsilon \to \omega)^\mu$, and $(\mu \to \omega)^\upsilon$ which means that (υ, μ, ω) is an A-triple. Note if we delete one of the edges which is incident to a vertex from $V_2^{G_k^1}$ in the graph then one of the three paths would not exist and hence no A-triple. So, the minimal and the maximal vertex degree for G_k^1 are equal to 2.

Let us consider the graph G_k^2. It has such a construction, so that there exists only one A-triple, namely (υ, μ, ω). Moreover the path $(\mu \to \omega)^\upsilon$ must not contain the vertexes a and b due to the definition of A-triples. So the single path which satisfies this condition goes through the white and the black vertexes except υ, a, and b. Let d^* correspond to a black vertex except υ, μ, and ω. The above described path contains necessarily this vertex. Every such vertex has the degree equal to 4. Actually, if we delete two edges which make this vertex incident to a and b then the path $(\mu \to \omega)^\upsilon$ would still exist and the A-triple as well. It means that these two edges are excessive and are not necessary to build the A-triple. It is also obvious that both of the two edges which outcome from the vertexes υ, μ, and ω are necessary to build the A-triple. So, the minimal and the maximal vertex degree for G_k^2 are equal to 2 in order to build the A-triple (υ, μ, ω).

Let us consider the graph G_k^3. The A-triple is constructed only by the vertexes υ, μ, and ω. We have the same path between μ and ω as in the case of G_k^2, the path goes only through the white and the black vertexes except υ, and a. Let d^* correspond to a black vertex except υ, μ, and ω. The edge which makes the vertex incident to a is actually excessive because the path $(\mu \to \omega)^\upsilon$ does not contain the vertex a. But there must be at least one black vertex which is incident to a for the existence of the paths $(\upsilon \to \mu)^\omega$ and $(\upsilon \to \omega)^\mu$. It is also obvious that the edge outcomming from υ, μ, and ω is necessary to build the A-triple. So, the vertex degree for G_k^3 lies in the interval from 1 to 3 in order to build the A-triple (υ, μ, ω).

Let us consider the graph G^4. It obvious that the deletion of an edge in the graph G^4 will break the coherence of the graph G^4 which leads to the graph with no A-triple. So, the vertex degree for G^4 lies in the interval from 1 to 2 in order to build the A-triple (υ, μ, ω).

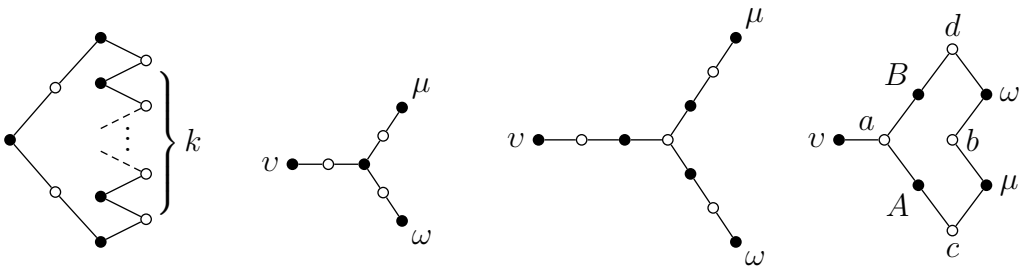
Let us consider the graph G^5. The vertexes υ, μ, and ω create the single A-triple, so that the paths $(\upsilon \to \mu)^\omega$, $(\upsilon \to \omega)^\mu$, and $(\mu \to \omega)^\upsilon$ are not empty. Let d^* correspond to μ or ω. In this case the edges (μ, b) and (ω, b) are excessive because the above described paths are not empty without these edges, i.e., $(\upsilon \to \mu)^\omega = \{\upsilon, A, c, \mu\}$, $(\upsilon \to \omega)^\mu = \{\upsilon, B, d, \omega\}$, and $(\mu \to \omega)^\upsilon = \{\mu, c, A, b, B, d, \omega\}$, so the one outcomming egde (μ, c) and (ω, d) for μ and ω is sufficient. Let d^* correspond to A or B. Then the edges (b, B), and (b, A) are excessive for the same reason, i.e., $(\upsilon \to \mu)^\omega$ and $(\upsilon \to \omega)^\mu$ are the same, but $(\mu \to \omega)^\upsilon = \{\mu, b, \omega\}$. It is also obvious that there must exist the edge (υ, a). Therefore, the vertex degree for G^5 lies in the interval from 1 to 2 in order to build the A-triple (υ, μ, ω).

The above implies the validity of $1 \leq \sum_{i \in I_*} d_i^* \leq 3$. □

Lemma 2.10. *If $\sum_{i \in I_*} d_i^* = 1$ then the vertex corresponding to d^* is contained in one of the A-triples.*

Proof. Since $C(I_*, J_C) \cup \{d^*\}$ does not have the C1P then due to Corollary 2.4 the corresponding bipartite graph contains at least one A-triple. Therefore, due to Corollary 2.6 the bipartite graph contains one or more graphs G_k^1, G_k^2, G_k^3, G^4, and G^5 as a subgraph. Since $\sum_{i \in I_*} d^* = 1$ is minimal then the corresponding vertex is terminal. In the above described graph all terminal vertexes correspond to the vertexes of the A-triple. □

Theorem 2.11. *If a subcolumn d^* is minimal then the corresponding bipartite graph contain one of the graphs $\bar{G}_k^1$, $\bar{G}^2$, $\bar{G}^3$, and $\bar{G}^4$.*



(a) $\bar{G}_k^1$ (b) $\bar{G}^2$ (c) $\bar{G}^3$ (d) $\bar{G}^4$

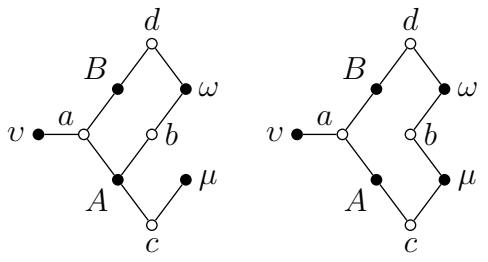
Figure 2.3: Forbidden subgraphs $\bar{G}_k^1, \bar{G}^2, \bar{G}^4, \bar{G}^4$.

Proof. Since $C(I_*, J_C) \cup \{d^*\}$ does not have the C1P then due to Corollary 2.4 the corresponding bipartite graph contains at least one A-triple. Therefore, due to Corollary 2.6 the bipartite graph contains one or more graphs G_k^1, G_k^2, G_k^3, G^4 and G^5 as a subgraph.

Let us consider the situation before we add the vertex corresponding to d^* to the bipartite graph. In the case of G_k^1 the graph lacks one of the black vertexes. It is obvious that connecting of two white vertexes with the vertex degree equal to 1 with the new vertex will create an A-triple. Moreover, this is the single connection with the minimal number of edges in order to get an A-triple. Other connections will contain the mentioned edges. The same is with the graph G^4, see Fig. 2.3c.

Let us consider the graphs G_k^2 and G_k^3. There exist only a single A-triple (υ, μ, ω). It means that there exist the paths $(\mu \to \omega)^{\upsilon}$, $(\upsilon \to \mu)^{\omega}$, and $(\upsilon \to \omega)^{\mu}$. The second and the third paths do not contain the edges (b, ω) and (a, μ), and the first path goes only through the right-hand white vertexes. So, the edges (b, ω) and (a, μ) can be removed. Hence the graphs G_k^2 and G_k^3 are equivalent in the sense of the mentioned paths. So, let us consider the graph G_k^3 before we add the vertex corresponding to d^*. The cases where the graph lacks υ, μ, and ω are obvious. In other case it is enough to add the edges starting from the white vertexes with the degree equal to 1, see the graph $\bar{G}^2$ in Fig. 2.3b. All other possible constructions building an A-triple will contain these edges.

Let us consider the graph G^5. There exists a single A-triple (υ, μ, ω) with the two different existing paths, see Fig. 2.4. It is obvious that the path (a, B, d, ω) in the graph $G^{5'}$ is excessive, so that the graph is equivalent to the graph $\bar{G}^2$. Let us consider



(a) $G^{5'}$ (b) $G^{5''}$

Figure 2.4: Forbidden subgraph G^5: different paths.

the graph $G^{5''}$ and the situation before we add the vertex corresponding to d^*. In the case where v do not exist, any three vertexes of $\{A, B, \mu, \omega\}$ can build an A-triple. That contradicts the assumption that before we add a vertex to the graph it contains no A-triple. So, v should be in the graph. In the case of μ or ω there exist only one possibility to build an A-triple, namely to add the edges (μ, c), (μ, b) or (ω, d), (ω, b), respectively. In the case of the vertex A, there exist two possibilities. The first one is to add the edge (A, d), so that an A-triple as in $\bar{G}^1_k$ arises. The second one is to add the edge (A, a). The same is with the vertex B.

□

Lemma 2.12. *Let d^* be a minimal ineligible subcolumn. If there exists an A-triple containing the vertex corresponding to d^* then $1 \leq \sum_{i \in I_*} d^*_i \leq 2$ is true.*

For the proof refer to the next lemma.

Lemma 2.13. *If d^* is a minimal ineligible subcolumn then $1 \leq \sum_{i \in I_*} d^*_i \leq 3$ is true.*

Proof. Before we add a vertex to the graph the latter does not contain any A-triples. Since we add the vertex corresponding to the minimal d^* then due to Theorem 2.11 there arise one or more graphs $\bar{G}^1_k$, $\bar{G}^2$, $\bar{G}^3$, and $\bar{G}^4$. Both Lemmas 2.12 2.13 are easy to prove considering the above mentioned graphs. Correctness of Lemma 2.12 follows from the fact that the maximal degree of the black vertexes which are contained in an A-triple is equal to 2. Correctness of Lemma 2.13 follows from the fact that the maximal degree of a black vertex in all graphs is equal to 3, see Fig. 2.3b. □

Let us go back to the column a^γ with $\gamma \in \bar{J}^-(u)$ and the subcolumn d^k with a minimal size. As noted in Remark 2.1 there exist two possibilities of how an A-triple arise due to Statement 2.7 in the bipartite graph corresponding to $\{a^j : j \in J^+(u)\} \cup \{a^\gamma\}$. In the first one, the obtained A-triple contains γ. In the other one, the obtained A-triple contains only vertexes from V_2^C with $C := \{a^j : j \in J^+(u)\}$ where a^γ serves only as a connection for the arising paths. In the first case, e.g., when the terms of Lemma 2.10 are satisfied, it is enough to find A-triples which starts with γ. For that purpose we use the characterization given in Lamma 2.12. This case is considered further in Algorithm 2.3. In the second case we apply Algorithm 2.4 which uses the characterization given in Lemma 2.13.

Let us consider the case where A-triples contain γ. Based on a set $C := \{a^j : j \in J^+(u)\}$ we build the corresponding convex bipartite graph B_γ without vertexes and edges which could be incident to γ, if we would add it to V_2^C. The resulting column set has the C1P due to Lemma 2.8 and the corresponding graph B_γ has no A-triples due to Corollary 2.3.

We search for all shortest paths between all two different vertexes of V_2^γ in B_γ, refer to Algorithm 2.3. Those pairs of vertexes for which there exist a path we consider as candidates entering to an A-triple together with the vertex γ due to Statement 2.7 and the assumption that the obtained A-triple contains γ. Now we test whether there exists a path between vertexes of these pairs and γ every time avoiding the closed neighborhood of the third vertex, respectively. If all three paths exist then (γ, p, q) builds an A-triple. All vertexes from I which are in these paths correspond to the desired subcolumn.

Algorithm 2.3 (SUBCOL-OF-A-COL-A)**.** *Determination of forbidden subcolumns of a given column in the case when A-triples start with the vertex corresponding to the given column.*
Input data: A node u, the column a^γ with $\gamma \in \bar{J}^-(u)$.
Output data: $D(u)$.

(*1*) *For each pair of vertexes $p, q \in V_2$ with $p < q$ find a shortest path $(p \to q)$ from p to q in*

$$B_\gamma := (V_1^\gamma := I \backslash N(\gamma), V_2^\gamma := J^+(u), E_\gamma := \{(i,j) \in V_1^\gamma \times V_2^\gamma : \ a_i^j = 1\}).$$

If $(p \to q) \neq \emptyset$ then add (p, q) to $\mathcal{P}_0$.

(*2*) *For each $p, q \in V_2$ with $p < q$ find a shortest path from γ to q without nodes in the closed neighborhood of p, i.e., find $(\gamma \to q)^p$ in*

$$B_p := (V_1^p := I \backslash N(p), V_2^p := J^+(u) \cup \{\gamma\} \backslash \{p\}, E_p := \{(i,j) \in V_1^p \times V_2^p : \ a_i^j = 1\}).$$

If $(\gamma \to q)^p \neq \emptyset$ then add (γ, q) to $\mathcal{P}_p$.

(*3*) *For each $(p, q) \in \mathcal{P}_0$:*
If $(\gamma, p) \in \mathcal{P}_q$ and $(\gamma, q) \in \mathcal{P}_p$ then (γ, p, q) forms an A-triple, hence form a subcolumn d^* of a^γ:
$d_i^* = a_i^\gamma$, $\forall i \in I_k$ with $I_k := [(p \to q) \cup (\gamma \to q)^p \cup (\gamma \to p)^q] \cap I$,
add d^* to $D(u)$, if $d^* \notin D(u)$.

Lemma 2.14. *Let $u \in V$ be a node of the branching tree T, and a^γ with $\gamma \in \bar{J}^-(u)$. The algorithm* SUBCOL-OF-A-COL-A *is correct and returns in $O(m^2(M + \log m))$ time a subcolumn of a^γ which breaks the C1P, if in the corresponding bipartite graph there exists any A-triple containing γ.*

Proof. The proof of the lemma consists of two parts. Let $C_u^+ := \{a^j : j \in J^+(u)\}$ and $\widetilde{C} := C_u^+ \cup \{a^\gamma\}$. Since $\widetilde{C}$ does not have the C1P and due to Statement 2.7 there exists an A-triple, we should prove that the algorithm finds this A-triple. Secondly, the time complexity should be calculated.

So, due to Corollary 2.6, the graph $B_{\widetilde{C}} := (V_1^{\widetilde{C}}, V_2^{\widetilde{C}}, E_{\widetilde{C}})$ has at least one of the graphs G_k^1, G_k^2, G_k^3, G^4, and G^5 as a subgraph. Let G_k^1 be such a graph. Due to Corollary 2.3, the bipartite graph $B_{C_u^+}$ has no A-triple. Since the sets $V_2^{\widetilde{C}}$ and $V_2^{C_u^+}$ differ only in one element, namely γ, it means that γ builds an A-triple together with two other vertexes p, q with $p \neq q$ from $V_2^{C_u^+}$ which are in the vertex set of G_k^1. Note there always exists a path $(p \to q)$ in $B_{C_u^+}$ which is found on step 1 of the algorithm. Fig. 2.2a indicates also that there exist the paths $(\gamma \to p)^q$ and $(\gamma \to q)^p$ which are found on step 2 of the algorithm. Since all three paths $(p \to q)$, $(\gamma \to p)^q$, and $(\gamma \to q)^p$ exist then (γ, p, q) is found on step 3 of the algorithm. The same can be proved for G_k^2, G_k^3, G^4, and G^5, see Fig. 2.2.

The time complexity is calculated as follows. It performs m^2 operations to construct a bipartite graph using a matrix. So, graph B_γ is build in $O(m^2)$ time (step 1). The rest can be done by one run of the Dijkstra's algorithm or the Floyd-Warshall's algorithm (WFI) [CLR90]. The complexity of the Dijkstra's algorithm is $O(|E_C| + (|V_1 + V_2|)\log(|V_1 + V_2|))$ where $|E_C|$ is the number of edges and $|V_1 + V_2| = 2m$ is the number of vertexes in the graph. Concerning that the maximal number of items fitting into a bin is $M := \max\{\sum_{i\in I} a_i : \sum_{i\in I} w_i a_i \leq W,\ a_i \in \{0,1\}\}$, so the number of 1's in the whole matrix, thus number of edges is $|E| = mM$. Hence, $O(|E_C| + (|V_1 + V_2|)\log(|V_1 + V_2|)) = O(mM + 2m\log 2m)$ and the complexity of step 1 is $O(m^2(M + \log m))$. Therefore, the complexity of step 2 is also the same because the graph B_p can be obtained by a linear modification of B_γ. Since the cardinality of the set $\mathcal{P}_0$ is $\frac{m}{2}(m-1)$, and we need a constant time to test whether a path is in $\mathcal{P}_p$ (step 3), the complexity of this step is $O(m^2)$. Therefore, SUBCOL-OF-A-COL-A's complexity is $O(m^2(M + \log m))$. □

Remark 2.2. *M in the formula of* SUBCOL-OF-A-COL-A*'s complexity tends to be constant, but large. Hence,* SUBCOL-OF-A-COL-A*'s complexity is $O(m^2 \log m)$.*

Let us consider the case, where A-triples do not contain γ. Based on $\widetilde{C} := C_u^+ \cup \{a^\gamma\}$ we build bipartite graph $\widetilde{B} = (V_1^{\widetilde{C}}, V_2^{\widetilde{C}}, E_{\widetilde{C}})$, which due to Statement 2.7 contains an A-triple. In order to find this A-triple we compute all shortest paths between two different vertexes p and q not equal to γ from $V_2^{\widetilde{C}}$ in $B^{\widetilde{C}}$ each time avoiding the direct neighborhood of the third vertex r. Herewith, we obtain the set $\mathcal{P}_{pq}^r$ for all p, q and r. Due to the assumption the obtained A-triple does not contain γ, so for all different p, q, r and not equal to γ we test whether paths $(p \to q)^r \in \mathcal{P}_{pq}^r$, $(p \to r)^q \in \mathcal{P}_{pr}^q$, and $(q \to r)^p \in \mathcal{P}_{qr}^p$. If so then p, q, and r build an A-triple. Algorithm 2.4 contains details of the above algorithm.

Algorithm 2.4 (SUBCOL-OF-A-COL-B)**.** *Determination of forbidden subcolumns of a given column in the case when A-triples do not contain γ.*
Input data: A node u, the column a^γ with $\gamma \in \bar{J}^-(u)$.
Output data: $D(u)$.

(1) *For each vertex $p, q, r \in J^+(u)$ with $p < q$, $r \neq p$ and $r \neq q$, find a shortest path $(p \to q)^r$ from p to q without the closed neighborhood of γ in*

$$B_r := (V_1^r := I \backslash N(r), V_2^r := (J^+(u)\backslash\{r\}) \cup \{\gamma\}, E_r := \{(i,j) \in V_1^r \times V_2^r :\ a_i^j = 1\}).$$

If $(p \to q)^r \neq \emptyset$ *then add* $(p,q)^r$ *to* $\mathcal{P}^r_{pq}$.

(2) *For each* $p, q, r \in J^+(u)$ *with* $p \neq q$, $q \neq r$, *and* $p \neq r$:
If $(p,q)^r \in \mathcal{P}^r_{pq}$ *and* $(q,r)^p \in \mathcal{P}^p_{qr}$ *and* $(p,r)^q \in \mathcal{P}^q_{pr}$ *then* (p,q,r) *forms an A-triple, hence form the subcolumn* d^* *of* a^γ:
$d^*_i = a^\gamma_i$, $\forall i \in I_k$ *with*
$I_k := [N(\gamma) \cap (p \to q)^r] \cup [N(\gamma) \cap (p \to r)^q] \cup [N(\gamma) \cap (q \to r)^p]$,
add d^* *to* $D(u)$, *if* $d^* \notin D(u)$.

Lemma 2.15. *Let* $u \in V$ *be a node of the branching tree* T, *and* a^γ *with* $\gamma \in \bar{J}^-(u)$. *The algorithm* Subcol-of-a-col-B *is correct and returns in* $O(m^3(M + \log m))$ *time a subcolumn of* a^γ *which breaks the C1P, if in the corresponding bipartite graph there exists any A-triple which does not contain* γ.

Proof. The time complexity can be calculated as follows. It performs m^2 operations to construct a bipartite graph using a matrix. So, the graph B_r is build in $O(m^2)$ time (step 1). The rest can be done by one run of the Dijkstra's algorithm or the Floyd-Warshall's algorithm (WFI) [CLR90]. As it was shown, it takes $O(|E_C| + (|V_1 + V_2|)\log(|V_1 + V_2|)) = O(mM + 2m\log 2m)$ operations, so the total complexity of step 1 is $O(m^3(M + \log m))$. Since the test whether $(p,q)^r \in \mathcal{P}^r_{pq}$ for any p, q, r is constant, and we perform this test for each p, q, and r then the complexity of step 2 is $O(m^3)$. Therefore, Subcol-of-a-col-B's complexity is $O(m^3(M + \log m) + m^3) = O(m^3(M + \log m))$. □

Remark 2.3. *M in the formula of* Subcol-of-a-col-B*'s complexity tends to be constant, but large. Hence,* Subcol-of-a-col-B*'s complexity is* $O(m^3 \log m)$.

2.4 Subcolumns potentially breaking the C1P

Let $u \in V$ be a node of the branching tree T. Let us now answer the following question: what combinations of elements of a column will break the C1P of $C^+_u := \{a^j : j \in J^+(u)\}$ being joint with a column? In order to solve this problem we perform the following steps. Firstly, we use the associated bipartite graph $B_0 := (V_1, V_2, E_0)$ based on C^+_u and search for every vertex all vertex-disjoint paths to any other vertex. Graph B_0 has no A-triples due to Corollary 2.3.

Definition 2.3. *Let* $(p \to q)_1 := \{t^1_1, \ldots, t^1_l\}, \ldots, (p \to q)_\chi := \{t^\chi_1, \ldots, t^\chi_l\}$ *be paths from* p *to* q *with* $(p \to q)_1 = \ldots = (p \to q)_\chi$. *Path* $(p \to q)_\phi$ *with* $\phi \in \{1, \ldots, \chi\}$ *is exclusive if* $t^\phi_j \leq t^f_j$, $f \in \{1, \ldots, \chi\} \backslash \{\phi\}$, $j = 1, \ldots, l$.

So, only exclusive paths are considered.

Let us consider an imaginary column $s^\gamma \in \{0,1\}^m$ represented by a vertex γ when added to B_0. γ is incident to every vertex of V_1. We suppose that now γ induces an A-triple together with the vertexes from V_2. Due to Statement 2.7 the set $C^+_u \cup \{s^\gamma\}$ does not have the C1P. Now for every initial and final vertex p, q of the above paths we determine whether there exist paths $(\gamma \to p)$ and $(\gamma \to q)$ each time avoiding a direct vertex and edge neighbors of q and p, respectively, and avoiding vertexes from

V_1, which are in the path $(p \to q)$. If all three paths exist then (γ, p, q) is an A-triple. In such a way all A-triples are searched which can result when an additional column is added to C_u^+.

Now going back to subcolumns we can conclude that items from V_1 which were in the paths $(\gamma \to p)^q$, $(\gamma \to q)^p$, and $(p \to q)$ should correspond to the desired subcolumn. Moreover, the vertexes from V_1 which are in $(p \to q)$ correspond to rows with 0 and vertexes which are in $(\gamma \to p)^q$, $(\gamma \to q)^p$ and not in $(p \to q)$ correspond to rows with 1 in the subcolumn. Algorithm 2.5 contains details of the above algorithm.

Algorithm 2.5 (SUBCOLS)**.** *Determination of forbidden subcolumns.*
Input data: A node u, the column set $\{a^j : \ j \in J^+(u)\}$.
Output data: $D(u)$.

(1) Set $B_0 := (V_1 := I, V_2 := J^+(u), E_0 := \{(i,j) \in V_1 \times V_2 : \ a_i^j = 1\})$.

(1.1) For each pair of vertexes $p, q \in V_2$ with $p < q$ find all exclusive paths from p to q, i.e., $(p \to q)$ in B_0. So, we obtain the set $\mathcal{P}_{pq}$ of paths from p to q in B_0.

(1.2) If two paths $(p \to q)_1 \in \mathcal{P}_{pq}$, $(p \to q)_2 \in \mathcal{P}_{pq}$ satisfy $V_1(p \to q)_1 \subsetneqq V_1(p \to q)_2$ then delete $(p \to q)_2$ from $\mathcal{P}_{pq}$ (where $V_1(p \to q) = \{i \in V_1 : \ i \in (p \to q)\}$).

(2) Set $B_p := (V_1^p := I \backslash N(p), V_2^p := V_2 \backslash \{p\}, E_p := \{(i,j) \in V_1^p \times V_2^p : \ a_i^j = 1\})$.

(2.1) For all $p \in V_2$ and all $i \notin N(p)$ find for all $q \in V_2 \backslash \{p\}$ all exclusive paths from i to q in B_p. So, we obtain set $\mathcal{P}_{iq}^p$ of paths from i to q without the closed neighborhood of vertex p in B_p.

(2.2) If two paths $(i \to q)_1^p \in \mathcal{P}_{iq}^p$, $(i \to q)_2^p \in \mathcal{P}_{iq}^p$ satisfy $V_1(i \to q)_1^p \subsetneqq V_1(i \to q)_2^p$ then delete $(i \to q)_2^p$ from $\mathcal{P}_{iq}^p$.

(3) For each nonempty path $(p \to q) \in \mathcal{P}_{pq}$,
for all $i_1 \notin V_1(p \to q)$ such that a path $(i_1 \to q)^p \in \mathcal{P}_{i_1 q}^p$ exists,
for all $i_2 \notin V_1(p \to q)$ such that a path $(i_2 \to p)^q \in \mathcal{P}_{i_1 q}^p$ exists,
set $k = k + 1$,

$$d_i^k = \begin{cases} 1, & i \in (V_1(i_1 \to q)^p \cap V_1(i_2 \to p)^q) \backslash V_1(p \to q); \\ 0, & i \in V_1(p \to q); \end{cases}$$

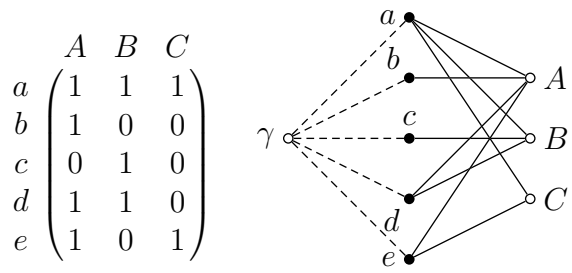
with $I_k = V_1(p \to q) \cup V_1(i_1 \to q)^p \cup V_1(i_2 \to p)^q$.

Lemma 2.16. *Let $u \in V$ be a node of the branching tree T. The algorithm* SUBCOLS *is correct. The time complexity tends to be exponential.*

Let us consider the following example which uses the algorithm SUBCOLS.

Example 2.1. *Let us consider a column set on Fig. 2.5a which has the C1P. Based on this matrix we build the corresponding bipartite graph, see Fig. 2.5b.*

1. *Find all exclusive paths between all pairs of the vertexes from V_2:*
   $(A \to *)$: (AaB), (AdB), (AaC), (AeC), $(AdBaC)$;
   $(B \to *)$: (BaA), (BdA), (BaC), $(BdAeC)$;
   $(C \to *)$: (CaA), (CeA), $(CaBdA)$, (CaB), $(CeAdB)$, $(CaAdB)$;



$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}$$

(a)

(b)

1)	2)	3)	4)	5)	6)
0	0	$*$	$*$	0	0
1	$*$	1	$*$	$*$	$*$
1	1	1	1	1	$*$
$*$	$*$	0	0	$*$	1
$*$	1	$*$	1	1	1

(c)

Figure 2.5: The SUBCOLS algorithm: (a) – A column set with the C1P; (b) – The bipartite graph corresponding to the column set in (a); (c) – The columns that would break the C1P of the column set in (a).

2. *Find all exclusive paths between an imaginary vertex γ and the vertexes from V_2:*
   $(\gamma \to *)^A$*:* (γcB)*;*
   $(\gamma \to *)^B$*:* (γbA)*,* (γeA)*,* (γeC)*;*
   $(\gamma \to *)^C$*:* (γbA)*,* (γcB)*,* (γdA)*,* (γdB)*;*

3. *The following vertex triples are suspected to form an A-triple:* (γ, A, B)*,* (γ, A, C)*,* (γ, B, C)*:*

   (a) $(A \to B)^\gamma + (\gamma \to A)^B + (\gamma \to B)^A$*?*
   $\begin{bmatrix} AaB \\ AdB \end{bmatrix} + \begin{bmatrix} \gamma bA \\ \gamma eA \end{bmatrix} + \begin{bmatrix} \gamma cB \end{bmatrix} \Rightarrow \left\{ \begin{matrix} (abc), & (aec) \\ (dbc), & (dec) \end{matrix} \right\} \Rightarrow (\gamma, A, B)$ *is an A-triple*
   1) $a = 0$*,* $b = c = 1$*;*
   2) $a = 0$*,* $e = c = 1$*;*
   3) $d = 0$*,* $b = c = 1$*;*
   4) $d = 0$*,* $e = c = 1$*;*

   (b) $(A \to C)^\gamma + (\gamma \to A)^C + (\gamma \to C)^A$*?*
   $\begin{bmatrix} AaC \\ AeC \\ AdBaC \end{bmatrix} + \begin{bmatrix} \gamma bA \\ \gamma dA \end{bmatrix} + \emptyset$*;*

   (c) $(B \to C)^\gamma + (\gamma \to B)^C + (\gamma \to C)^B$*?*
   $\begin{bmatrix} BaC \\ BdAeC \end{bmatrix} + \begin{bmatrix} \gamma cB \\ \gamma dB \end{bmatrix} + \begin{bmatrix} \gamma eC \end{bmatrix} \Rightarrow \left\{ \begin{matrix} (ace), & (ade) \\ \emptyset \end{matrix} \right\} \Rightarrow (\gamma, B, C)$ *is an A-triple*
   5) $a = 0$*,* $c = e = 1$*;*
   6) $a = 0$*,* $d = e = 1$*;*

Note subcolumns 2 and 5 are equal. The subcolumns that breaks the C1P are shown on Fig. 2.5c. So, setting, e.g., $a = 0$*,* $b = c = 1$ *leads to an A-triple.*

So, the set $D(u) := \{d^1, \ldots, d^5\}$*, where* $d^1 = d^2 = d^5 = d^6 = \{0, 1, 1\}$*,* $d^3 = \{1, 1, 0\}$*,* $d^4 = \{1, 0, 1\}$*, and* $I_1 = \{1, 2, 3\}$*,* $I_2 = I_5 = \{1, 3, 5\}$*,* $I_3 = \{2, 3, 4\}$*,* $I_4 = \{3, 4, 5\}$*,*

$I_6 = \{1, 4, 5\}$. *Thus, the constraints to be added into* $\mathrm{CG}(u)$ *are:*

$$a_1 + (1 - a_2) + (1 - a_3) > 0$$
$$a_1 + (1 - a_3) + (1 - a_5) > 0$$
$$(1 - a_2) + (1 - a_3) + a_4 > 0$$
$$(1 - a_3) + a_4 + (1 - a_5) > 0$$
$$a_1 + (1 - a_3) + (1 - a_5) > 0$$
$$a_1 + (1 - a_4) + (1 - a_5) > 0$$

If the cardinality of the set $\mathcal{P}_{pq}$ is acceptable then we can improve the algorithm SUBCOLS by using less calls of the "all exclusive paths" procedure, refer to Algorithm 2.6.

Algorithm 2.6 (SUBCOLS'). *Determination of forbidden subcolumns.*
Input data: A node u*, the column set* $\{a^j : \ j \in J^+(u)\}$.
Output data: $D(u)$.

(*1*) *Construct the bipartite graph* $B_0 := (V_1 := I, V_2 := \{1, \ldots, |J^+(u)|\}, E_0 := \{(i, j) : \ j \in V_2, \ i \in N(j)\})$.
(*2*) *For each pair of vertexes* $p \in V_2$, $q \in V_2 \backslash \{p\}$: $p < q$ *in* B_0 *find all exclusive paths* ALLPATHS$(B_0, p \to q)$ *and save these in* $\mathcal{P}_{pq}$.
(*3*) $V_1 := I \backslash [(p \to q) \cap I]$, $V_2 := \{1, \ldots, |J^+(u)|, \gamma\}$,
$E_\gamma := E_0 \cup \{(i, \gamma) : \ i \in I\} \backslash \{(\gamma, o), (o, r) : \ o \in (p \to q) \cap I, \ r \in V_2\}$.
For each pair of vertexes $(p \to q) \in \mathcal{P}_{pq}$ *build the following two graphs:*
$B_p := (V_1 \backslash N(p), V_2 \backslash \{p\}, E_\gamma \backslash \{(\gamma, o), (o, r) : \ o \in N(p), \ r \in V_2\})$;
$B_q := (V_1 \backslash N(q), V_2 \backslash \{q\}, E_\gamma \backslash \{(\gamma, o), (o, r) : \ o \in N(q), \ r \in V_2\})$;
If there exists a path $(\gamma \to p)$ *in* B_q *and* $(\gamma \to q)$ *in* B_p *then* (γ, p, q) *builds an A-triple. Then* $D(u) := D(u) \cup \{d^*\}$ *with*
$$\begin{cases} d^*_i := 0, & i \in (p \to q) \cap I; \\ d^*_i := 1, & i \in [(\gamma \to p) \cup (\gamma \to q)] \backslash [(p \to q) \cap I]; \end{cases}$$

Algorithm 2.7 (ALLPATHS). *The all paths algorithm: the algorithm is a modification of the depth-first search (DFS) which stores at every step a list of the traversed vertexes and the modified incidence matrix.*
Input data: Start S*,* E*, and the incident matrix* M.
Output data: Set W *of all exclusive ways between* S *and* E.

(*1*) *Push node* S *and matrix* M *to the stack, Push(*S*,* M*).*
(*2*) *Restore the current node* o *and the matrix* M *from the stack,* $(o, M) \leftarrow$ *Pop(). If* $o \neq \emptyset$ *then all paths from* S *to* E *are investigated and goto Exit.*
(*3*) *If node* o *is equal to* E *then a new path from* S *to* E *is found, otherwise restore the next node: If* $o = E$ *then store this path* $W := W \cup \{(S \to E)\}$ *and goto 2. Else find each node which is incident to* o*, delete the incidence record between them in* M *and push them into the stack: For all* j *with* $m^j_o = 1$ *set* $m^j_o = m^o_j = 0$ *and Push(*j*,* M*). Goto 2.*

Note Algorithm 2.7 has an exponential time complexity and dm^2 space complexity where d is the maximal depth of the tree.

Example 2.2. *On Fig. 2.6a there is a bipartite graph which corresponds to a column set. On Fig. 2.6b a search tree is shown which is build according to the* AllPaths *algorithm for finding the all exclusive paths between nodes A and C. So, we have found the following paths: AaC, $AaBbC$, AbC, $AbBaC$. The dashed edges of the tree mark the cycles.*

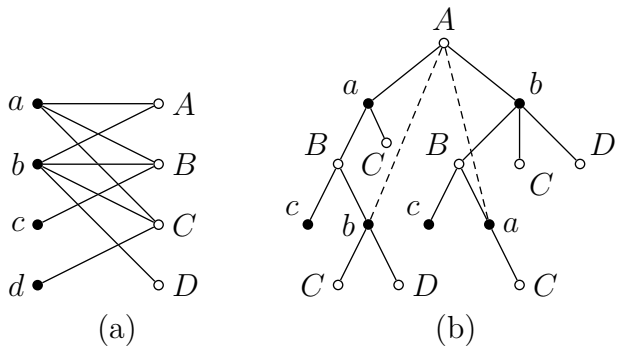


Figure 2.6: The AllPaths algorithm: (a) – A bipartite graph; (b) – A search tree which is build for finding all exclusive paths between nodes A and C.

2.5 Numerical study

In this section we discuss numerical experiments for the CBPP-1 instances which are obtained from SPP-2 from different sources.

The algorithm was implemented as a single-threaded application in C++ based on Visual Studio 2008, compiler version 9.0.30729, on an AMD Athlon 64 Dual Core 4200+ (2.2 GHz) CPU. IBM ILOG CPLEX 12.5 was used as an LP solver. The test instances, detailed results and source code are available on the CaPaD website[2] and in [MSB13b].

In Table 2.1 the number of nodes and time are the mean values over the solved instances. From a rational number we take only the integer part without rounding.

Here are the following implementation issues to consider:

1. The time limit for each instance and method was set to 900 seconds.

2. For obtaining a good performance of the branch-and-price algorithms we limit the iterations for the algorithms from Sections 2.3, 2.4 by 1000.

3. We do not consider any methods of stabilization or optimization of the column generation process, since this is not a subject of the research of this paper. Considering of these steps may result in a better computational behavior of the branch-and-price method.

[2]http://www.math.tu-dresden.de/∼capad

4. In Table 2.1 the results for the decision problem of CBPP-1 are shown. The decision problem asks whether a set of items can be packed into the bins feasibly, so that the number of used bins does not exceed 20. If so then the instance is called feasible, otherwise infeasible.

2.6 Conclusions

Here we have proposed and studied news branch-and-price methods for the 1D contiguous bin packing problem.

The main theoretical and experimental observations of the paper are the following:

1. Given a column set with the consecutive 1's property and a column which breaks this property it is computationally not very consuming to find all the subcolumns of the column which break the property. If we the column breaking the consecutive 1's property is not given, the problem becomes computationally hard.

2. The branching strategy based on the enumeration of the columns behaves on the whole better on feasible instances where we have to find a solution. On the other hand the branching strategy based on the enumeration of the subcolumns is of a beneficial use for the infeasible instances where the infeasibility has to be proven by increasing the bound.

3. The following issues are subject to further study: can we model (possibly remaining in the linear programming) the 2D strip packing problem using two models of the 1D contiguous bin packing in a single formulation as we did in [MSB13a]?

2.7 Acknowledgments

We appreciate the Academic Initiative of IBM which enables many researchers all over the world to compare their methods using state-of-the-art IBM ILOG Optimization Software.

Table 2.1: Results of the decision version of CBPP-1 on the 2D instances from [CJM08]: A – Branch-and-price-cols; B – Branch-and-price-subcols; n – total number of nodes; t – total time.

	A			B		
inst	opt	n	t	opt	n	t
Infeasible instances						
00N10	1	1	0	1	1	0
00N15	1	1	0	1	1	0
00N23	1	1	0	1	1	0
00X23	0	-	-	0	-	-
02N20	1	1	0	1	1	0
03N10	1	84	1	1	27	1
03N15	1	1	0	1	1	0
03N16	1	461	4	1	324	4
03N17	1	1	0	1	1	0
04N15	1	1	0	1	1	0
04N17	1	1	0	1	1	0
04N18	1	1	0	1	1	0
05N15	1	1	0	1	1	0
05N17	1	215	4	1	101	3
05X15	1	1	0	1	1	0
07N10	1	1	0	1	1	0
07N15	1	1	0	1	1	0
07X15	1	80	2	1	79	2
08N15	1	3	0	1	3	0
10N10	1	1	0	1	1	0
10N15	1	1	0	1	1	0
10X15	1	1	0	1	1	0
13N10	1	466	1	1	420	1
13N15	1	1	0	1	1	0
13X15	1	1	0	1	1	0
15N10	1	60	1	1	57	0
15N15	1	1	0	1	1	0
mean	26	53	1	26	40	0
Feasible instances						
02F17	1	10371	14	1	17415	24
02F20	1	2350	5	1	7203	15
02F22	1	3201	9	1	8011	20
03X18	1	1280	6	1	1407	7
04F15	1	633	3	1	914	3
04F17	1	131	1	1	135	1
04F19	1	965	4	1	1134	5
04F20	1	318	7	1	510	8
05F15	1	382	2	1	479	2
05F18	1	101	1	1	108	1
05F20	1	415	2	1	813	4
07F15	1	544	2	1	1437	4
08F15	1	54	1	1	60	1
20F15	1	49	1	1	78	1
20X15	1	39	0	1	40	0
mean	15	1389	4	15	2650	6

Chapter 3

Constraint Programming Approaches for Orthogonal Packing

We consider the 2D orthogonal feasibility problem (OPP-2) and the 2D strip packing problem (SPP-2). Given a set of rectangular items, OPP-2 is to decide whether all items can be orthogonally packed into the given rectangular container; SPP-2 is to find a packing of all items occupying the minimal height of the given semi-infinite strip. We investigate the known Constraint Programming (CP) approaches for OPP-2, in particular the dichotomy and disjunctive branching strategies and adapt 1D relaxation bounds based on linear programming (LP) into the constraint propagation process of the CP. Using the dichotomic search procedure the developed methods for OPP-2 are transformed for the case of SPP-2. Numerical results demonstrate the efficiency of the proposed strategies and of the combination of CP and LP-based pruning rules.

3.1 Introduction

Let us consider a set of m rectangular items (w_i, h_i), with $i \in I := \{1, \ldots, m\}$. The 2-*dimensional orthogonal packing feasibility problem* (OPP-2) [FS04a, CJCM08] asks whether all the items (w_i, h_i) with $i \in I$ can be orthogonally packed into the given container (W, H) without rotations. The guillotine constraint [MAVdC10, CJM08] is not considered. All input data are positive integers, i.e., $W, H \in \mathbb{Z}_+$, and $w_i \in \{1, \ldots, W\}$, $h_i \in \{1, \ldots, H\}$ for $i \in I$. All problems which are introduced in this paper can be easily generalized for higher dimensions, so sometimes they will be mentioned without an indication of the dimension.

In the case when the items cannot be packed into the container, it is enough to declare a negative response, otherwise the solution is a feasible packing layout with the items allocated orthogonally and non-overlapped within the container. OPP is a subproblem in solution methods for orthogonal bin packing (BPP) and knapsack problems (OKP) [FS04a, BB07, PS07]. OPP is polynomially equivalent to the orthogonal strip-packing problem (SPP) [Hif98, AVPT09, KIN$^+$09]. The latter problem can be solved by the dichotomic search, see Section 3.4.8.

3.1.1 Formulation of OPP and overview of solution methods

Suppose we have a coordinate system with origin $(0, 0)$ matched with the left bottom point of the container whose x, y-axis are associated with its W, H-sides, respectively.

Let us introduce sets of variables $X := \{x_i : i \in I\}$ and $Y := \{y_i : i \in I\}$ which represent the allocation points for the items in the $(0, x)$-, $(0, y)$-directions, respectively. An assignment of certain values to the variables is feasible in the sense of OPP-2, if the following constraints are satisfied:

$$\begin{aligned} &x_i + w_i \le x_j \vee x_j + w_j \le x_i \vee \\ &y_i + h_i \le y_j \vee y_j + h_j \le y_i, \quad (i,j) \in I \times I : \ i < j; && (3.1)\\ &0 \le x_i \le W - w_i, \quad i \in I; && (3.2)\\ &0 \le y_i \le H - h_i, \quad i \in I; && (3.3)\\ &x_i, y_i \in \mathbb{Z}_+, \quad i \in I. && (3.4) \end{aligned}$$

It is assured that the items do not overlap, the constraints (3.1), and lie within the container, the constraints (3.2), (3.3). If there exist values of the variables so that the constraints (3.1)-(3.4) are satisfied then the instance is called feasible, otherwise infeasible.

So, the above tiny formulation [CJM08, PS07] is a valid non-linear integer model of OPP-2. In order to solve OPP-2 in this formulation, some constraint programming methods are successfully applied, which leads to the best results today [CJM08, SO08, Sim08, BCDP11, PS07, KMP10]. The approaches can be divided into two groups, the first one fixes the coordinates of the items [CJM08, SO08, Sim08], the other fixes the mutual position of the items [PS07, SO08, Sim08, BCDP11].

One of the reasons of the success of the constraint programming paradigm is the efficient *constraint propagation* technique. In every node of the search tree it tries to decide whether the set of the constraints (3.1)-(3.4) is consistent. In other words, it tries to prove the infeasibility of the current partial solution when certain values are assigned to some variables or domains of possible values of the variables are restricted. If an inconsistency of the set of constraints cannot be proven then the procedure tries to reduce the domain of possible values for the variables.

There exist many ILP models [Bea85, Pad00, BB07, BKRS09] for OPP-2 based on different representations of a feasible solution. Exact solution of OPP-2 in the above ILP formulations is difficult because of the weak LP bounds of some models [Pad00], quadratic number of intersection variables, and/or pseudo-polynomial number of position-indexed variables [Bea85, BB07] in some models.

Modeling with rotations

Rotation of items by 90° withing (3.1)-(3.4) can by modeled by an introduction of binary variables r_i, $i \in I$ which indicate in case of $r_i = 1$ that item i is rotated by 90°.

The resulting model is:

$$\begin{aligned}
x_i + w_i(1 - r_i) + h_i r_i \leq x_j \vee & \\
y_i + h_i(1 - r_i) + w_i r_i \leq y_j \quad & (i,j) \in I \times I : \ i \neq j; && (3.5)\\
0 \leq x_i \leq W - w_i(1 - r_i) - h_i r_i, \quad & i \in I; && (3.6)\\
0 \leq y_i \leq H - h_i(1 - r_i) - w_i r_i, \quad & i \in I; && (3.7)\\
x_i, y_i \in \mathbb{Z}_+, \quad & i \in I. && (3.8)\\
r_i \in \{0,1\}, \quad & i \in I. && (3.9)
\end{aligned}$$

Further in this paper we consider OPP without rotations.

3.1.2 Relaxations and bounds for OPP

In order to decide whether an instance of OPP is infeasible sometimes it is enough to solve a relaxation. These are of different kinds, i.e., *volume bounds*, *dual-feasible functions* (DFF) [CAVdC10a], *conservative scales* (CS) [FS04b, BKRS13], *1D bar relaxations* [Sch99, BKRS09], and relaxations of ILP models. All of the mentioned bounds are discussed in [BKRS13].

Conservative scales (CS) are modified items sizes such that if the OPP instance is feasible then it is also feasible with the modified sizes. Thus, the volume bound for the modified instances is valid for the original instance. In fact, CS are valid inequalities for a certain 0-1 knapsack polyhedron.

Dual feasible functions (DFFs) produce a subset of CS [BKRS13]. The concept of DFF has been firstly used to obtain algorithmic lower bounds for bin packing problems [Lue83]. Most of the methods proposed for computing DFFs are polynomial and rely on known families of functions [CAVdC10a, CCM07, FS01, FS04b, CAVdC10b]. For the 2D problems, DFFs are rather weak [BKRS13] in contrast to 3D problems. So-called *maximal CS* cannot always be obtained by DFFs; corresponding methods are proposed in [BKRS13].

The 1D bar relaxation [Sch99, BKRS09] is a double relaxation of OPP-2. Firstly, we divide the container and items into 1D bars with unit thickness. Further we formulate the minimization problem over the number of used 1D bars which are needed to pack all the split items without repetition in a single bar. Secondly, we formulate a set-partitioning model of the above 1D problem, continuously relax it, and solve it by the column generation method [KZ51, GG61, GG63]. The 1D bar relaxation bound can be further strengthened [BKRS09] by the additional information, i.e., from a probing procedure which restricts the set of items combined in the bars.

Up to know, there were only few efforts to use the bar relaxation in an algorithm for OPP. In [BR13] the 1D bar relaxation bounds were integrated into a modified interval-graph algorithm from [FS04a]. The bound was also tightened in each dimension using the overlapping information from the graphs. This extended information was used in the column generation. The tightened bound was applied in every node of the branching tree.

3.1.3 Our contributions

In order to solve OPP we investigate and modify the state-of-the-art constraint programming approaches [CJM08, SO08, Sim08]. We compare some known branching strategies, e.g., contour, interval, disjunctive, and propose new *dichotomy* and *intersection* strategies, in Section 3.4. In Section 3.5 we propose new pruning rules based on the tightened 1D bar relaxations [Sch99, BKRS09] of various kinds, e.g., intervals, forbidden pairs, and an advanced bar relaxation. The input data for the bar relaxation is obtained from the local partial solution, the information from the constraint propagation procedure, and the relative positions of the items in the container. The final part of the paper reports numerical results and conclusion.

3.2 An overview of the algorithm of Clautiaux et al.

In this section we consider the algorithm of Clautiaux et al. [CJM08] and give some necessary definitions.

3.2.1 Basic definitions

Clautiaux et al. [CJM08] proposed an algorithm for OPP-2 which solves it as two scheduling problems. Let be given sets $A^W := \{A_1^W, \ldots, A_m^W\}$ and $A^H := \{A_1^H, \ldots, A_m^H\}$ of activities and two types of resources of H and W units. Each of the activities A_i^d with $d \in \{W, H\}$ and $i \in \{1, \ldots, m\}$ has its time interval $[start_i^d, end_i^d)$ where the activity A_i^d can occur. $start_i^d$ designates the earliest point of time where activity A_i^d can start, and end_i^d is the latest point of time where activity A_i^d can end. Each activity $A_i^W \in A^W$, $A_i^H \in A^H$ has its level of consumption h_i of the resource H, w_i of the resource W, and durations w_i, h_i, respectively. A schedule is called continuous, if in the schedule each activity is not interrupted during the execution, and cumulative, if all activities are consuming the same resource. At every discrete point of time the resource capacity is limited by a certain value and must not be exceeded.

Let us now superpose each feasible start time point of the activity A_i^W with the variables from X, and A_i^H with the variables from Y then $x_i \in [start_i^W, end_i^W - w_i] \cap \mathbb{Z} = [\underline{x}_i, \overline{x}_i] \cap \mathbb{Z}$ and $y_i \in [start_i^H, end_i^H - h_i] \cap \mathbb{Z} = [\underline{y}_i, \overline{y}_i] \cap \mathbb{Z}$, i.e., $[\underline{x}_i, \overline{x}_i]$ and $[\underline{y}_i, \overline{y}_i]$ are feasible domains for the variables. If we now assume that $\underline{x}_i = \underline{y}_i := 0$, $\overline{x}_i := W - w_i$, $\overline{y}_i := H - h_i$ with $i \in I$, both schedules are cumulative and continuous, and if the constraints (3.1) for the variables from X and Y are satisfied then the model based on two scheduling problems connected through the constraints (3.1) is a feasible model of OPP [CJM08].

The formulation of OPP-2 modeled by two scheduling problems is solved by the branch-and-bound method, i.e., a branching tree $T := (V, E)$ is built. Two nodes $u, v \in V$ differ by the local set of branching restrictions. Based on the kind of branching restrictions, various branching strategies are possible.

3.2.2 Branching strategy

Let $u \in V$ be a node of the branching tree T. Node u is also called a subproblem. If a value is assigned to a variable then the variable is called fixed. Let

$$\bar{I}_x^u := \{i \in I : \ \underline{x}_i \neq \overline{x}_i\}, \quad \bar{I}_y^u := \{i \in I : \ \underline{y}_i \neq \overline{y}_i\},$$

be the index sets of the unfixed variables from X and Y, respectively. Herewith, $I_x^u := I \backslash \bar{I}_x^u$ and $I_y^u := I \backslash \bar{I}_y^u$ are the index sets of fixed variables.

The original branching strategy [CJM08] is a two-step approach where the variables from X are fixed first, and the variables from Y are fixed next, see Algorithm 3.1. That means, if only $\bar{I}_x^u = \emptyset$ then we start to branch on the variables from Y.

An important issue is the selection of a branching variable from the unfixed ones, steps 1 and 2. It is based on the following observation. If we pack all the large items at first instead of packing all the items in an arbitrary order then we can rapidly obtain the inconsistency of the system (3.1)-(3.4), if any, because we do not lose much effort during the allocation of the small items.

As soon as a variable for branching is selected, it is fixed to the value which is equal to its lower bound (the first branch) or the lower bound for its domain is increased (the second branch), see step 4.

For describing the algorithm in a simpler way let us introduce the order of the variables, so $x_i \prec x_j \Leftrightarrow i < j$ and $y_i \prec y_j \Leftrightarrow i < j$ for $i, j = 1, \ldots, m$.

Algorithm 3.1 (FixMin)**.** *Creation of two descendants of a node $u \in V$ according to the original branching strategy [CJM08].*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1) *If $\bar{I}_x^u \neq \emptyset$ then set:*

$$\mathcal{P} := \{x_i \in X : \ i \in \bar{I}_x^u, \ w_i h_i = \max\{w_i h_i : \ i \in \bar{I}_x^u\}\},$$

and goto step 3.

(2) *If $\bar{I}_y^u \neq \emptyset$ then set:*

$$\mathcal{P} := \{y_i \in Y : \ i \in \bar{I}_y^u, \ w_i h_i = \max\{w_i h_i : \ i \in \bar{I}_y^u\}\},$$

and goto step 3, else goto Exit.

(3) *Select the variable with the smallest lower bound for its domain and then with the lowest index:*

$$p := \min_{\prec}\{p \in \operatorname{argmin}\{\underline{p} : \ p \in \mathcal{P}\}\}.$$

(4) *Definition of the descendant nodes:*

v_1: $\overline{p} := \underline{p}$ (fixation of the variable);
v_2: $\underline{p} := \underline{p} + 1$ (p is fixed later).

The following statement is true.

Lemma 3.1. *The depth of the branching tree by* FixMin *is* $O(m(W+H))$.

Proof. Each item is tried to be fixed in each feasible position in the $(0,x)$-direction, mW possible variants, and in the $(0,y)$-direction, mH possible variants. Since for each feasible position we have one branch then the depth of the branching tree is $O(m(W+H))$. □

3.3 Minor modifications

In this section we consider minor modifications of the original algorithm which are used either in each node of the branching tree as *raster points* and local preprocessing or only in the root node as an *initial preprocessing*. Let $E := (m, W, H, w, h)$ be an OPP-2 instance.

3.3.1 Raster points

It is often excessive to consider for a variable p each point from its domain $[\underline{p}, \overline{p}] \cap \mathbb{Z}$. For instance, if we have only three activities with the durations 4, 7, 9 then a schedule with the starting point 5 has never to be considered. The points which are of interest are called *raster points* [Sch08] and are calculated, e.g., for the $(0,x)$-direction as follows:

$$R_x(N) := \{0 \le x \le N : \ x = \sum_{i \in I} w_i a_i, \ a_i \in \{0,1\}, \ i \in I\}.$$

In the mentioned book, the author proposes an approach for a reduction of the number of raster points by the consideration of *reduced set* of raster points.

$$\widetilde{R}_x(N) := \{\max\{k \in R_x(N) : \ k \le N - r\} : \ r \in R_x(N)\}.$$

In order to obtain raster points which are situated to the left and right of a point α, let us consider the following definitions:

$$\underline{R}_x(\alpha, N) := \max\{\beta \in \widetilde{R}_x(N) : \ \beta \le \alpha\}, \quad \overline{R}_x(\alpha, N) := \min\{\beta \in \widetilde{R}_x(N) : \ \beta \ge \alpha\}$$

Similarly, we define $\underline{R}_y(\alpha, N)$ and $\overline{R}_y(\alpha, N)$ for the $(0,y)$-direction. Let designate $\underline{R}_x(\alpha) := \underline{R}_x(\alpha, W)$ and $\underline{R}_y(\beta) := \underline{R}_y(\beta, H)$.

Further we propose some branching strategies and pruning approaches which use raster points and a reduced set of raster points, see Sections 3.4.1, 3.4.2, 3.4.3, 3.4.8.

3.3.2 Initial preprocessing

The procedure of an *initial preprocessing* is performed only once for the root node. The idea of the procedure is to eliminate some symmetrical and equivalent solutions which satisfy the constraints (3.1)-(3.4). The results of the procedure are additional

constraints which are appended to the model (3.1)-(3.4) or replace some of its constraints. A similar procedure was proposed in [MMBS11] for the 1D contiguous bin packing problem (CBPP-1). For further restrictions refer to papers [BM03, CCM07].

In order to eliminate the solutions which are obtained through the symmetry over the vertical and horizontal lines going over the geometrical center of the container we apply:

$$x_{i^*} \leq \left\lfloor \frac{W - w_{i^*}}{2} \right\rfloor + 1, \quad y_{i^*} \leq \left\lfloor \frac{H - h_{i^*}}{2} \right\rfloor + 1, \quad i^* := \min\{i \in \mathcal{Q}\},$$

where $\mathcal{Q} := \{i \in I : \ w_i h_i = \max\{w_k h_k : \ k \in I\}\}$ is the set of the item indexes with the largest area. Herewith, according to the FixMin strategy the item i^* is fixed first, see steps 1 and 3 of Algorithm 3.1. This fact will be used in Section 3.5.3.

Let

$$F_x := \{(i,j) \in I \times I : \ i < j, \ w_i + w_j > W\},$$

$$F_y := \{(i,j) \in I \times I : \ i < j, \ h_i + h_j > H\}$$

be the sets of item pairs which do not fit together over the $(0,x)$-, and $(0,y)$-directions, respectively. Then instead of (3.1), the following constraints are applied:

$$x_i + w_i \leq x_j \vee x_j + w_j \leq x_i, \quad (i,j) \in F_y; \tag{3.10}$$

$$y_i + h_i \leq y_j \vee y_j + h_j \leq y_i, \quad (i,j) \in F_x; \tag{3.11}$$

$$\begin{aligned} &x_i + w_i \leq x_j \vee x_j + w_j \leq x_i \vee \\ &y_i + h_i \leq y_j \vee y_j + h_j \leq y_i, \quad (i,j) \in (I \times I) \backslash (F_x \cup F_y) : \ i < j. \end{aligned} \tag{3.12}$$

If two items i and j are identical, i.e., $w_i = w_j$ and $h_i = h_j$ then we apply the following constraints:

$$\begin{aligned} &x_i < x_j \vee \\ &x_i = x_j \wedge y_i + h_i \leq y_j, \quad (i,j) \in I \times I : \ i < j, \ w_i = w_j, \ h_i = h_j. \end{aligned}$$

Instead of (3.2) and (3.3), the following constraints are applied:

$$0 \leq x_i \leq \underline{R}_x(W - w_i), \quad i \in I; \tag{3.13}$$

$$0 \leq y_i \leq \underline{R}_y(H - h_i), \quad i \in I. \tag{3.14}$$

3.3.3 Local preprocessing

This type of the preprocessing is performed for each node of the branching tree. The main idea is to reduce the domain of the variables to a possibly small size. In terms of the constraint programming paradigm the local preprocessing is called *constraint propagation* [Apt03]. Since for the solution of our model we use ILOG CP (see Section 3.6 for further details), the local preprocessing is done automatically for each node of the branching tree after its creation.

3.4 New branching strategies

In this section we review some known and propose new strategies which are based on different principles. First group of the branching strategies fixes coordinates of the items or divides the domain of the variables. The other group branches on the mutual positions of the items, e.g., overlapping relations. The description of the new strategies uses notations from Sections 3.1, 3.2. For a numerical study of the strategies, refer to Section 3.6.

Strategies FixMin, FixMinR and Contour from this section fix a variable or increase the lower bound of its domain in each branch. In that case we will use the notation of a contour and a block-structure. The related but different notions of the contour were proposed in [Sch95] for 2D packing layouts. Let $u \in V$ be a node of the branching tree and $I_x^u(t) := \{i \in I_x^u : \ t \in [x_i, x_i + w_i)\}$ be the item set with the fixed x-coordinate whose x-projection intersects a given point $t \in [0, W)$.

Definition 3.1. *The X-contour C_x^u corresponding to a node $u \in V$ is the graph of the function:*

$$C_x^u(t) := \sum_{i \in I_x^u(t)} h_i, \quad t \in [0, W).$$

$C_x^u(t)$ is a step function with, in general, some discontinuity points in $(0, W)$. Let the ordered sequence $\{\chi_k^u\}_{k=1}^{q_x^u+1}$ contain exactly all the discontinuity points and the border values, such that $0 = \chi_1^u \leq \chi_2^u \leq \ldots \leq \chi_{q_x^u+1}^u = W$, i.e.,

$$\lim_{t \to \chi_k^x - 0} C_x^u(t) \neq \lim_{t \to \chi_k^x + 0} C_x^u(t) = C_x^u(\chi_k^x), \quad k \in \{2, \ldots, q_x^u\},$$

where $q_x^u + 1$ is the number of jump discontinuity points in $(0, W)$ plus two border points. For convenience, let

$$C_x^u(W) := \lim_{t \to W - 0} C_x^u(t).$$

Definition 3.2. *If $C(\chi_k^x) > C(\chi_{k+1}^x)$, $\forall k \in Q_x^u := \{1, \ldots, q_x^u\}$ then the X-contour is called monotonically decreasing or monotone.*

Remark 3.1. *Similarly, we define the Y-contour C_y^u corresponding to a node $u \in V$.*

Assigned to an interval $[\chi_k^u, \chi_{k+1}^u)$ with $k \in Q_x^u$ we define a vertical block as the rectangle $[\chi_k^u, \chi_{k+1}^u) \times [0, H)$ lying above C_x^u, see Figure 3.7.

Definition 3.3. *The k-th block corresponding to a contour C_x^u is the rectangle $[\chi_k^x, \chi_{k+1}^x) \times [C_x^u(\chi_k^u), H)$, denoted by $(\chi_k^x, \lambda_k^x, \rho_k^x)$ where $\lambda_k^x = H - C_x^u(\chi_k^x)$, and $\rho_k^x = \chi_{k+1}^x - \chi_k^x$.*

In terms of scheduling, the k-th block represents the non-used resource during the period $[\chi_k^x, \chi_{k+1}^x)$. Papers [BSM08, MMBS11] define a related notion of a slice describing the complete layout in the period $[\chi_k^x, \chi_{k+1}^x)$. Here a block is a part of a slice.

Definition 3.4. *The sequence of the vertical blocks* $\{(\chi_k^x, \lambda_k^x, \rho_k^x)\}_{k=1}^{q_x^u}$ *is called the vertical block-structure corresponding to a contour* C_x^u *and is denoted by* $S_{\|}(u)$.

Remark 3.2. *Similarly, we define the horizontal block-structure* $S_{=}(u)$ *consisting of the horizontal blocks* $\{(\chi_k^y, \lambda_k^y, \rho_k^y)\}_{k=1}^{q_x^u}$ *corresponding to a contour* C_y^u.

In the strategies DICHOTOMY and INTERVAL we change the lower and the upper bounds for the domain of a variable to branch. A principally different strategy is DISJUNCTIVE. In every branch we fix a mutual item position.

3.4.1 Original branching strategy with raster points

Here we try to heuristically reduce the large branching depth of the FIXMIN strategy caused by choosing individual values for the variables. For that purpose we bring the reduced set of raster points from Section 3.3.1 into FIXMIN.

In the proposed strategy we change step 4 of Algorithm 3.2 as follows. While a node $u \in V$ of the branching tree is processed, two descendant nodes are created where in the second descendant v_2 we set the lower bound for the variable domain to the next raster point, see Algorithm 3.2.

Let the type of a variable $p \in X \cup Y$ be determined in the following algorithms by

$$d(p) := \begin{cases} x, \text{ if } p \in X; \\ y, \text{ if } p \in Y. \end{cases}$$

Algorithm 3.2 (FIXMINR)**.** *Creation of two descendants of a node* $u \in V$ *according to the improved* FIXMIN.
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1-3) Same as in Algorithm 3.1.
(4) Definition of the descendant nodes:

v_1*: Same as in Algorithm 3.1;*
v_2*:* $\underline{p} := \overline{R}_{d(p)}(\underline{p} + 1)$ *(*p *is fixed later).*

The worst case depth of the branching tree by FIXMINR remains the same as by FIXMIN.

Remark 3.3. *The* X*-contour generated by* FIXMIN *and* FIXMINR *is, in general, non-monotonic.*

3.4.2 Contour branching strategy

Here we propose a branching strategy which is a modification of the contour concept [Sch95, MMV03] for the 1D case. This approach is a classical branching strategy, a modification of which was used in the algorithm [MMBS11] for the solution of CBPP-1.

Similarly to FixMin and FixMinR, the approach starts to fix the variables from X. After all variables from X are fixed, the approach omits the information concerning the fixed variables from X, and fixes the variables from Y, see Algorithm 3.3.

The strategy sequentially builds a contour. On each step we select the earliest block in which at least one item from the unfixed ones can be fixed. Further we decide which item is fixed within the block (one node for each choice). Additionally, we create a node in which we demand that none of the unfixed items is fixed within the selected block, see step 3 of Algorithm 3.3.

Algorithm 3.3 (Contour)**.** *Creation of descendants of a node $u \in V$ according to the contour principle.*
Input data: A node $u \in V$, block-structures $S^x_{\|}(u)$, $S^y_{\|}(u)$.
Output data: Descendant nodes $v_0, \ldots, v_n$.

(*1*) *If $\bar{I}^u_x \neq \emptyset$ then set:*

$$k^* := \operatorname{argmin}\{\chi^x_k : \ \exists i \in \bar{I}^u_x, \ h_i \leq \lambda^x_k\}, \quad \mathcal{P} := \{x_i \in X : \ h_i \leq \lambda^x_k\}.$$

goto step 3.

(*2*) *If $\bar{I}^u_y \neq \emptyset$ then set:*

$$k^* := \operatorname{argmin}\{\chi^y_k : \ \exists i \in \bar{I}^u_y, \ w_i \leq \lambda^y_k\}, \quad \mathcal{P} := \{y_i \in Y : \ w_i \leq \lambda^y_k\}.$$

goto step 3, else Exit.

(*3*) *Definition of the descendant nodes:*

v_0: $\forall p \in \mathcal{P}$ set $\underline{p} := \overline{R}_{d(p)}(\chi^{d(p)}_{k^*} + 1)$.

For $j = 1, \ldots, |\mathcal{P}|$:

v_j: $\overline{p} = \underline{p} := \chi^{d(p)}_{k^*}$.

Some papers [Sch95, MMBS11] discuss anti-symmetry rules for Contour in order to eliminate vertical and horizontal equivalent solutions, also with respect to identical items, etc. Here we do not use them.

Lemma 3.2. *The depth of the branching tree by* Contour *is $O(m)$.*

Proof. On each depth one item is fixed. Thus, the depth of the tree is $O(m)$. □

Remark 3.4. *The contour generated by* Contour *is monotone.*

3.4.3 Dichotomy

Here we propose a new branching strategy. The idea is to obtain a well balanced and a small branching tree. Now, instead of checking each coordinate value from the domain as by FixMin we try to divide the domain of a branching variable into two intervals.

First of all, on each step we select a variable to branch. Here we either select the variable as by FixMin or select the variable with the biggest domain over the unfixed ones which results in two different approaches, see step 3 of Algorithm 3.4. If a variable g is selected, two branches are created where g is restricted to the first half or to the second half of the interval $[\underline{g}, \overline{g}]$, see step 4.

Algorithm 3.4 (DICHOTOMY)**.** *Creation of two descendants of a node $u \in V$ according to the dichotomy principle.*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1-2) Same as in Algorithm 3.1.
(3) DICHOTOMY*(coordinate): Step 3 of Algorithm 3.1.*
DICHOTOMY*(interval): Select the variable with the biggest domain and then with the lowest index:*

$$p := \min_{\prec}\{p \in \operatorname{argmax}\{\overline{p} - \underline{p} : \ p \in \mathcal{P}\}\}.$$

(4) Definition of the descendant nodes:

$$v_1 : \ \overline{p} := \underline{R}_{d(p)}(\lfloor \frac{\underline{p} + \overline{p}}{2} \rfloor); \quad v_2 : \ \underline{p} := \overline{R}_{d(p)}(\lfloor \frac{\underline{p} + \overline{p}}{2} \rfloor + 1).$$

Lemma 3.3. *The depth of the branching tree by* DICHOTOMY *is $O(m(\log W + \log H))$.*

Proof. Each item is tried to be fixed in every interval after division in half. The depth of the branching subtree for an item $i \in I$ is equal to the number of divisions in the $(0, x)$-direction of the interval $\overline{x}_i - \underline{x}_i$ which is equal in the worst case to W, and H in the $(0, y)$-direction. Since the number of divisions of W is $\log W$ (for H is $\log H$) and these divisions are done for each item $i \in I$ then the depth is $O(m(\log W + \log H))$. □

3.4.4 Interval

In [SO08, Sim08] different strategies are discussed. The combination of the *naive* and the *xtheny* strategies from the paper yields the FIXMIN strategy. The most successful approach according to that paper is the *interval* strategy. It was firstly proposed in [BCDP11] for the perfect square packing problem.

The interval strategy consists in splitting of the domain of the variables from X into intervals then fixing the values for them, followed by splitting the domain of the variables from Y, and finally fixing the values for them. The splitting of the domain for a variable depends on the size of the corresponding item. It should be chosen small enough so that a mandatory part of the item occurs in each interval. In the original *interval* strategy, one third of the item size occurring obligatory in one of the split intervals, yields the best results [Sim08]. In comparison to *interval*, here we use DICHOTOMY when the intervals are too small to be split, see steps 1 and 2 of Algorithm 3.5.

Algorithm 3.5 (INTERVAL)**.** *Creation of descendants of a node $u \in V$ according to interval principle [Sim08].*
Input data: A node $u \in V$, a ratio r (here and further $r = 0.3$).
Output data: Descendant nodes $v_0, \ldots, v_n$.

(*1*) *If* $\bar{I}_x^u \neq \emptyset$ *then:*

$$\mathcal{Q} := \{i \in \bar{I}_x^u : \ \overline{x}_i - \underline{x}_i \geq \lceil rw_i \rceil\}, \quad \mathcal{P} := \{x_i \in X : \ i \in \mathcal{Q}, \ w_i h_i = \max_{i \in \mathcal{Q}}\{w_i h_i\}\}$$

else goto 2. If $\mathcal{Q} \neq \emptyset$ *then goto step 3, else* DICHOTOMY.

(*2*) *If* $\bar{I}_y^u \neq \emptyset$ *then:*

$$\mathcal{Q} := \{i \in \bar{I}_y^u : \ \overline{y}_i - \underline{y}_i \geq \lceil rh_i \rceil\}, \quad \mathcal{P} := \{y_i \in Y : \ i \in \mathcal{Q}, \ w_i h_i = \max_{i \in \mathcal{Q}}\{w_i h_i\}\}$$

else goto Exit. *If* $\mathcal{Q} \neq \emptyset$ *then goto step 3, else* DICHOTOMY.

(*3*) *Select the variable with the greatest domain and then with the lowest index:*

$$p := \min_{\prec}\{p \in \operatorname{argmax}\{\overline{p} - \underline{p} : \ p \in P\}\}.$$

(*4*) *Definition of the descendant nodes:*
If $d(p) = x$ *then set* $s := \lceil rw_i \rceil$ *with* $i \in \bar{I}_x^u \wedge x_i = p$, *else set* $s := \lceil rh_i \rceil$ *with* $i \in \bar{I}_y^u \wedge y_i = p$. *For* $j = 1, \ldots, \lceil \frac{\overline{p} - \underline{p}}{s} \rceil$ *create:*

$$v_j : \ \underline{p} := \underline{p} + (j-1)s; \ \overline{p} := \min\{\underline{p} + js - 1, \overline{p}\}.$$

Lemma 3.4. *The depth of the branching tree by* INTERVAL *is* $O(m + \sum_{i=1}^{m}(\log rw_i + \log rh_i))$.

Proof. In the worst case we divide the interval for each item, $O(m)$, and then fix the items within the intervals according to DICHOTOMY which are equal in this case to rw_i and rh_i, respectively, hence $O(m + \sum_{i=1}^{m}(\log rw_i + \log rh_i))$. □

3.4.5 Disjunctive

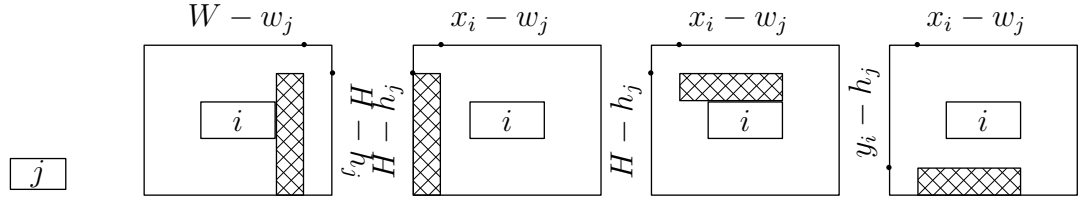
This strategy is based neither on the coordinate interval division nor on the assignment of coordinate variables as before. It is principally another strategy which iteratively fixes the set of all possible mutual positions for a pair of items.

Let $l, r, b, t \in \mathbb{N}$ be such $l < r < b < t$ and $R_{ij} \subseteq \{l, r, b, t\}$ be the set which represents the possible mutual locations of items $i, j \in I$ with $i < j$, i.e.,

$$R_{ij} = \{l\} \Leftrightarrow x_i + w_i \leq x_j, \qquad R_{ij} = \{r\} \Leftrightarrow x_j + w_j \leq x_i,$$

$$R_{ij} = \{b\} \Leftrightarrow \begin{cases} x_i + w_i > x_j, \\ x_j + w_j > x_i, \\ y_i + h_i \leq y_j, \end{cases} \qquad R_{ij} = \{t\} \Leftrightarrow \begin{cases} x_i + w_i > x_j, \\ x_j + w_j > x_i, \\ y_j + h_j \leq y_i. \end{cases}$$

In Figure 3.1a we see the dimensions of an item j. Figures 3.1b-3.1e show by the hatched areas the feasible locations of the left-bottom corner of the item j for $R_{ij} = l, r, b, t$, respectively. Now, we associate with each node $u \in V$ the set $R(u) := \{R_{ij} : \ i < j, \ i, j \in I\}$ of relations for the item pairs.



(a) Item j (b) $R_{ij} = \{l\}$ (c) $R_{ij} = \{r\}$ (d) $R_{ij} = \{b\}$ (e) $R_{ij} = \{t\}$

Figure 3.1: Mutual locations of the items i and j. The areas where the left-bottom arrangement point of the item j can be allocated are marked by hatching.

In order to prioritize item pairs selected for branching let us introduce a pair ordering denoted by the symbol $\sqsupset$. Suppose $(p, q), (r, k) \in I \times I$: $p < q$ and $r < k$. So, $(p, q) \sqsupset (r, k)$, iff $w_p h_p > w_r h_r \vee (w_p h_p = w_r h_r \wedge w_q h_q > w_k h_k) \vee (w_p h_p = w_r h_r \wedge w_q h_q = w_k h_k \wedge p < r)$.

The main idea of the disjunctive strategy [Sim08] is to fix a relation for every item pair, see Algorithm 3.6. After a pair is selected, see steps 1 and 2, we select a relation to fix for this pair, step 3. At last, when every item pair got a relation, a packing layout (item coordinates) can be computed. If the items lie within the container bounds then the layout is feasible.

Algorithm 3.6 (DISJUNCTIVE)**.** *Creation of two descendants of a node $u \in V$ according to the disjunctive principle.*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(*1*) *If* $\exists R_{ij} \in R(u) : \ |R_{ij}| > 1$ *then set:*

$$\mathcal{P} := \{(i, j) \in I \times I : \ i < j, \ |R_{ij}| > 1\},$$

else goto Exit.

(*2*) *Select the pair to branch:*

$$(i, j) := \min_{\sqsupset}\{(i, j) : \ (i, j) \in \mathcal{P}\}.$$

(*3*) *Definition of the descendant nodes:*
Set $f := \min\{k : \ k \in R_{ij}, \ R_{ij} \in R(u)\}$ *(note that we assume the ordering* $l < r < b < t$*). Create:*

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{f\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = R_{ij} \backslash \{f\}.$$

Once we have fixed all mutual locations of the items, the domain of variables from X, Y can still contain not just a single value. It depends on the quality of the local preprocessing procedure, see Section 3.3.3. Note that the local preprocessing procedure is preformed for every node after its creation. Several values of a variable from X, Y can also be feasible for the model. In our case after all relations are fixed we do not branch variables X, Y to a single value.

The following statement is true.

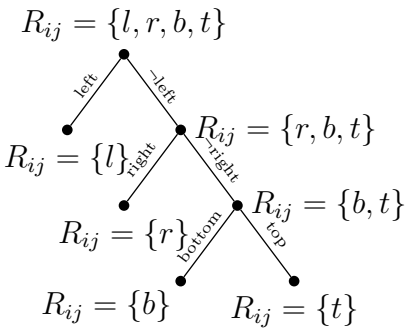


Figure 3.2: Binary branching tree corresponding to DISJUNCTIVE.

Theorem 3.5. *If* $\forall i, j \in I : \ i < j, \ |R_{ij}| = 1$ *and the system* (3.1)-(3.4) *holds then the corresponding OPP-2 instance is feasible.*

Lemma 3.6. *The depth of the branching tree by* DISJUNCTIVE *is* $O(m^2)$.

Proof. On each depth of the branching tree we fix one of the constant number of relations for a pair of items. Herewith, the depth of the tree is not larger than the number of pairs, $O(m^2)$. □

3.4.6 Partition

As we can see at the DISJUNCTIVE strategy the decision that the projections of two items are partitioned over the $(0, x)$-, or over the $(0, y)$-direction is made only in the depth of the branches. In some cases it would be better to assert the disjoint state of the items over one of the axis at first steps. The decision that projections of the items are disjoint can help, e.g., to strength some of the pruning rules as, i.e., LP-pair (for the pruning rules refer to Section 3.5).

Let $p_x, p_y, l, r, b, t \in \mathbb{N}$ and $R_{ij} \subseteq \{p_x, p_y, l, r, b, t\}$ be the set which represents the possible mutual locations of items $i, j \in I$ with $i < j$, i.e.,

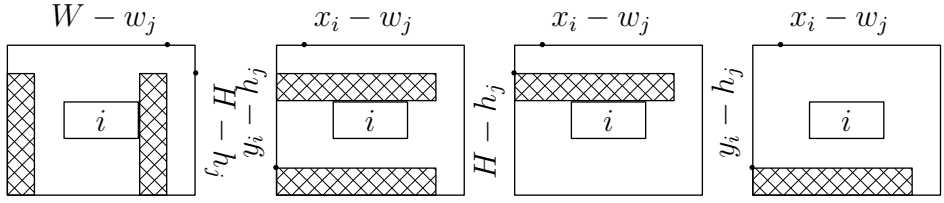
$$R_{ij} = \{p_x\} \Leftrightarrow \left[\begin{array}{l} x_i + w_i \leq x_j, \\ x_j + w_j \leq x_i, \end{array} \right. \qquad R_{ij} = \{p_y\} \Leftrightarrow \left[\begin{array}{l} y_i + h_i \leq y_j, \\ y_j + h_j \leq y_i, \end{array} \right.$$

$$R_{ij} = \{l\} \Leftrightarrow x_i + w_i \leq x_j, \qquad R_{ij} = \{r\} \Leftrightarrow x_j + w_j \leq x_i,$$

$$R_{ij} = \{b\} \Leftrightarrow y_i + h_i \leq y_j, \qquad R_{ij} = \{t\} \Leftrightarrow y_j + h_j \leq y_i.$$

In Figure 3.3 we show by the hatched areas the feasible locations of the left-bottom arrangement point of the item j for $R_{ij} = p_x, p_y, l, r, b, t$, respectively. Now we associate with each node $u \in V$ the set $R(u) := \{R_{ij} : \ i < j, \ i, j \in I\}$ of relations for the item pairs. The item pairs have an ordering relation defined by $\sqsupset$ as in Section 3.4.5.

Let $\mathcal{R} := \{R \subseteq \{p_x, p_y, l, r, b, t\} : \ |R| > 1 \vee R = \{p_x\} \vee R = \{p_y\}\}$ be the set of all item sets with no or $p_x - p_y$ fixed relations. The main idea of the partition strategy is to decide at first steps that projections of two items without a fixed relation are disjoint over one of the axis, see Algorithm 3.7. After a pair is selected, see steps 1 and 2, we assert that the projections of the item pair do not overlap over the $(0, x)$-direction,



(a) $R_{ij} = \{p_x\}$ (b) $R_{ij} = \{p_y\}$ (c) $R_{ij} = \{b\}$ (d) $R_{ij} = \{t\}$

Figure 3.3: Possible mutual locations of the items i and j according to PARTITION. The areas where the left-bottom arrangement point of the item j can be allocated are marked by hatching. The size of the item j is depicted in Figure 3.1a. $R_{ij} = \{l\}$ and $R_{ij} = \{r\}$ are depicted in Figure 3.1b and 3.1c.

the first branch, or over the $(0, y)$-direction, the second branch. When this decision is made, the further partition is followed in the leafs, see step 3.

Algorithm 3.7 (PARTITION)**.** *Creation of two descendants of a node $u \in V$ according to the partition principle.*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1) If $\exists R_{ij} \in R(u) : \ R_{ij} \in \mathcal{R}$ then set:

$$\mathcal{P} := \{(i,j) \in I \times I : \ i < j, \ R_{ij} \in \mathcal{R}\},$$

else goto Exit.

(2) Selection of the pair to branch:

$$(i,j) := \min_{\sqsupset}\{(i,j) \in \mathcal{P}\}.$$

(3) Definition of the descendant nodes:
If $\{p_x, p_y\} \subset R_{ij}$ then

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{p_x\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{p_y\}.$$

If $R_{ij} = \{p_x\}$ then:

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{l\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{r\}.$$

If $R_{ij} = \{p_y\}$ then:

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{b\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{t\}.$$

For the binary branching tree of the PARTITION refer to Figure 3.4. The following statement is true.

Theorem 3.7. *If $\forall i, j \in I : \ i < j, \ |R_{ij}| = 1$ and the system (3.1)-(3.4) holds then the corresponding OPP-2 instance is feasible.*

Lemma 3.8. *The depth of the branching tree by* PARTITION *is $O(m^2)$.*

Proof. We fix at most six relations for every pair which make in sum $6m^2$ branches. Herewith the depth of the tree is $O(m^2)$. □

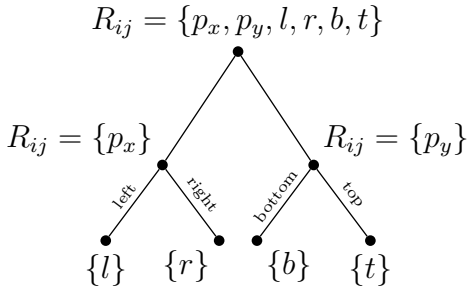


Figure 3.4: Binary branching tree corresponding to PARTITION.

3.4.7 Intersection

In contrast to PARTITION here we propose a branching strategy where we decide whether an item pair projection intersects one of the axis.

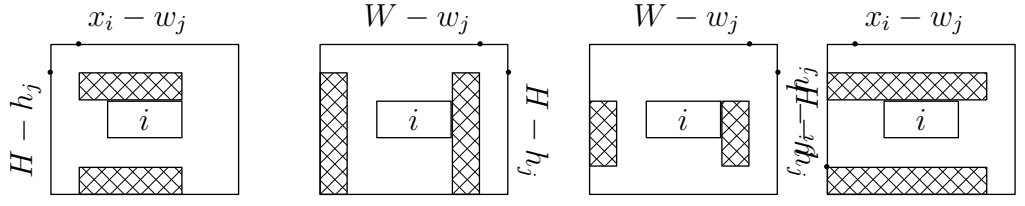
Let $i_x, \bar{i}_x, i_y, \bar{i}_y, l, r, b, t \in \mathbb{N}$ and $R_{ij} \subseteq \{i_x, \bar{i}_x, i_y, \bar{i}_y, l, r, b, t\}$ be the set which represents the possible mutual locations of items $i, j \in I$ with $i < j$, i.e.,

$$R_{ij} = \{i_x\} \Leftrightarrow \begin{cases} x_i < x_j + w_j, \\ x_j < x_i + w_i. \end{cases} \qquad R_{ij} = \{\bar{i}_x\} \Leftrightarrow \begin{bmatrix} x_i + w_i \leq x_j, \\ x_j + w_j \leq x_i, \end{bmatrix}$$

$$R_{ij} = \{i_y\} \Leftrightarrow \begin{cases} y_i < y_j + h_j, \\ y_j < y_i + h_i. \end{cases} \qquad R_{ij} = \{\bar{i}_y\} \Leftrightarrow \begin{bmatrix} y_i + h_i \leq y_j, \\ y_j + h_j \leq y_i, \end{bmatrix}$$

The other mutual positions of the items corresponding to $R_{ij} = \{l\}, \{r\}, \{b\}, \{t\}$ is the same as in PARTITION, see Section 3.4.6.

Figure 3.5 shows by the hatched areas the feasible locations of the left-bottom arrangement point of the item j for $R_{ij} = \{i_x\}, \{\bar{i}_x\}, \{i_y\}, \{\bar{i}_y\}$, respectively. For the other values of R_{ij} refer to Figure 3.3. Now we associate with each node $u \in V$ the set $R(u) = \{R_{ij} : \ i < j, \ i, j \in I\}$ of relations for the item pairs. The item pairs have an ordering relation defined by $\sqsupset$ as in Section 3.4.5.



(a) $R_{ij} = \{i_x\}$ (b) $R_{ij} = \{\bar{i}_x\}$ (c) $R_{ij} = \{i_y\}$ (d) $R_{ij} = \{\bar{i}_y\}$

Figure 3.5: Possible mutual locations of items i and j according to INTERSECTION. The areas where the left-bottom arrangement point of item j can be allocated are marked by hatching. Item j size is depicted in Figure 3.1a. $R_{ij} = \{l\}$, $R_{ij} = \{r\}$ are depicted in Figure 3.1b and 3.1c, $R_{ij} = \{b\}$, $R_{ij} = \{t\}$ are depicted in Figure 3.3c and 3.3d

.

Let

$$\mathcal{R} = \{R \subseteq \{i_x, \bar{i}_x, i_y, \bar{i}_y, l, r, b, t\} : \ |R| > 1 \vee R = \{i_x\} \vee R = \{\bar{i}_x\} \vee R = \{i_y\} \vee R = \{\bar{i}_y\}\}$$

be the set of all item sets with no or one of i_x, $\bar{i}_x$, i_y or $\bar{i}_y$ fixed relations. The main idea of the partition strategy is to decide at first steps that projections of two items without fixed relations overlap over one of the axis, see Algorithm 3.8.

Algorithm 3.8 (INTERSECTION)**.** *Creation of two descendants of a node $u \in V$ according to the partition principle.*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1) If $\exists R_{ij} \in R(u) : \ R_{ij} \in \mathcal{R}$ then set:

$$\mathcal{P} := \{(i,j) \in I \times I : \ i < j, \ R_{ij} \in \mathcal{R}\},$$

else goto Exit.

(2) Selection of the pair to branch:

$$(i,j) := \min_{\sqsupset}\{(i,j) : \ (i,j) \in \mathcal{P}\}.$$

(3) Definition of the descendant nodes:
If $\{i_x, \bar{i}_x\} \subset R_{ij}$ then:

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{i_x\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{\bar{i}_x\}.$$

If $R_{ij} = \{i_x\}$ then:

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{b\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{t\}.$$

If $R_{ij} = \{\bar{i}_x\}$ then:

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{i_y\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{\bar{i}_y\}.$$

If $R_{ij} = \{i_y\}$ then:

$$v_1 : \ R(v_1) := R(u) \wedge R_{ij} = \{l\}; \quad v_2 : \ R(v_2) := R(u) \wedge R_{ij} = \{r\}.$$

For the binary branching tree of the INTERSECTION refer to Figure 3.6. The following statement is true.

Theorem 3.9. *If $\forall i,j \in I : \ i < j \wedge |R_{ij}| = 1$ and the system (3.1)-(3.4) holds then the corresponding OPP-2 instance is feasible.*

Lemma 3.10. *The depth of the branching tree by* PARTITION *is $O(m^2)$.*

Proof. We fix at most six relations for every pair which make in sum $6m^2$ branches. Herewith the depth of the tree is $O(m^2)$. □

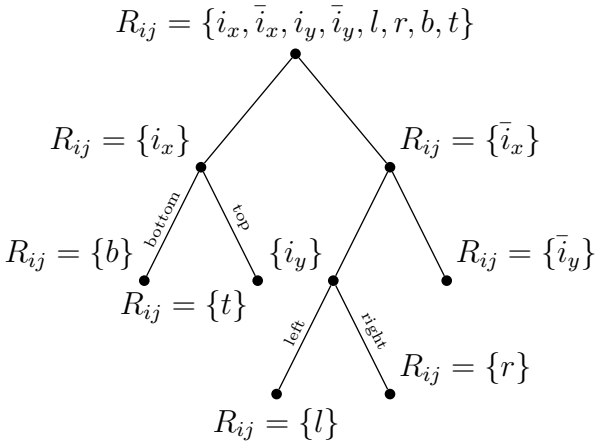


Figure 3.6: Binary branching tree corresponding to INTERSECTION.

3.4.8 Solution search of SPP-2

An SPP-2 instance is solved using the corresponding OPP-2 instances. We fix the width of the OPP-2 container and vary its height. The OPP-2 instance with the container height equal to a known upper bound is feasible. Once an OPP-2 instance with the container height h is feasible and the OPP-2 instance with the next possible smaller container height is not then h is the optimal value.

Since DISJUNCTIVE brings very good results for feasible instances, see Section 3.6.1 and 3.6.2 then it is better to have a strategy which searches for an optimum from an upper bound downwards. So each of instances (W, γ, m, w, h) with $\gamma = \overline{H}, \ldots, H_{opt}$ is feasible except the last one with the height equal to $\max\{r \in R_y(\overline{H}) : \ r \leq H_{opt} - 1\}$ which is infeasible, see Algorithm 3.9.

Algorithm 3.9 (OPTDOWNWARDS)**.** *Solution search of SPP-2 based on OPP-2 downwards from an upper bound.*
Input data: An SPP-2 instance (W, m, w, h)*, upper bound* $\overline{H}$*.*
Output data: optimal value H_{opt}*.*

(*1*) *Set* $\beta := \overline{H}$*.*
(*2*) *Set* $\gamma := \max\{r \in R_y(\overline{H}) : \ r \leq \beta - 1\}$*. If the OPP-2 instance* (W, γ, m, w, h) *is feasible then set* $\beta := \gamma$ *and goto 2, else set* $H_{opt} := \beta$ *and goto* Exit*.*

For the other branching strategies which work equally for the feasible and unfeasible OPP-2 instances as well, we apply the dichotomy principle in order to increase the efficiency of the search. The idea is similar to that from Section 3.4.5.

The approach works as follows. We have the interval from a lower to an upper bound for the optimal height of the container. We divide this interval in two and obtain a new value of the height by that. Further we test whether the items fit into the container with the new height. If so then the value of the current height of the container is the new upper bound, otherwise the new lower bound. The process is performed until the difference between the lower and the upper bounds becomes 1, see

Algorithm 3.10. For the solution of the OPP-2 instances (W, γ, m, w, h) we can use one of the described approaches from Section 3.4.

Algorithm 3.10 (OptDichotomy)**.** *Solution search of SPP-2 based on OPP-2 using dichotomy principle.*
Input data: An SPP-2 instance (W, m, w, h), a lower $\underline{H}$ and an upper $\overline{H}$ bounds.
Output data: optimal value H_{opt}.

(*1*) *Set $\alpha := \underline{H} - 1$, $\beta := \overline{H}$.*
(*2*) *If $\beta - \alpha = 1$ then $H_{opt} := \beta$ and goto Exit. Set $\gamma := \max\{r \in R_y(\overline{H}) : r \leq \lceil \frac{\alpha+\beta}{2} \rceil\}$. If the OPP-2 instance (W, γ, m, w, h) is feasible then set $\beta := \gamma$, else $\alpha := \gamma$, and goto 2.*

Remark 3.5. *If we stop the optimization process considering, i.e., the time limit, then the value of β will be an improved upper bound.*

3.5 Advanced constraint propagation

In this section we consider different LP-based approaches for the nodes of the branching tree in order to reduce the search process. Every LP-based approach constructs a relaxation by the 1D bin packing problem and then its Dantzig-Wolfe decomposition.

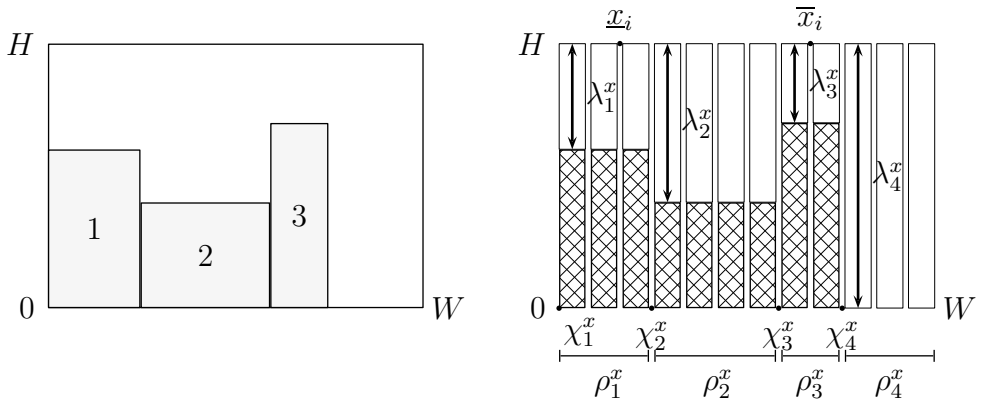
One group of the approaches from this section, LP1a and LP1b, uses only information concerning the x-coordinates of the fixed items from I_x^u. The second group, LP-int, is tightened by the information concerning the domain of the coordinate variables which results from the constraint propagation procedure. The third group, LP1c, LP-pair, LP-int-pair, uses the information concerning the overlapping of item projections on the axes. The fourth group, LP-advanced, improves the first group through extra connecting constraints.

3.5.1 Vertical bar relaxation (LP1a)

Here we propose a pruning rule which uses the information concerning the coordinates of the fixed items. Herewith, it can be used especially with the branching strategies which produce a contour, i.e., FixMin, FixMinR, Contour.

Let us consider the partial solution in Figure 3.7 which corresponds to a node $u \in V$. Items from I_x^u with the fixed x-coordinates give us the information concerning the X-contour. Based on the X-contour we build the corresponding block-structure $S_{\|}^x(u)$ which was defined in Section 3.4. The items $\bar{I}_x^u$ are not fixed but we know the intervals in which they can be fixed. For the LP1a approach we omit the latter information.

The input information for the 1D bar relaxation is constructed as follows. Each block $(\chi_k^x, \lambda_k^x, \rho_k^x)$ with $k \in Q_x^u$ is considered as a set of bins with length λ_k^x and quantity ρ_k^x. In addition to the obtained bins, we also consider one extra type 0 of bins with $\chi_0^x := W$, $\lambda_0^x := H$, $\rho_0^x := \infty$. In the 2D case each 2D item has a certain geometrical location. Here we relax this condition and consider instead of 2D items 1D items with lengths h_i and quantities w_i where $i \in \bar{I}_x^u$ and I_x^u is the index set of the unfixed items.



(a) Feasible 2D packing (b) X-contour

Figure 3.7: Vertical bar relaxation of the partial solution in the $(0, x)$-direction corresponding to a node $u \in V$. Items from $I_x^u = \{1, 2, 3\}$ are marked by hatching. There exist blocks $\{(\chi_k^x, \lambda_k^x, \rho_k^x)\}$ with $k = 1, \ldots, 4$.

The 1D vertical bar relaxation can be described as follows. In order to describe a packing of the bins with the obtained items let us introduce vertical packing patterns in the following manner. For each 1D bin of type $k \in Q_x^u \cup \{0\}$ let J_k^x denote the index set of all binary vectors $a^{jk} := (a_1^{jk}, \ldots, a_m^{jk}) \in \{0, 1\}^m$ with

$$a_i^{jk} = 0, \quad \forall i \in I_x^u, \quad \sum_{i \in \bar{I}_x^u} h_i a_i^{jk} \leq \lambda_k^x, \quad j \in J_k^x, \tag{3.15}$$

where the i-th component a_i^{jk} of the vector a^{jk} in the case of $a_i^{jk} = 1$ indicates the j-th pattern of type k which contains one 1D item of type $i \in I$. Whether item i's x-coordinate is fixed, is indicated in the following model by

$$\delta_i = \begin{cases} 0, \text{ if } i \in I_x^u; \\ 1, \text{ if } i \in \bar{I}_x^u. \end{cases}$$

The main idea of the approach consists in minimizing the number of used bins of type 0. If at least a small part of that type of the bins is used then there exists no packing of the residual items $\bar{I}_x^u$ which fit into the container.

Let us formulate the following continuous relaxation of the set-partitioning model [KZ51, GG61, GG63] of the 1D multiple-capacity bin packing problem[1] (MCBPP-1) on vectors (3.15) and variables x^{jk} with $j \in J_k^x$ and $k \in Q_x^u \cup \{0\}$ which indicate the

[1]Usually, the 1D multiple stock size cutting stock problem (MSSCSP-1) [WHS07, AV08] is considered.

intensity of usage of vertical packing patterns.

$$z_a^* = \min \sum_{j \in J_0^x} x^{j,0}, \quad \text{s.t.} \tag{3.16}$$

$$\sum_{k=0}^{q_x^u} \sum_{j \in J_k^x} a_i^{jk} x^{jk} = w_i \delta_i, \quad i \in I; \tag{3.17}$$

$$\sum_{j \in J_k^x} x^{jk} \leq \rho_k^x, \quad k \in Q_x^u; \tag{3.18}$$

$$x^{jk} \geq 0, \quad k \in Q_x^u \cup \{0\},\ j \in J_k^x. \tag{3.19}$$

This problem is called the vertical the bar relaxation. In the subsequent sections we define several other types of bar relaxation.

The formulation (3.16)-(3.19) is an LP problem which is solved by the column generation method [KZ51, GG63, GG63, AV08]. The solution process is started from a initial set of variables (columns) which contains m variables with a large coefficient in the objective function for each constraint (3.17), and initial dual simplex multipliers $d := (d_1, \ldots, d_{m+q_x^u})$ which are obtained by the execution of the simplex method on the initial set of variables.

The restricted master problem of (3.16)-(3.19) contains variable pools for each type of columns. Each iteration consists of the generation of a column (slave problem) for each pool, its addition into the corresponding pool, and execution of the simplex method on the restricted master problem.

The generation of a column is aimed to maximize the sum of the dual simplex multipliers which is done by the solution of the following 0-1 linear programs:

$$\bar{c}_0^x = 1 - \max\{\sum_{i \in \bar{I}_x^u} d_i a_i : \sum_{i \in \bar{I}_x^u} h_i a_i \leq \lambda_0^x,\ a_i \in \{0,1\}\}; \tag{3.20}$$

$$\bar{c}_k^x = -\max\{d_{m+k} + \sum_{i \in \bar{I}_x^u} d_i a_i : \sum_{i \in \bar{I}_x^u} h_i a_i \leq \lambda_k^x,\ a_i \in \{0,1\}\}, \quad k \in Q_x^u. \tag{3.21}$$

Thus, on each step $q_x^u + 1$ slave 0-1 linear programs are solved and the column with $\operatorname{argmin}\{\bar{c}_0^x, \bar{c}_k^x : k \in Q_x^u\}$ is considered as a candidate to be appended to the corresponding pool. The variables which correspond to type 0 of bins have the coefficient in the objective function equal to 1, in contrast to the obtained ones. The column generation process is performed as long as there exists a column which can improve the value of the objective function. That means $-\bar{c}_k^x > \epsilon$ for some $k \in Q_x^u \cup \{0\}$ where $\epsilon > 0$ is a small enough constant.

Remark 3.6. *If problem* (3.21) *is solved by the dynamic programming method, the obtained dynamic programming table (DPT) can be used to get the solution of* (3.20) *without construction of a second DPT. So, problems* (3.20), (3.21) *can be solved at once.*

For the coefficient matrix of the formulation (3.16)-(3.19) refer to Table 3.1. There exist $q_x^u + 1$ pools of variables.

Table 3.1: Coefficient matrix of the formulation (3.16)-(3.19).

$k=1$	$k=2$	$\cdots$	$k=q_x^u$	$k=0$	
$x^{11} \cdots x^{\lvert J_1^x\rvert,1}$	$x^{12} \cdots x^{\lvert J_2^x\rvert,2}$	$\cdots$	$x^{1,q_x^u} \cdots x^{\lvert J^x_{q_x^u}\rvert,q_x^u}$	$x^{1,0} \cdots x^{\lvert J_0^x\rvert,0}$	
$a_1^{11} \cdots a_1^{\lvert J_1^x\rvert,1}$	$a_1^{12} \cdots a_1^{\lvert J_2^x\rvert,2}$		$a_1^{1,q_x^u} \cdots a_1^{\lvert J^x_{q_x^u}\rvert,q_x^u}$	$a_1^{1,0} \cdots a_1^{\lvert J_0^x\rvert,0}$	$= w_1\delta_1$
$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\cdots$	$\vdots \ddots \vdots$	$\vdots \ddots \vdots$	$\vdots$
$a_m^{11} \cdots a_m^{\lvert J_1^x\rvert,1}$	$a_m^{12} \cdots a_m^{\lvert J_2^x\rvert,2}$		$a_m^{1,q_x^u} \cdots a_m^{\lvert J^x_{q_x^u}\rvert,q_x^u}$	$a_m^{1,0} \cdots a_m^{\lvert J_0^x\rvert,0}$	$= w_m\delta_m$
$1 \cdots 1$					$\leq \rho_1^x$
	$1 \cdots 1$				$\leq \rho_2^x$
		$\ddots$			$\vdots$
			$1 \cdots 1$		$\leq \rho^x_{q_x^u}$

3.5.2 Horizontal bar relaxation with a monotone contour (LP1b)

Similarly to LP1a, we propose here a pruning rule which uses the information concerning the fixed items which build a monotone X-contour. Herewith, it can be used with the CONTOUR branching strategy.

Suppose we have a monotone X-contour. Assigned to an interval $[\chi_k^x, \chi_{k+1}^x)$ with $k = 1, \ldots, q_u^x - 1$, we define a horizontal block as the rectangle $[0, W) \times [H - \lambda_k^x, H - \lambda_{k-1}^x)$, see Figure 3.8.

Definition 3.5. *The k-th horizontal block corresponding to a contour C_x^u is the rectangle $[\lambda_k^x, W) \times [H - \lambda_k^x, H - \lambda_{k-1})$, denoted by $(\chi_k^x, \omega_k^x, \sigma_k^x)$ where $\omega_k^x := W - \chi_k^x$, and $\sigma_k^x := \lambda_k^x - \lambda_{k-1}^x$ if $k > 1$, and $\sigma_1^x := \lambda_1^x$ if $k = 1$.*

Definition 3.6. *The sequence of horizontal blocks $\{(\chi_k^x, \omega_k^x, \sigma_k^x)\}_{k=1}^{q_x^u}$ is called the horizontal block-structure corresponding to a contour C_x^u and denoted by $S_=^x(u)$.*

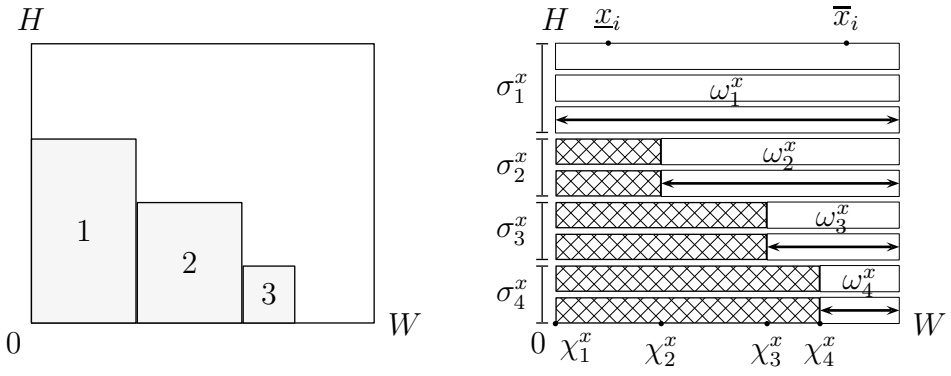
The input information for the 1D bar relaxation is constructed as follows. Each obtained block $(\chi_k^x, \omega_k^x, \sigma_k^x)$ with $k \in Q_x^u$ is considered as a set of bins with length ω_k^x and quantity σ_k^x. In addition to the obtained bins, we also consider one extra type 0 of bins with $\chi_0^x := 0$, $\omega_0^x := W$, $\sigma_0^x := \infty$. As it was done before in Section 3.5.1 here we consider instead of 2D items 1D items with lengths w_i and quantities h_i where $i \in \bar{I}_x^u$ is the set of the indexes of the unfixed items.

The 1D horizontal bar relaxation can be described as follows. In order to describe a packing of bins with obtained items let us introduce horizontal packing patterns in the following manner. For each horizontal bin of type $k \in Q_x^u \cup \{0\}$ let J_k^y denote the index set of all binary vectors $b^{jk} := (b_1^{jk}, \ldots, b_m^{jk}) \in \{0,1\}^m$ with

$$b_i^{jk} = 0, \quad \forall i \in I_x^u, \quad \sum_{i \in \bar{I}_x^u} w_i b_i^{jk} \leq \omega_k^x, \quad j \in J_k^y, \tag{3.22}$$

where i-th component b_i^{jk} of the vector b^{jk} in the case of $b_i^{jk} = 1$ indicates the j-th pattern of type k which contains one 1D items of type $i \in I$.

The similar idea to that from Section 3.5.1 underlies the 1D horizontal bar relaxation. Let us formulate the following continuous relaxation of the set-partitioning



(a) Feasible 2D packing (b) Monotone X-contour

Figure 3.8: Horizontal bar relaxation of the partial solution in the $(0, x)$-direction corresponding to a node $u \in V$ with a monotone contour. Items from $I_x^u = \{1, 2, 3\}$ are marked by hatching. There exist blocks $\{(\chi_k^x, \omega_k^x, \sigma_k^x)\}$ with $k = 1, \ldots, 4$.

model of MCBPP-1 on vectors (3.22) and variables y^{jk} with $j \in J_k^y$ and $k \in Q_x^u \cup \{0\}$ which indicate the intensity of usage of horizontal packing patterns:

$$z_b^* = \min \sum_{j \in J_0^y} y^{j,0}, \quad \text{s.t.} \tag{3.23}$$

$$\sum_{k=0}^{q_x^u} \sum_{j \in J_k^y} b_i^{jk} y^{jk} = h_i \delta_i, \quad i \in I; \tag{3.24}$$

$$\sum_{j \in J_k^y} y^{jk} \leq \sigma_k^x, \quad k \in Q_x^u; \tag{3.25}$$

$$y^{jk} \geq 0, \quad k \in Q_x^u \cup \{0\},\ j \in J_k^y. \tag{3.26}$$

This problem is called the horizontal bar relaxation. It is an LP problem which is solved by the column generation method. In order to solve the relaxation problem the following slave 0-1 linear programs are solved:

$$\bar{c}_0^y = 1 - \max\{\sum_{i \in \bar{I}_x^u} d_i a_i : \sum_{i \in \bar{I}_x^u} w_i a_i \leq \omega_0^x \wedge a_i \in \{0, 1\}\}; \tag{3.27}$$

$$\bar{c}_k^y = -\max\{d_{m+k} + \sum_{i \in \bar{I}_x^u} d_i a_i : \sum_{i \in \bar{I}_x^u} w_i a_i \leq \omega_k^x \wedge a_i \in \{0, 1\}\}, \quad k \in Q_x^u, \tag{3.28}$$

where $d := (d_1, \ldots, d_{m+q_x^u})$ is the vector of the simplex multipliers. Thus, on each step $q_x^u + 1$ slave 0-1 linear programs are solved and the column with $\operatorname{argmin}\{\bar{c}_0^y, \bar{c}_k^y : k \in Q_x^u\}$ is considered as a candidate to be appended to the corresponding pool of variables. Remark 3.6 is also valid here.

The following statement is true.

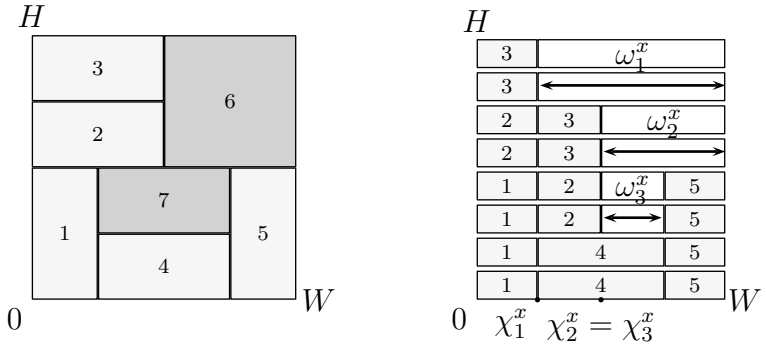
Lemma 3.11. *If $z_a^* + z_b^* > \epsilon$, $\epsilon > 0$ then there exists no feasible packing of items $\bar{I}_x^u$ into container (W, H) with fixed items I_x^u.*

Proof. Let us assume the opposite, $z_a^* > \epsilon$ and there exists a feasible packing. The latter means that $\sum_{i \in J_{q_0^u}^x} a_i^{j,q_0^u} x^{j,q_0^u} > 0$ for an $i \in I$. Since the left side of the expression is positive then from (3.17) follows that $w_i \delta_i - \sum_{k=1}^{q_x^u} \sum_{j \in J_k^x} a_i^{jk} x^{jk} > 0$ for an $i \in I$ which means that not all items are packed into the bins obtained from the container through the bar relaxation. Hence, the packing is not feasible. □

Remark 3.7. *Both LP-based pruning rules LP1a and LP1b can be applied separately. Then it is enough that at least one of the inequalities $z_a^* > \epsilon$, $z_b^* > \epsilon$ is valid in order to prune the current node.*

Remark 3.8. *Pruning rule LP1b can be applied only for a monotone contour as by* Contour.

In order to demonstrate Remark 3.8 let us consider the 2D packing of items $1, \ldots, 7$, where items $6, 7$ with $w_6 = w_7 := 4$, $h_6 := 2$, $h_7 := 4$ are not allocated, see Figure 3.9a. The corresponding X-contour of the packing is depicted in Figure 3.9b. Based on the X-contour we build the corresponding block-structure $S_=^x(u) := \{(2,6,2); (4,4,2); (4,2,2)\}$. According to the above, in order to formulate the horizontal relaxation problem, we consider these blocks as 1D bins together with bins type 0: $\chi_0^x := 0$, $\omega_0^x := W$, $\sigma_0^x := \infty$.



(a) 2D packing (b) X-contour

Figure 3.9: Horizontal bar relaxation of the partial solution in the $(0, x)$-direction corresponding to a node $u \in V$. Non-monotone X-contour. $I_x^u := \{1, \ldots, 5\}$, $\bar{I}_x^u := \{6, 7\}$.

With accordance to Lemma 3.11, items with length 4 and quantity 6 must be packed into bins of the first three types so that a packing exists. The number of bins with the length greater than or equal to 4 is 4 but we need 6. Herewith, $z_b^* > \epsilon$. So, in the case of non-monotone contour the current node will be pruned despite the fact that there exists the corresponding 2D packing which fits into the container.

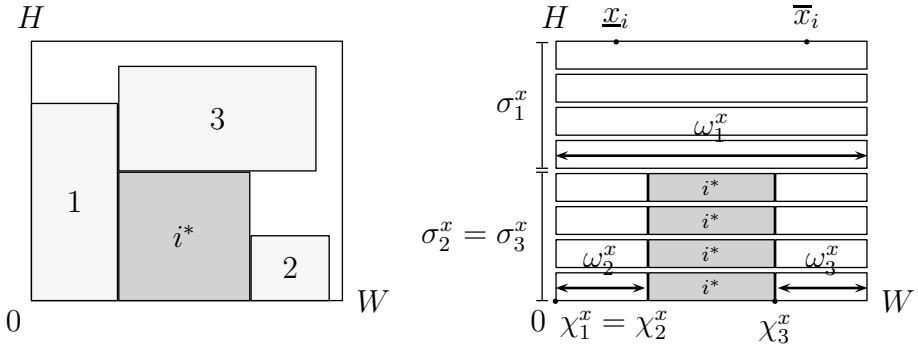
3.5.3 Horizontal bar relaxation with any contour (LP1c)

In order to apply the horizontal bar relaxation as in LP1b also in the case when no monotone X-contour is given we consider another approach. This approach can be

applied with FixMin and FixMinR. In this approach we use both the information concerning the x-coordinate of the fixed items I_x^u and the information concerning the domain of the variables which are not fixed.

In FixMin and FixMinR, item $i^* \in \operatorname{argmin}\{w_i h_i : \ i \in I\}$ is fixed foremost. Herewith, $i^* \in I_x^u$ for any node $u \in V$ where at least one item is fixed. The information concerning x-coordinate of i^* is used in order to define a horizontal bar relaxation. Here we consider only a reduced contour which consist of item i^*. If $0 < x_{i^*} < W - w_{i^*}$ then at most three blocks result, see Figure 3.10b. So, let $\{(\chi_k^x, \omega_k^x, \sigma_k^x)\}_{k=1}^3$ be the resulting horizontal blocks where $\chi_1^x = \chi_2^x := 0$, $\chi_3^x := x_{i^*} + w_{i^*}$, $\omega_1^x := W$, $\omega_2^x := x_{i^*}$, $\omega_3^x := W - x_{i^*} - w_{i^*}$, $\sigma_1^x := H - h_{i^*}$, $\sigma_2^x := \sigma_3^x = h_{i^*}$.

We obtain the input information for the 1D bar relaxation as follows. Each block k is considered as a set of bins with length ω_k^x and quantity σ_k^x together with extra type 4 of bins of length W and unlimited quantity, i.e., $\chi_4^x = 0$, $\omega_4^x = W$, $\sigma_4^x = \infty$. Instead of 2D items we consider 1D items of lengths w_i and quantities h_i where $i \in \bar{I}_x^u$.



(a) 2D packing (b) Partial X-contour

Figure 3.10: Horizontal bar relaxation of the partial solution corresponding to a node $u \in V$ with a non-monotone contour. There exist horizontal blocks $\{(\chi_k^x, \omega_k^x, \sigma_k^x)\}_{k=1}^3$.

Principally, the proposed approach is similar to LP1b but it is tightened by the following observation. Since the allocation of items from $\bar{I}_u^x$ is restricted by intervals $[\underline{x}_i, \overline{x}_i]$, the approach can be tightened by generalization of the meaning of constraints (3.1): If two items $i \neq j$ overlap in their x-projections then their y-projections must not overlap. It is true for fixed items and unfixed items with obligatory overlapping parts as well. For that reason let θ_i^x and θ_i^y with $i \in I$ be defined by

$$\theta_i^x := \begin{cases} 1, \text{ if } \underline{x}_i + w_i - \overline{x}_i > 0; \\ 0, \text{ otherwise,} \end{cases} \qquad \theta_i^y := \begin{cases} 1, \text{ if } \underline{y}_i + h_i - \overline{y}_i > 0; \\ 0, \text{ otherwise.} \end{cases}$$

θ_i^y we define here but we will use it later in Section 3.5.5. Let

$$P_x^u := \{(i,j) \in I \times I : \ i < j, \ \theta_i^x = 1, \ \overline{x}_i < \underline{x}_j + w_j, \ \overline{x}_j < \underline{x}_i + w_i\}$$

be the set of item pairs which overlap in the $(0,x)$-direction. Then projections of any pair from P_x^u must not overlap in the $(0,y)$-direction.

The 1D horizontal bar relaxation of another type can be described as follows. In order to describe horizontal packing patterns with P_x^u taken into account let us redefine binary vectors (3.22) in the following manner. For each horizontal bin of type $k = 1, \ldots, 4$, let J_k^y denote the index set of all binary vectors $b^{jk} = (b_1^{jk}, \ldots, b_m^{jk}) \in \{0,1\}^m$ with

$$b_{i^*}^{jk} = 0, \quad b_i^{jk} = 0, \quad \forall i \in I: \ (\min\{i^*, i\}, \max\{i^*, i\}) \in P_x^u, \quad k = 2, 3,$$
$$\sum_{i \in I \setminus \{i^*\}} w_i b_i^{jk} \le \omega_k^x, \quad b_f^{jk} + b_g^{jk} \le 1, \quad (f,g) \in P_x^u, \quad j \in J_k^y, \quad (3.29)$$

where i-th component b_i^{jk} of vector b^{jk} in the case of $b_i^{jk} = 1$ indicates the j-th pattern of type k which contains one 1D item of type $i \in I$. Note that we also exclude items from the patterns type 2 and 3 whose x-projections overlap with the x-projection of item i^*.

The main idea of the approach is the same as in LP1a and LP1b. Let us formulate the following relaxation problem of MCBPP-1 on vectors (3.29) and variables y^{jk} with $j \in J_k^x$ and $k = 0, \ldots, 3$:

$$z_c^* = \min \sum_{j \in J_0^y} y^{j,0}, \quad \text{s.t.} \quad (3.30)$$

$$\sum_{k=0}^{3} \sum_{j \in J_k^y} b_i^{j,k} y^{j,k} = h_i, \quad i \in I \setminus \{i^*\}; \quad (3.31)$$

$$\sum_{j \in J_k^y} y^{jk} \le \sigma_k^x, \quad k = 1, \ldots, 3; \quad (3.32)$$

$$y^{jk} \ge 0, \quad k = 0, \ldots, 3, \ j \in J_k^y. \quad (3.33)$$

The formulation (3.30)-(3.33) is an LP problem which is solved by the column generation method. In order to solve the relaxation problem the following slave 0-1 linear programs are solved:

$$\bar{c}_0^y = 1 - \max\{\sum_{i \in I \setminus \{i^*\}} d_i a_i : \sum_{i \in I \setminus \{i^*\}} w_i a_i \le W, \ a_i \in \{0,1\}\}, \quad (3.34)$$

$$\bar{c}_1^y = -\max\{d_m + \sum_{i \in I \setminus \{i^*\}} d_i a_i : \sum_{i \in I \setminus \{i^*\}} w_i a_i \le \omega_k^x, \ a_f + a_g \le 1,$$
$$(f,g) \in P_x^u, \ a_i \in \{0,1\}\}; \quad (3.35)$$

$$\bar{c}_k^y = -\max\{d_{m-1+k} + \sum_{i \in (I \setminus \{i^*\}) \setminus P} d_i a_i : \sum_{i \in (I \setminus \{i^*\}) \setminus P} w_i a_i \le \omega_k^x, \ a_f + a_g \le 1,$$
$$(f,g) \in P_x^u, \ a_i \in \{0,1\}\}, \quad k = 2, 3; \quad (3.36)$$

where $P := \{i \in I : \ (\min\{i^*, i\}, \max\{i^*, i\}) \in P_x^u\}$ is the set of items whose x-projections overlap with the x-projection of item i^*.

Lemma 3.12. *If $z_c^* > \epsilon$, $\epsilon > 0$ then there exists no feasible packing of the items $\bar{I}_x^u$ into container (W, H) with the fixed items I_x^u.*

Lemma 3.13. *LP-based pruning rule LP1c can be applied for any X-contour.*

Remark 3.9. *Pruning rules LP1a-LP1c can be defined also for Y-contour.*

3.5.4 Intervals (LP-int)

In pruning rules LP1a and LP1b we did not exploit the information about the domain of variables. In the case of LP1c the domains of variables were used in order to obtain the set of forbidden pairs P_x^u which was used in the slave problems. Here we propose further improvements of the proposed pruning rules which exploit the domain of variables, i.e., the results of the constraint propagation procedure.

Let us consider the vertical bar relaxation from Section 3.5.1. The improvements are the following. Since each variable $g \in X$ can be fixed only within its domain $[\underline{g}, \overline{g}] \cap \mathbb{Z}$ then not every item from $\bar{I}_x^u$ can be packed in an arbitrary block. So, for a block we obtain the set of items which can be packed into the block. Let us define the set of items which can be packed into bins of type k by

$$I_k^x := \{i \in \bar{I}_x^u : \ \overline{x}_i + w_i > \chi_k^x \wedge \underline{x}_i < \chi_k^x + \rho_k^x\}, \quad k \in Q_x^u,$$

so, that $I_0^x := \bar{I}_x^u$, i.e., items from $\bar{I}_x^u$ can be packed into bins of type 0.

Let $a_i^{jk} = 0$, $\forall i \in \bar{I}_x^u \backslash I_k^x$ with $j \in J_k^x$, $k \in Q_x^u$, hold for vectors (3.15). Then we tighten LP-based pruning rule LP1a by introducing the following slave 0-1 linear programs:

$$\bar{c}_k^x = -\max\{d_{m+k} + \sum_{i \in I_k^x} d_i a_i : \ \sum_{i \in I_k^x} h_i a_i \le \lambda_k^x, \ a_i \in \{0,1\}\}, \quad k \in Q_x^u. \tag{3.37}$$

The difference to (3.20) lies in the set of items over which the 0-1 linear programs are defined, $\bar{I}_x^u$ for (3.20) and I_k^x for (3.37).

Remark 3.10. *Pruning rule LP1a for the Y-contour can be tighten in a similar way based on the domain of variables from Y.*

Another improvement can be obtained through the further division of blocks. For instance, in the case of vertical bar relaxation, if we have for an item $i \in I_k^x$ that $\chi_k^x < \overline{x}_i + w_i < \chi_k^x + \rho_k^x$ then it should occur in vertical bins at most $\overline{x}_i + w_i - \chi_k^x$ times. Even better is to divide the k-th block into two blocks $(\chi_k^x, \lambda_k^x, \overline{x}_i + w_i - \chi_k^x)$, $(\overline{x}_i + w_i, \lambda_k^x, \chi_{k+1} - \overline{x}_i - w_i)$ with different sets I_k^x.

In this way, a more detailed information from the partial solution, corresponding to $u \in V$, is used. That leads in general to MCBPP-1 with more bin types but a more tightened pruning rule for OPP.

3.5.5 Forbidden pairs (LP-pair)

This LP-based pruning rule is a generalization of LP1a. It can be used with all strategies, but especially with DISJUNCTIVE. Therefore, the further description is oriented on the application with DISJUNCTIVE. The modifications for the other branching strategies are obvious.

The proposed pruning rule is similar to LP1c. Here we consider bar relaxations in the $(0, x)$- and $(0, y)$-directions. Depending on the direction we forbid certain pairs to

appear in the packing patterns. One set of pairs can be obtained from the information concerning the fixed mutual relations as follows:

$$F_x^u := \{(i,j) \in I \times I : \ i < j \wedge R_{ij} \subseteq \{l,r\}, R_{ij} \in R(u)\};$$
$$F_y^u := \{(i,j) \in I \times I : \ i < j \wedge R_{ij} \subseteq \{b,t\}, R_{ij} \in R(u)\}.$$

Set F_x^u contains item pairs which have possible mutual relation that can be assigned to the left or to the right, and F_y^u item pairs that can be assigned to the bottom or the top. So, projections of items in a pair $(i,j) \in F_x^u$ must not overlap over the $(0,x)$-direction, and $(i,j) \in F_y^u$ over the $(0,y)$-direction.

The second and the third set of pairs are obtained from the following observations. As it follows from inequalities (3.1), if two items $i,j \in I$ overlap in the $(0,x)$-direction, i.e., $(i,j) \in P_x^u$ then these items must not overlap in the $(0,y)$-direction and vice versa. Let

$$P_y^u := \{(i,j) \in I \times I : \ i < j, \ \theta_i^y = 1, \ \overline{y}_i < \underline{y}_j + h_j, \ \overline{y}_j < \underline{y}_i + h_i\}$$

be the set of item pairs which overlap in the $(0,y)$-direction. The definitions of P_x^u and θ_i^y were given in Section 3.5.4. From the other hand, if the domains of two items do not overlap, i.e.,

$$G_x^u := \{(i,j) \in I \times I : \ i < j, \ (\underline{x}_j \geq \overline{x}_i + w_i \vee \underline{x}_i \geq \overline{x}_j + w_j)\},$$
$$G_y^u := \{(i,j) \in I \times I : \ i < j, \ (\underline{y}_j \geq \overline{y}_i + h_i \vee \underline{y}_i \geq \overline{y}_j + h_j)\},$$

then these items cannot overlap over the $(0,x)$-, $(0,y)$-directions, respectively.

The input information for the bar relaxations is constructed as follows. Container (W,H) is considered as a set of 1D bins with length H and quantity W. Instead of 2D items (w_i,h_i) we consider 1D items with lengths h_i and quantities w_i where $i \in I$ is the set of item indexes. Similarly, we define the bar relaxation for the other direction, i.e., 1D bins with length W and quantity H, and 1D items with lengths w_i and quantities h_i are obtained where $i \in I$.

In order to describe the bar relaxations we define vertical and horizontal packing patterns by analogy to vectors (3.15) and (3.22) from Sections 3.5.1, 3.5.2. Let for each vertical bin, J^x denote the index set of binary vectors $a^j = (a_1^j, \ldots, a_m^j) \in \{0,1\}^m$ with

$$\sum_{i \in I} h_i a_i^j \leq H, \quad a_f^j + a_g^j \leq 1, \quad (f,g) \in F_1^u \cup P_y^u \cup G_x^u, \quad j \in J^x, \tag{3.38}$$

and let for horizontal bins, J^y denote the index set of binary vectors $b^j = (b_1^j, \ldots, b_m^j) \in \{0,1\}^m$ with

$$\sum_{i \in I} w_i b_i^j \leq W, \quad b_f^j + b_g^j \leq 1, \quad (f,g) \in F_2^u \cup P_x^u \cup G_y^u, \quad j \in J^y. \tag{3.39}$$

The similar idea from Section 3.5.1 underlies the following bar relaxations. Let us formulate the following continuous relaxation of the set-partitioning model of MCBPP-1 on vectors (3.38), (3.39) and variables x^j with $j \in J^x$ and y^j with $j \in J^j$ which indicate the intensity of usage of vertical and horizontal packing patterns, respectively:

$$z^*_{\parallel} = \sum_{j \in J^x} x^j \to \min, \quad (3.40) \qquad z^*_{=} = \sum_{j \in J^y} y^j \to \min, \quad (3.43)$$

$$\sum_{j \in J^x} a^j_i x^j = w_i,\ i \in I; \quad (3.41) \qquad \sum_{j \in J^y} b^j_i y^j = h_i,\ i \in I; \quad (3.44)$$

$$x^j_i \geq 0,\ i \in I,\ j \in J^x. \quad (3.42) \qquad y^j_i \geq 0,\ i \in I,\ j \in J^y. \quad (3.45)$$

The formulations (3.40)-(3.42) and (3.43)-(3.45) are LP problems which are solved by the column generation method. In order to solve the relaxation problems the following slave 0-1 linear problems are solved:

$$\bar{c}^x = 1 - \max\{\sum_{i \in I} d_i a_i :\ \sum_{i \in I} h_i a_i \leq H,\ a_f + a_g \leq 1,\ (f,g) \in F^u_1 \cup P^u_y \cup G^u_x,\ a_i \in \{0,1\}\}, \quad (3.46)$$

$$\bar{c}^y = 1 - \max\{\sum_{i \in I} d_i b_i :\ \sum_{i \in I} w_i b_i \leq W,\ b_f + b_g \leq 1,\ (f,g) \in F^u_2 \cup P^u_x \cup G^u_y,\ b_i \in \{0,1\}\}, \quad (3.47)$$

where $d := (d_1, \ldots, d_m)$ is the vector of the simplex multipliers. The following statement is true.

Lemma 3.14. *If for an $\epsilon > 0$ $z^*_{\parallel} + \epsilon > W \vee z^*_{=} + \epsilon > H$ then there exists no feasible packing of items I into container (W, H).*

3.5.6 Forbidden pairs and intervals (LP-int-pair)

The constraint propagation procedure provides information concerning the domains of the variables. This information was indirectly used in pruning rules LP1c, LP-int, LP-pair in order to obtain the set of overlapping items or to define the sets of items, which can be packed into the certain blocks, or to define precisely the sizes of blocks. In the proposed approach this information is directly used in slave problems of the column generation method. Herewith, this approach can be considered as an improvement of LP-pair.

Suppose we have the feasible intervals of values which can be assigned to variables X. So, while we generate the columns through solution of slave problem (3.47), the position of the items in the patterns is restricted by that intervals. That condition we impose in the slave problems of the column generation method. Similar considerations are also valid for intervals of Y and slave problem (3.46).

Let l be the vector of items sizes, i.e., $l := w$ or $l := h$, respectively, L the knapsack capacity, P the set of forbidden pairs of items, $[\underline{z}_i, \overline{z}_i]$ the interval of feasible coordinates of item $i \in I$, and d the vector of simplex multipliers. Let designate the following

generalized 0-1 knapsack problem by $KP(l, d, L, P, [\underline{z}_1, \overline{z}_1], \ldots, [\underline{z}_m, \overline{z}_m])$:

$$\sum_{i \in I} d_i \gamma_i \to \max, \tag{3.48}$$

$$\sum_{i \in I} l_i \gamma_i \leq L, \tag{3.49}$$

$$\underline{z}_i \gamma_i \leq z_i \leq \overline{z}_i \gamma_i, \quad i \in I; \tag{3.50}$$

$$z_i + l_i \leq z_j + L(1 - u_{ij}), \quad \forall i, j \in I : \ i < j; \tag{3.51}$$

$$z_j + l_j \leq z_i + L(1 - u_{ji}), \quad \forall i, j \in I : \ i < j; \tag{3.52}$$

$$u_{ij} + u_{ji} \leq \gamma_i, \quad i \in I; \tag{3.53}$$

$$\gamma_i + \gamma_j \leq 1 + u_{ij} + u_{ji}, \quad \forall i, j \in I : \ i \neq j; \tag{3.54}$$

$$\gamma_i + \gamma_j \leq 1, \quad (i, j) \in P; \tag{3.55}$$

$$\gamma_i, \ u_{ij} \in \{0, 1\}, \quad \forall i, j \in I : \ i \neq j. \tag{3.56}$$

The model (3.48)-(3.56) is a simplification for the 1D case of the Padberg model [Pad00] for orthogonal packing. (3.49) is not relevant for the feasibility of the solution but improves the solution time for the problem.

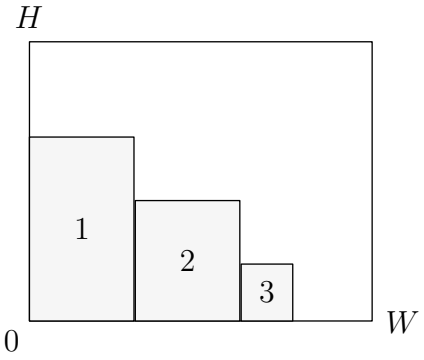
Hereby, the LP problems (3.40)-(3.42) and (3.43)-(3.45) are solved by the column generation method based not on the slave problems (3.46) and (3.47), but on $KP(h, d, H, P^u_y, [\underline{y}_1, \overline{y}_1], \ldots, [\underline{y}_m, \overline{y}_m])$ and $KP(w, d, W, P^u_x, [\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_m, \overline{x}_m])$. Let k^*_x and k^*_y designate the values of an optimal solution of the former and the latter problems. The column generation is performed until $-1 + k^*_x < \epsilon$ and $-1 + k^*_y < \epsilon$, respectively. Lemma 3.14 holds here.

3.5.7 Advanced bar relaxation with a monotone contour (LP-advanced)

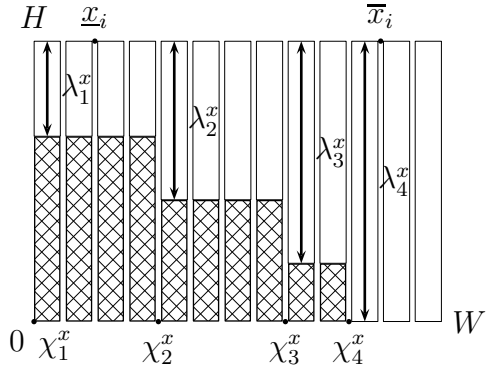
In this section we consider improvements of the pruning rules LP1a and LP1b. Therefore, it can be used with CONTOUR. Let be given a monotone X-contour and corresponding block-structures $S^x_{\parallel}(u)$, $S^x_{=}(u)$. Based on the obtained information we consider both vertical and horizontal bar relaxations as in LP1a and LP1b together as a single problem. The single LP problem is tightened by connecting constraints which are explained further.

Let us consider Figure 3.11 in order to describe the approach. From geometric considerations, the area of an item $i \in I$ which is packed in the first vertical block $(\chi^x_1, \lambda^x_1, \rho^x_1)$, i.e., $h_i \sum_{j \in J^x_1} a^{j1}_i x^{j1}$, see Figure 3.11b, is obviously less or equal to the area of this item which is packed into horizontal block $(\chi^x_1, \omega^x_1, \sigma^x_1)$, i.e., $w_i \sum_{j \in J^y_1} b^{j1}_i y^{j1}$, see Figure 3.11c. Furthermore, the sum of areas of an item which is packed in vertical blocks $\{(\chi^x_k, \lambda^x_k, \rho^x_k)\}$ with $k = 1, 2$ is less or equal to the area of this item which is packed into horizontal blocks $\{(\chi^x_k, \omega^x_k, \sigma^x_k)\}$ with $k = 1, 2$, etc.

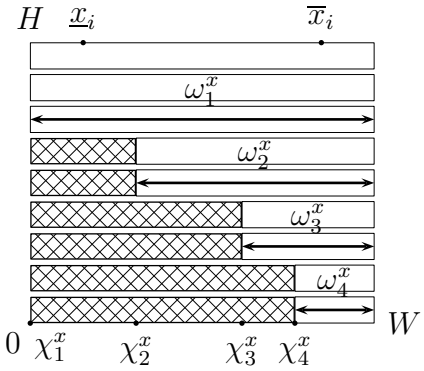
From the other hand, the area of an item which is packed into horizontal block $(\chi^x_{q^u_x}, \omega^x_{q^u_x}, \sigma^x_{q^u_x})$, is less or equal to the area of this item which is packed into vertical block $(\chi^x_{q^u_x}, \lambda^x_{q^u_x}, \rho^x_{q^u_x})$. Furthermore, the sum of areas of an item which is packed into horizontal blocks $\{(\chi^x_k, \omega^x_k, \sigma^x_k)\}$ with $k = q^u_x, q^u_x - 1$ is less or equal to the area of this item which is packed into vertical blocks $\{(\chi^x_k, \lambda^x_k, \rho^x_k)\}$ with $k = q^u_x, q^u_x - 1$, etc.



(a) Feasible 2D packing



(b) Monotone X-contour: Vertical blocks



(c) Monotone X-contour: Horizontal blocks

Figure 3.11: Advanced bar relaxation of the partial solution in the $(0, x)$-direction corresponding to a node $u \in V$ with a monotone contour. Items from $I_x^u := \{1, 2, 3\}$ are marked by hatching. There exist vertical $\{(\chi_k^x, \lambda_k^x, \rho_k^x)\}$ and horizontal $\{(\chi_k^x, \omega_k^x, \sigma_k^x)\}$ blocks with $k = 1, \ldots, 4$.

Based on the above geometric observations we can tighten the relaxation problem by addition of the connection constraints. Let us formulate the following relaxation problem of MCBPP-1 on vectors (3.15), (3.22) and variables x^{jk} with $j \in J_k^x$, y^{jk} with $j \in J_k^y$, $k \in Q_x^u \cup \{0\}$.

$$z_d^* = \min \sum_{j\in J_0^x} x^{j,0} + \sum_{j\in J_0^y} y^{j,0}, \quad \text{s.t.} \tag{3.57}$$

$$\sum_{k=0}^{q_x^u} \sum_{j\in J_k^x} a_i^{jk} x^{jk} = w_i \delta_i, \quad i \in I; \tag{3.58}$$

$$\sum_{k=0}^{q_x^u} \sum_{j\in J_k^y} b_i^{jk} y^{jk} = h_i \delta_i, \quad i \in I; \tag{3.59}$$

$$\sum_{j\in J_k^x} x^{jk} \le \rho_k^x, \quad k \in Q_x^u; \tag{3.60}$$

$$\sum_{j\in J_k^y} y^{jk} \le \sigma_k^x, \quad k \in Q_x^u; \tag{3.61}$$

$$h_i \sum_{k=1}^{s} \sum_{j\in J_k^x} a_i^{jk} x^{jk} \le w_i \sum_{k=1}^{s} \sum_{j\in J_k^y} b_i^{jk} y^{jk}, \quad i \in \bigcup_{k=1}^{s} I_k^x,\ s \in \{1,\dots,\bar{s}\}; \tag{3.62}$$

$$h_i \sum_{k=s}^{q_x^u} \sum_{j\in J_k^x} a_i^{jk} x^{jk} \ge w_i \sum_{k=s}^{q_x^u} \sum_{j\in J_k^y} b_i^{jk} y^{jk}, \quad i \in \bigcup_{k=s}^{q_x^u} I_k^x,\ s \in \{1,\dots,\bar{s}\}; \tag{3.63}$$

$$x^{jk} \ge 0, \quad j \in J_k^x,\ k \in Q_x^u \cup \{0\}; \tag{3.64}$$

$$y^{jk} \ge 0, \quad j \in J_k^y,\ k \in Q_x^u \cup \{0\}; \tag{3.65}$$

The formulation (3.57)-(3.65) is an LP problem which is solved by the column generation method. Let ψ_k^i and ϕ_k^i be defined by

$$\psi_1^k := \begin{cases} 1, \text{ if } i \ge k; \\ 0, \text{ if } i < k; \end{cases} \qquad \phi_k^i := \begin{cases} 1, \text{ if } i \ge k - q_x^u + m; \\ 0, \text{ if } i < k - q_x^u + m. \end{cases}$$

In order to solve the relaxation problem the following slave 0-1 linear programs are solved:

$$\bar{c}_0^x = 1 - \max\{\sum_{i\in \bar{I}_x^u} d_i a_i : \sum_{i\in \bar{I}_x^u} h_i a_i \le \lambda_0^x,\ a_i \in \{0,1\}\}, \tag{3.66}$$

$$\bar{c}_0^y = 1 - \max\{\sum_{i\in \bar{I}_x^u} d_{m+i} a_i : \sum_{i\in \bar{I}_x^u} w_i a_i \le \omega_0^x,\ a_i \in \{0,1\}\}, \tag{3.67}$$

$$\bar{c}_k^x = -\max\{d_{2m+k} + \sum_{i\in I_k^x} (d_i + \psi_k^i d_{i+2m+2q_x^u} + \phi_k^i d_{i+3m+2q_x^u}) a_i : \sum_{i\in I_k^x} h_i a_i \le \lambda_k^x,\ a_i \in \{0,1\}\},\ k \in Q_x^u, \tag{3.68}$$

$$\bar{c}_k^y = -\max\{d_{2m+q_x^u+k} + \sum_{i\in I_k^y} (d_{i+m} + \psi_k^i d_{i+2m+q_x^u} + \phi_k^i d_{i+3m+2q_x^u}) a_i : \sum_{i\in I_k^y} w_i a_i \le \omega_k^x,\ a_i \in \{0,1\}\},\ k \in Q_x^u. \tag{3.69}$$

where $d := (d_1, \ldots, d_{4m+2q_x^u})$ is the vector of the simplex multipliers. Thus, on each step $2(q_x^u + 1)$ slave 0-1 linear problems are solved and the column with argmin$\{\bar{c}_0^x$, $\bar{c}_0^y$, $\bar{c}_k^x$, $\bar{c}_k^y$: $k \in Q_x^u\}$ is considered as a candidate to be appended to the corresponding variable pool.

Herewith, there are $2(q_x^u + 1)$ pools of variables which corresponds to each type of vertical and horizontal bins, see Table 3.2.

Table 3.2: Coefficient matrix of the formulation (3.57)-(3.65).

$k=1$	$\cdots$	$k=q_x^u$	$k=0$	$k=1$	$\cdots$	$k=q_x^u$	$k=0$	
x^{j1}	$\cdots$	x^{j,q_x^u}	$x^{j,0}$	y^{j1}	$\cdots$	y^{j,q_x^u}	$y^{j,0}$	
a_1^{j1}	$\cdots$	a_1^{j,q_x^u}	$a_1^{j,0}$					$= w_1\delta_1$
$\vdots$	$\ddots$	$\vdots$	$\vdots$					$\vdots$
a_m^{j1}	$\cdots$	a_m^{j,q_x^u}	$a_m^{j,0}$					$= w_m\delta_m$
				b_1^{j1}	$\cdots$	b_1^{j,q_x^u}	$b_1^{j,0}$	$= h_1\delta_1$
				$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
				b_m^{j1}	$\cdots$	b_m^{j,q_x^u}	$b_m^{j,0}$	$= h_m\delta_m$
1								$\leq \rho_1^x$
	$\ddots$							$\vdots$
		1						$\leq \rho_{q_x^u}^x$
				1				$\leq \sigma_1^x$
					$\ddots$			$\vdots$
						1		$\leq \sigma_{q_x^u}^x$
$h_1\psi_1^1 a_1^{j1}$	$\cdots$	$h_1\psi_1^{q_x^u} a_1^{j,q_x^u}$		$-w_1\psi_1^1 b_1^{j1}$	$\cdots$	$-w_1\psi_1^{q_x^u} b_1^{j,q_x^u}$		≤ 0
$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\ddots$	$\vdots$		$\vdots$
$h_m\psi_m^1 a_m^{j1}$	$\cdots$	$h_m\psi_m^{q_x^u} a_m^{j,q_x^u}$		$-w_m\psi_m^1 b_m^{j1}$	$\cdots$	$-w_m\psi_m^{q_x^u} b_m^{j,q_x^u}$		≤ 0
$h_1\phi_1^1 a_1^{j1}$	$\cdots$	$h_1\phi_1^{q_x^u} a_1^{j,q_x^u}$		$-w_1\phi_1^1 b_1^{j1}$	$\cdots$	$-w_1\phi_1^{q_x^u} b_1^{j,q_x^u}$		≥ 0
$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\ddots$	$\vdots$		$\vdots$
$h_m\phi_m^1 a_m^{j1}$	$\cdots$	$h_m\phi_m^{q_x^u} a_m^{j,q_x^u}$		$-w_m\phi_m^1 b_m^{j1}$	$\cdots$	$-w_m\phi_m^{q_x^u} b_m^{j,q_x^u}$		≥ 0

The following statement is true.

Lemma 3.15. *If $z_d^* \geq \epsilon$, $\epsilon > 0$ then there exists no feasible packing of items $\bar{I}_x^u$ into container (W, H) with fixed items I_x^u.*

Remark 3.11. *Pruning rule LP-advanced can be tightened by the LP-int approach.*

3.6 Numerical study

In this section we discuss numerical experiments for both pure OPP-2 instances from different sources and SPP-2 instances.

The algorithm was implemented as a single-threaded application in C++ based on Visual Studio 2008, compiler version 9.0.30729, on an AMD Athlon 64 Dual Core 4200+

(2.2 GHz) CPU. IBM ILOG CPLEX 12.1 was used as an LP and ILP solver. ILOG CP 1.6 with ILOG Scheduler 6.8 was used as a constraint programming framework. The test instances, detailed results and source code are available on the CaPaD website[2].

The slave problems (3.20)-(3.21), (3.27)-(3.28), and (3.37) were solved by the dynamic programming approach with strong bounds [MPT99], implementation of which was taken from the personal website[3] of D. Pisinger. Slave problems (3.34)-(3.36), (3.46)-(3.47) were solved as the 0-1 knapsack problem with forbidden pairs of items by our own implementation of a branch-and-bound approach. Slave problem (3.48)-(3.56) was solved as an ILP problem by the CPLEX software.

Time limit for each instance and method was set to 900 seconds. Here we consider results for all described branching strategies and the interval graph algorithm from [BR13]. Note, that we can compare the number of nodes only for the algorithms from a common group, i.e., FixMin, FixMinR, Contour, Dichotomy, and Interval. The latter algorithms, interval graph algorithm [BR13], and Disjunctive can be compared with each other on the percentage of the solved instances and solution time. In Tables 3.3–3.7 the number of nodes and time are the mean values over solved instances. From a rational number we take only the integer part without rounding.

3.6.1 OPP-2 instances of Clautiaux et al.

The set of 42 instances [CJM08] of OPP-2 is divided into 15 feasible (F) and 27 infeasible (N) instances. For each instance, the container is a square 20×20 and the number of items $10 \leq m \leq 23$. The name of instances has form wwSm, where ww is the percentage of waste, $S \in \{F, N, X\}$, X is a designation of an instance with unknown solution for the time, when the instances were published.

The problems were generated as follows. Four parameters $W, H, m \in \mathbb{Z}_+$, ww $\in \{0, 1\}$ are initialized. Then numbers $n_1, \ldots, n_m \in \mathbb{Z}_+$ are generated so that $\sum_{i=1}^{m} n_i = WH(1 - \text{ww}) \in \mathbb{Z}_+$. From the obtained numbers, items (w_i, h_i) are generated so, that w_i is random and h_i is determined by w_i. All mentioned distributions are uniform. The generation process for an instance is performed until an "interesting" instance is generated. An instance is of interest, if a simple heuristic and a DFF-based lower bound are not able to determine whether the problem is feasible or not.

Columns A, B of Table 3.3 report results of the original and transposed instances from [CJM08] divided into feasible and infeasible ones. Column $s.\%$ indicates the percentage of solved instances; n indicates the number of nodes which were used in order to obtain the solution; t indicates the total time for the solution (in seconds).

Obviously, the number of nodes for solution in the case with raster points must be less or equal than without them. For the infeasible instances of this set, the number of nodes for both FixMin and FixMinR is equal, since each coordinate $t = 0, \ldots, 19$: $r \in \widetilde{R}_y(20)$. If we transpose the instances (swap width and height) then the number of nodes for solution changes. Herewith, the selection of the dimension, for which the assignment is performed firstly is essential. The selection criteria for the primal direction is not yet clear. We have tested also the efficiency of the branching strategies

[2] http://www.math.tu-dresden.de/∼capad
[3] http://www.diku.dk/∼pisinger

Table 3.3: Instances from [CJM08]: Comparison of the efficiency of the branching strategies with LP-based pruning rules. The numbers marked by a star are the mean values over instances which were solved by FixMin, FixMinR, Dichotomy(c), and Dichotomy(i) branching strategies. Instances: A–original; B–transposed.

	A						B					
Algorithm	Infeasible			Feasible			Infeasible			Feasible		
	s.%	n	t	s.%	n	t	s.%	n	t	s.%	n	t
FixMin	100	804	1	100	714	0	100	81642	57	100	1446	0
								*34679	*27			
FixMin+SS3	100	545	1	100	714	0	96	30594	29	100	1440	0
FixMinR	100	804	1	100	504	0	100	62528	45	100	943	0
								*33707	*27			
FixMinR+SS3	100	545	1	100	504	0	100	56909	48	100	941	0
								*29622	*28			
FixMinR+LP1	100	241	1	100	500	0	96	178	1	100	940	1
FixMinR+LP-int	100	177	4	100	492	1	96	177	1	100	937	1
Dichotomy(c)	100	704	1	100	440	0	100	49245	36	100	738	0
								*30629	*24			
Dichotomy(c)+LP1	100	189	1	100	436	0	96	139	1	100	735	1
Dichotomy(c)+LP-int	100	118	3	100	430	1	96	136	1	100	734	1
Dichotomy(i)	100	708	1	100	439	0	100	38658	29	100	1041	0
								*21406	*18			
Dichotomy(i)+LP1	100	199	1	100	435	0	96	139	1	100	1039	1
Dichotomy(i)+LP-int	100	115	3	100	429	1	96	135	1	100	1037	1
Contour	89	120225	34	100	4209	1	63	149147	45	100	157508	19
Contour+LP1	89	13494	31	100	4089	1	67	33687	118	100	154526	43
Contour+LP-int	67	675	36	100	4022	3	58	1050	171	100	153378	49
Contour+LP-advanced	67	598	40	100	4001	3	58	1029	170	100	153130	51
Interval	96	31466	33	100	1900	0	100	32031	24	100	17911	2
Interval+LP1	96	420	1	100	1709	2	96	70	0	100	17802	8
Interval+LP-int	96	305	2	100	1692	3	96	70	0	100	17798	8
Disjunctive	100	471343	21	100	2790	0	93	4083	0	100	977	0
Disjunctive+LP-pair	100	3029	3	100	591	0	96	57	0	100	864	0
Disjunctive+LP-int-pair	96	95	36	100	401	204	96	47	63	100	849	258
Partition	100	690216	30	100	2916	0	93	4758	0	100	868	0
Partition+LP-pair	100	3772	3	100	620	0	96	109	0	100	758	0
Partition+LP-int-pair	96	162	30	100	410	172	96	98	36	100	747	228
Algorithm [BR13]	96	948	3	100	32	0	96	184	0	100	814	3

with raster points through the enlargement of the physical sizes of the container. If we multiply a dimension of items by 2^{11} then the original method is able to solve only the half of them in contrast to the branching strategies with raster points, which solves all the enlarged instances.

On average, see Table 3.3, Dichotomy is better than FixMin or even FixMinR with respect to the number of nodes and solution time. LP1 designates the usage of LP1a and LP1b for Contour, and LP1a and LP1c for the other strategies. Disjunctive becomes especially better for infeasible instances at the cost of slightly increased time, if LP pruning rules are applied.

3.6.2 Self-generated OPP-2 instances

We have tested the algorithms according to two scenarios. In the first one, the test set, containing 630 instances of OPP-2, both infeasible and feasible ones as well, is divided into three classes with different maximal items side ratios $r_{\max}$, i.e., $r_{\max} = 1, 3, 20$. Each of these classes is further divided into subclasses according to the waste ratio from 0 to 40 with step 2. Table 3.4 reports results only for packing waste ratios from 0 to 10 with step 2 and from 10 to 40 with step 10, i.e., for 270 instances. Container is a square with side length 1000. Number of items $m = 20$.

In the second scenario, the test set contains 450 instances and is divided into five classes with different number of items for an instance from 10 to 30 with step 5. Here we present the results for the instances with at least 15 items. For the complete results refer to the CaPaD website[4]. Each class is divided into subclasses as in the first scenario, see Table 3.6. For each of the 450 instances, the ratio of item sides is at most 2.

Every instance was generated [BR13, BKRS13] as follows. The total volume of the items $10^9(1-e)$, where e is the waste volume (%), is separated into m intervals by $m-1$ uniformly distributed numbers $z_1, \ldots, z_{m-1}$ in $(0, 10^9(1-e))$. The numbers $z_1, \ldots, z_{m-1}$ are sorted and item volumes are set as follows: $v_1 = z_1$, $v_m = 10^9(1-e) - z_{m-1}$ and $v_i = z_i - z_{i-1}$ for $i = 2, \ldots, m-1$. If ratio v_i/v_j of the volumes of some two items $i, j \in I$ was greater than 8000 then the generation procedure restarted. To obtain both sides of an item, its volume v_i, $i = 1, \ldots, m$ is factorized with the help of two random numbers a_1 and a_2, whose sum is 2. These numbers are calculated in the same way as the volumes. The sides of item i are set $w_i = \lfloor v_i^{a_1/2} \rfloor$ and $h_i = \lfloor v_i^{a_2/2} \rfloor$. If item i is in one dimension larger than 1000 or $r_{\max}$ times the length of another side then volume v_i is factorized again.

The second part of Table 3.4 reports results for the original method [CJM08], but equipped with raster points, i.e., FixMinR branching strategy, and Dichotomy branching strategy. Both have almost the same effectiveness, Dichotomy solves 38% (105) of instances FixMinR solves 38% (103). Moreover Dichotomy solves the mentioned instances and needs thereby on 20% less nodes and time than FixMinR. From the set of 630 instances 310 (49%) were solved by FixMinR, 24 were proven as infeasible, and 286 were proven as feasible. Dichotomy has solved 328 instances (52%), where 24 were proven as infeasible, and 304 as feasible. For the results on the restricted set of instances refer to Table 3.4.

The last part of the table reports results for Disjunctive branching strategy and that with the LP bounds LP-pair. Both have the same effectiveness about 91% (Disjunctive – 247, Disjunctive+LP-Pair – 247) which is clearly higher than of those of FixMinR and Dichotomy. The application of LP bounds in Disjunctive, i.e., Disjunctive+LP-pair, reduces the number of processed nodes and solution time by almost 12%. From the set of 630 instances 607 (96%) were solved by Disjunctive+LP-pair, 58 were proven as infeasible, and 549 as feasible in contrast to Disjunctive which has solved on 1 instance less.

All instances with smaller waste ratio were very hard to solve for all strategies. The most difficult were instances with the maximal side ratio 1. Only a half of such

[4] http://www.math.tu-dresden.de/∼capad

Table 3.4: Self-generated instances: Comparison of the efficiency of the branching strategies. Number of instances is 630, $W = H = 1000$, m=20, maximal side ratio $r_{\max} = 1, 3, 20$. The column w.% is the waste ratio (%), s.% is the amount of solved instances (%). The column n indicates the number of nodes which were used in order to obtain the solution, t indicates the total time for the solution (in seconds). Algorithms: A–algorithm from [BR13]; B–FixMinR; C–Dichotomy; D–Disjunctive; E–Disjunctive+LP-pair.

		A			B			C			D			E		
$r_{\max}$	w.%	s.%	n	t	s.%	n	t	s.%	n	t	s.%	n	t	s.%	n	t
1	0	90	1855	4	0	-	-	0	-	-	50	3623133	289	50	3568171	314
	2	20	1	0	0	-	-	0	-	-	30	5701880	411	30	5455373	469
	4	0	-	-	0	-	-	0	-	-	90	2522761	171	90	2504440	183
	6	60	29255	105	40	165192	13	40	74343	6	80	609380	39	80	608416	43
	8	100	5583	16	40	1206280	113	40	429878	43	100	41719	3	100	41719	3
	10	100	2306	7	30	279	0	30	8449	1	100	168	0	100	168	1
	20	100	214	2	60	92101	7	60	23124	3	100	176	0	100	176	1
	30	100	172	2	60	212355	21	60	138020	16	100	194	0	100	194	1
	40	100	116	2	50	3398	0	50	305	1	100	180	0	100	180	1
		74	4937	17	31	279934	25	31	-	-	83	1388843	101	83	1353204	112
3	0	100	1	0	0	-	-	0	-	-	60	2176138	171	70	237328	22
	2	0	-	-	0	-	-	0	-	-	60	448942	33	60	446017	39
	4	30	35169	110	10	184	1	10	116	1	100	704760	52	100	703702	57
	6	80	31970	103	0	-	-	0	-	-	100	861460	53	100	859324	63
	8	100	3225	9	40	1107017	65	50	2591289	101	100	32874	2	100	32819	3
	10	100	1300	5	80	1206821	78	80	258292	45	100	187	0	100	187	1
	20	100	246	2	80	629079	26	80	68125	4	100	175	0	100	175	1
	30	100	127	2	80	128507	5	80	22565	1	100	169	1	100	169	1
	40	100	136	2	70	1971	0	80	1940345	112	100	189	1	100	189	1
		78	9021	29	40	512262	28	42	813455	43	91	469432	34	92	253323	20
20	0	100	1	0	30	1	0	30	1	0	100	15204	2	100	11371	2
	2	60	9052	23	20	1	0	20	1	0	90	61291	8	90	61006	9
	4	60	23814	65	40	1	0	50	17619	91	100	630827	56	100	630738	60
	6	100	15080	40	30	1105395	142	30	25955	12	100	1495	1	100	1495	1
	8	100	6145	16	30	11756	1	20	11	0	100	264	1	100	264	1
	10	100	519	3	50	908657	57	50	269729	25	100	1712	1	100	1712	1
	20	100	182	1	80	207698	10	80	45322	3	100	323	1	100	322	1
	30	100	120	1	40	134203	7	40	27043	3	100	142	1	100	142	1
	40	100	131	2	70	450004	29	70	59623	5	100	164	1	100	164	1
		91	6116	16	43	313079	27	43	49478	15	98	79046	7	98	78579	8
Mean:		81	6668	20	38	360518	27	38	285721	22	91	645774	48	91	561702	47

instances were solved by Disjunctive and Disjunctive+LP-pair, and none of them by FixMinR and Dichotomy. The same tendency we observe for the second set of test instances, see Table 3.6.

The effectiveness of Disjunctive and Disjunctive+LP-pair increases while the waste and maximal side ratio rise. The strategies FixMin, FixMinR and Dichotomy behave contrarily, namely their effectiveness decreases while the waste ratio rises. From waste ratio 8% Disjunctive and Disjunctive+LP-pair could solve all instances. For

Table 3.5: Self-generated instances. Algorithms: A–algorithm from [BR13]; B–Partition; C–Partition+LP-pair.

		A			B			C		
$r_{\max}$	w.%	s.%	n	t	s.%	n	t	s.%	n	t
1	0	90	1855	4	40	4041145	281	40	3974547	290
	2	20	1	0	30	5667442	327	30	5419262	374
	4	0	-	-	90	2354947	135	90	2337868	141
	6	60	29255	105	80	572515	32	80	571459	34
	8	100	5583	16	100	39225	2	100	39225	2
	10	100	2306	7	100	122	0	100	122	1
	20	100	214	2	100	127	0	100	127	1
	30	100	172	2	100	137	0	100	137	1
	40	100	116	2	100	122	0	100	122	1
		74	4937	17	82	1408420	86	82	1371429	93
3	0	100	1	0	60	2239589	151	70	219305	17
	2	0	-	-	60	470315	29	60	467245	32
	4	30	35169	110	100	691403	43	100	690351	45
	6	80	31970	103	100	822701	39	100	820701	44
	8	100	3225	9	100	27621	2	100	27570	2
	10	100	1300	5	100	139	0	100	139	1
	20	100	246	2	100	123	0	100	123	1
	30	100	127	2	100	115	1	100	115	1
	40	100	136	2	100	126	1	100	126	1
		78	9021	29	91	472458	29	92	247296	16
20	0	100	1	0	100	17283	2	100	12171	2
	2	60	9052	23	100	1182913	91	100	1182583	92
	4	60	23814	65	100	662122	51	100	662032	52
	6	100	15080	40	100	1519	1	100	1519	1
	8	100	6145	16	100	244	1	100	244	1
	10	100	519	3	100	1816	1	100	1816	1
	20	100	182	1	100	298	1	100	296	1
	30	100	120	1	100	109	1	100	109	1
	40	100	131	2	100	116	1	100	116	1
		91	6116	16	100	207380	16	100	206765	16
	Mean:	81	6668	20	91	696086	44	91	608497	42

these instances the effectiveness of FixMinR and Dichotomy remained on the average level. For the instances with the bigger waste ratio, Disjunctive and Disjunctive+LP-pair could even solve them using almost constant number of nodes and time. Thus, Dichotomy is stronger than the original method [CJM08], and Disjunctive and its combination with LP bounds is even stronger.

3.6.3 OPP-2 instances of Hopper & Turton

Here we consider the waste-free instances from [Hop00, HT00]. These instances are divided into three classes C, N, T, each having 21, 35, and 35 instances, respectively. In Table 3.8 we present only results for the original and transposed instances which were solved by at least one of the algorithms. The first mean value is given only over

instance which were solved by all algorithms. The second one is given over all instances. In column A we present the results of the slightly improved algorithm from [BR13], which was kindly provided by the author.

3.7 Conclusions

Here we investigated and modified the state-of-the-art CP approaches for orthogonal packing problems and adapted LP-based pruning rules of different types into the constraint propagation process of the CP.

The main theoretical and experimental observations of the paper are the following:

- The "fix or postpone" strategy is good, if one successfully chooses the "proper" branching direction. There exist no obvious criteria for the selection of the "proper" direction.
- The dichotomy branching strategy reduces the number of nodes and time in comparison to the "fix or postpone" strategy.
- All strategies solve infeasible instances usually slower than feasible ones. It is explained by the fact that for a feasible instance we only need to find a feasible solution. In the case of an infeasible instance we have to enumerate much of the solution space.
- LP-based pruning rules significantly reduce the number of nodes and time for the solution of infeasible instances.
- In contrast to the conclusions in [SO08, Sim08], the disjunctive branching strategy brings the best results on average for both infeasible and feasible instances as well. However, usually it solves feasible instances extremely faster than infeasible ones. Combination of the disjunctive strategy with LP-based pruning rules improves the results.

3.8 Acknowledgments

We thank François Clautiaux for the kindly provided code of the original algorithm; David Pisinger for the provided code for the solution of 0-1 knapsack problems. We appreciate the Academic Initiative of IBM which enables many researchers all over the world to compare their methods using state-of-the-art IBM ILOG Optimization Software.

Table 3.6: Self-generated instances: Comparison of the efficiency of the branching strategies. Number of instances is 450, $W = H = 1000$, $r_{\max} = 2$. The column w.% is the waste ratio (%), s.% is the amount of solved instances (%). The column n indicates the number of nodes for the CP algorithms, which were used in order to obtain the solution, t indicates the total time for the solution (in seconds). Algorithms: A–algorithm from [BR13]; B–FixMinR; C–Dichotomy; D–Disjunctive; E–Disjunctive+LP-pair.

		A			B			C			D			E		
m	w.%	s.%	n	t	s.%	n	t	s.%	n	t	s.%	n	t	s.%	n	t
15	0	70	30944	34	20	52228	111	30	92590	219	100	405957	17	100	388411	22
	2	100	4040	4	80	473691	96	80	321576	67	100	25984	1	100	24952	3
	4	100	1014	2	100	14065	31	100	4072	9	100	4265	0	100	4199	1
	6	100	620	1	100	3824	7	100	1867	4	100	281	0	100	281	0
	8	100	162	1	100	882515	71	100	464850	45	100	124	0	100	124	0
	10	100	90	1	100	47	0	100	24	0	100	98	0	100	98	0
	20	100	86	1	100	27	0	100	15	0	100	101	0	100	101	0
	30	100	64	1	100	32	0	100	18	0	100	96	0	100	96	0
	40	100	60	1	100	127	0	100	107	0	100	93	0	100	93	0
		96	4120	5	88	158506	35	90	98346	38	100	48555	2	100	46483	2
20	0	70	47649	157	30	51	0	50	668702	201	100	2177629	110	90	748190	135
	2	100	9646	30	90	642493	77	90	344971	40	100	420320	17	100	417688	81
	4	100	4454	12	90	2242836	106	90	622734	34	100	240	0	100	240	0
	6	100	839	3	80	50	0	80	34	0	100	1426	0	100	1426	0
	8	100	808	3	100	1222948	87	100	673449	56	100	173	0	100	173	0
	10	100	624	2	90	29419	2	90	19693	1	100	175	0	100	175	0
	20	100	293	2	70	105379	11	80	992271	91	100	171	0	100	171	0
	30	100	136	1	70	35280	2	70	20659	2	100	173	0	100	173	0
	40	100	183	1	100	397	0	100	318	0	100	191	0	100	191	0
		96	7181	23	80	475428	31	83	371425	47	100	288944	14	98	129825	24
25	0	20	12658	69	60	1843387	153	60	1018281	138	90	1913886	95	70	338401	74
	2	60	8967	45	30	228	0	30	72	0	100	6281	0	100	6000	2
	4	100	13318	88	60	227447	30	60	163268	26	100	270	0	100	270	0
	6	100	6580	36	50	4481	0	50	904	0	100	271	0	100	271	0
	8	100	2115	10	20	1046538	97	20	558383	70	100	279	0	100	279	0
	10	100	1891	9	50	702698	48	50	591568	45	100	269	0	100	269	0
	20	100	427	3	40	81423	7	40	54526	5	100	274	0	100	274	0
	30	100	254	3	60	760	0	60	343	0	100	270	0	100	270	0
	40	100	254	3	80	3669	0	90	1273222	82	100	283	0	100	283	0
		86	5162	29	50	434514	37	51	406729	40	98	213564	10	96	38479	8
30	0	0	-	-	20	504	1	30	570085	79	100	366645	21	80	155148	78
	2	20	15599	182	20	9159	1	30	2137570	250	100	2601328	149	80	3920	2
	4	50	13824	130	50	2263726	159	50	767518	72	100	392	0	100	392	1
	6	50	7106	81	50	68162	7	50	20546	4	100	392	0	100	392	1
	8	90	8258	87	10	96113	7	10	40529	5	100	392	0	100	392	1
	10	100	10966	128	20	27837	4	20	12544	2	100	419	0	100	419	1
	20	100	2121	20	10	1044759	79	10	475476	45	100	406	0	100	406	1
	30	100	2673	30	40	534	0	40	225	1	100	416	0	100	416	1
	40	100	537	6	60	2212	0	60	807	1	100	399	0	100	399	1
		67	7635	82	31	390333	28	33	447255	51	100	330087	19	95	17986	9
Mean:		89	4762	26	70	291849	26	71	264788	35	99	176242	9	98	46566	9

Table 3.7: Self-generated instances. Algorithms: A – algorithm from [BR13]; B– Partiton; C – Partition+LP-pair.

		A			B			C		
m	w.%	s.%	n	t	s.%	n	t	s.%	n	t
15	0	70	30944	34	100	370132	15	100	352909	19
	2	100	4040	4	100	25248	1	100	24173	2
	4	100	1014	2	100	3813	0	100	3752	0
	6	100	620	1	100	235	0	100	235	0
	8	100	162	1	100	91	0	100	91	0
	10	100	90	1	100	66	0	100	66	0
	20	100	86	1	100	70	0	100	70	0
	30	100	64	1	100	63	0	100	63	0
	40	100	60	1	100	58	0	100	58	0
		96	4120	5	100	44419	1	100	42379	2
20	0	70	47649	157	100	2184825	100	90	742171	122
	2	100	9646	30	100	381132	15	100	378783	73
	4	100	4454	12	100	183	0	100	183	0
	6	100	839	3	100	1363	0	100	1363	0
	8	100	808	3	100	125	0	100	125	0
	10	100	624	2	100	128	0	100	128	0
	20	100	293	2	100	123	0	100	123	0
	30	100	136	1	100	117	0	100	117	0
	40	100	183	1	100	130	0	100	130	0
		96	7181	23	100	285347	12	98	124791	21
25	0	20	12658	69	90	1939020	90	70	323833	67
	2	60	8967	45	100	6874	0	100	6571	2
	4	100	13318	88	100	201	0	100	201	0
	6	100	6580	36	100	201	0	100	201	0
	8	100	2115	10	100	210	0	100	210	0
	10	100	1891	9	100	203	0	100	203	0
	20	100	427	3	100	192	0	100	192	0
	30	100	254	3	100	191	0	100	191	0
	40	100	254	3	100	191	0	100	191	0
		86	5162	29	98	216364	10	96	36865	7
30	0	0	-	-	100	348355	19	80	148984	75
	2	20	15599	182	100	2714513	144	80	3594	2
	4	50	13824	130	100	296	0	100	296	1
	6	50	7106	81	100	298	0	100	298	1
	8	90	8258	87	100	295	0	100	295	1
	10	100	10966	128	100	324	0	100	324	1
	20	100	2121	20	100	300	0	100	300	1
	30	100	2673	30	100	306	0	100	306	1
	40	100	537	6	100	278	0	100	278	1
		67	7635	82	100	340551	18	95	17186	9
Mean:		89	4762	26	99	177346	8	98	44253	8

Table 3.8: Instances from [Hop00, HT00]: Comparison of the efficiency of the branching strategies. W, H are the container sizes. The column n indicates the number of nodes which were used in order to obtain the solution. The column t indicates the total time for the solution (in seconds). Algorithms: A–algorithm from [BR13]; B–FixMinR; C–Dichotomy; D–Disjunctive; E–Disjunctive+LP-pair.

				A		B		C		D		E	
Inst.	m	W	H	n	t	n	t	n	t	n	t	n	t
C1_1	16	20	20	87	1	63	0	59	0	3529	0	232	0
C1_2	17	20	20	-	-	6593	9	4298	6	15393	1	9239	2
C1_3	16	20	20	1590	3	17	0	25	0	1242	0	521	0
C2_3	25	40	15	9528	42	31	0	68	0	36735	2	1511	2
N1a	17	200	200	19	1	40493	139	4476	18	48226	3	30278	3
N1b	17	200	200	11	1	-	-	-	-	8725	0	5724	1
N1c	17	200	200	12	1	-	-	-	-	1520	0	1317	0
N1d	17	200	200	23	1	543	1	403	1	4093	0	2323	0
N1e	17	200	200	23	1	-	-	-	-	34770	2	30533	3
T1a	17	200	200	60	1	67958	253	16490	71	1151	0	942	0
T1b	17	200	200	23	1	543	1	403	1	4093	0	2323	0
T1c	17	200	200	50	1	1318	4	386	1	314	0	314	0
T1d	17	200	200	13	1	81	0	67	0	931	0	931	0
T1e	17	200	200	6	1	3591	12	874	3	952	0	823	0
T2a	25	200	200	21090	102	-	-	-	-	3179720	174	1915659	387
				1139	5	11463	41	2325	9	10126	0	4019	0
Mean C1_1-T2b:				2323	11	11021	38	2504	9	222759	12	133511	26
TC1_1	16	20	20	275	1	119	0	99	0	11564	0	1001	0
TC1_2	17	20	20	-	-	27	0	38	0	503461	23	428050	95
TC1_3	16	20	20	184	1	26	0	39	0	301	0	253	0
TC2_3	25	15	40	-	-	38	0	65	0	-	-	-	-
TC3_1	28	60	30	-	-	73056	9	51363	6	-	-	-	-
TC3_3	28	60	30	-	-	81169	131	62066	110	341063	17	145112	141
TN1a	17	200	200	8	1	-	-	-	-	3180	0	2849	0
TN1b	17	200	200	14	1	126913	440	55748	202	3371	0	1860	0
TN1c	17	200	200	9	1	104379	417	14233	74	569	0	545	0
TN1d	17	200	200	24	1	75129	228	40147	126	13663	1	1669	1
TN1e	17	200	200	21	1	-	-	-	-	48800	3	40894	3
TT1a	17	200	200	23	1	115	0	104	0	559	0	502	0
TT1b	17	200	200	24	1	75129	228	40147	126	13663	1	1669	1
TT1c	17	200	200	20	1	466	1	188	1	352	0	352	0
TT1d	17	200	200	9	1	64	0	45	0	283	0	283	0
TT1e	17	200	200	13	1	52	0	31	0	1949	0	1704	0
TT2a	25	200	200	-	-	-	-	-	-	1954947	118	1699767	133
TT2b	25	200	200	18987	124	-	-	-	-	413776	25	243942	104
				59	1	38239	131	15078	52	4627	0	983	0
Mean TC1_1-TT2b:				1508	10	38334	103	18879	46	206968	11	160653	29

Chapter 4

Constraint Programming Approaches for 3D Orthogonal Packing

We consider the 3D orthogonal feasibility problem (OPP-3). Given a set of rectangular items, OPP-3 is to decide whether all items can be orthogonally packed into the given rectangular container. We propose an approach based on the ideas of the known recent constraint programming (CP) approaches for OPP-2 and adapt 1D relaxation bounds based on linear programming (LP) into the constraint propagation process of the CP. Numerical results demonstrate the efficiency of the proposed strategies and of the combination of CP and LP-based pruning rules.
Keywords: constraint programming, linear programming, column generation.

4.1 Introduction

The task under consideration is as follows: m 3-dimensional items with sizes (w_i^1, w_i^2, w_i^3) are to be packed into a container (W^1, W^2, W^3). The input data are:

- container sizes $W^d \in \mathbb{Z}_+$ for $d \in D := \{1, \ldots, 3\}$;
- index set of items $I := \{1, \ldots, m\}$;
- items sizes $w_i^d \in \{1, \ldots, W^d\}$ for $i \in I$ and $d \in D$.

The 3-*dimensional orthogonal packing feasibility problem* (OPP-3) [BR13] asks whether all the items can be orthogonally packed into the container without rotations. The guillotine constraint [MAVdC10, CJM08] is not considered. All input data are positive integers. This problem can be easily generalized for higher or reduced down to two dimensions, so sometimes it will be mentioned without an indication of the dimension.

In the case when the items cannot be packed into the container, it is enough to declare a negative response, otherwise the solution is a feasible packing layout with

all items allocated orthogonally and non-overlapped within the container. OPP is a subproblem in solution methods for orthogonal bin packing (BPP) and knapsack problems (OKP) [FS04a, BB07, PS07]. OPP is polynomially equivalent to the orthogonal strip-packing problem (SPP) [Hif98, AVPT09, KIN+09].

4.1.1 Formulation of OPP-3 and overview of solution methods

Suppose we have a coordinate system with origin $(0, 0, 0)$ and axes $1, 2, 3$ which are associated with W^1-, W^2-, W^3-sides of the container, respectively.

Let us introduce sets of variables $X^d := \{x_i^d : i \in I\}$ with $d \in D$, which represent the allocation points for the items in directions $1, 2, 3$, respectively. An assignment of certain values to the variables is feasible in the sense of OPP-3, if the following constraints are satisfied:

$$x_i^d + w_i^d \leq x_j^d \vee x_j^d + w_j^d \leq x_i^d, \quad \text{for at least one } d,\ \forall (i,j) \in I \times I :\ i < j; \tag{4.1}$$

$$0 \leq x_i^d \leq W^d - w_i^d, \quad i \in I,\ d \in D; \tag{4.2}$$

$$x_i^d \in \mathbb{Z}_+, \quad i \in I,\ d \in D. \tag{4.3}$$

It is assured that items do not overlap, constraints (4.1), and lie within the container, constraints (4.2).

Definition 4.1. *If there exist values of the variables X^d with $d \in D$, so that constraints* (4.1)-(4.3) *are satisfied then the corresponding instance is called feasible, otherwise infeasible.*

Formulation (4.1)-(4.3) is a valid non-linear integer model of OPP-3 which is derived from that of OPP-2 proposed in [CJM08, PS07]. In order to solve OPP-2 in that formulation, some constraint programming methods are successfully applied, which leads to the best results today [CJM08, SO08, Sim08, BCDP11, PS07, KMP10]. The approaches can be divided into two groups, the first one fixes the coordinates of items [CJM08, SO08, Sim08], the other fixes the mutual position of items [PS07, SO08, Sim08, BCDP11]. All described approaches for OPP-2 are discussed in [MSB12a] also together with some linear programming lower bounds.

One of the reasons of the success of the constraint programming paradigm is the efficient *constraint propagation* technique. In every node of the search tree it tries to decide whether the set of constraints (4.1)-(4.3) is consistent. In other words, it tries to prove the infeasibility of the current partial solution when certain values are assigned to some variables or domains of possible values of the variables are restricted. If an inconsistency of the set of constraints cannot be proven then the procedure tries to reduce the domain of possible values for the variables.

There exist many ILP models [Bea85, Pad00, BB07, BKRS09] for OPP-3 based on different representations of feasible solutions. Exact solution of OPP-3 in ILP formulations in cited papers is difficult because of the weak LP bounds of some models [Pad00], quadratic number of intersection variables, and/or pseudo-polynomial number of position-indexed variables [Bea85, BB07] in some models.

4.1.2 Relaxations and bounds for OPP

In order to decide whether an instance of OPP is infeasible, sometimes it is enough to compute a lower bound, e.g., *volume bounds*, *dual-feasible functions* (DFF) [CAVdC10a], *conservative scales* (CS) [FS04b, BKRS13], or to solve a relaxation, e.g., *1D bar relaxations* [Sch99, BKRS09], and relaxations of ILP models. All of the mentioned bounds are discussed in [BKRS13].

The 1D bar relaxation [Sch99, BKRS09] is a double relaxation of OPP-2. Firstly, we divide the container and items into 1D bars with unit thickness. Further we formulate the minimization problem over the number of used 1D bars which are needed to pack all the split items without repetition in a single bar. Secondly, we formulate a set-partitioning model of the above 1D problem, continuously relax it, and solve it by the column generation method [KZ51, GG61, GG63]. The 1D bar relaxation bound can be further strengthened [BKRS09] by additional information, i.e., from a probing procedure which restricts the set of items combined in the bars.

Up to now, there were only few efforts to use the bar relaxation in an algorithm for OPP. In [BR13] the 1D bar relaxation bounds were integrated into a modified interval-graph algorithm from [FS04a]. The bound was also tightened in each dimension using the overlapping information from the graphs. This extended information was used in the column generation. The tightened bound was applied in every node of the branching tree.

In [MSB12a] the 1D bar relaxation bounds were integrated into the constraint propagation procedure of various constraint programming approaches. The 1D bar relaxation starting from simple ones, the 1D bar relaxation on the information obtained from different types of *contour*, and completed by sophisticated ones, the 1D bar relaxation with forbidden pairs and feasible item allocation intervals, was applied in every node of the branching tree. This lead in some cases to increased solution time but in most cases to reduced number of branching decisions, especially for infeasible instances.

4.1.3 Our contributions

Inspired by the results [MSB12a] of the state-of-the-art constraint programming approaches for OPP-2, we investigate, modify and transform them into solution methods for OPP-3. We discuss basics of the algorithm in Section 4.2 and propose some minor modifications in Section 4.3. We compare the basic branching strategy (fix at the lower bound or increase the lower bound also known as "schedule or postpone") with the most successful according to [MSB12a] disjunctive strategy in Section 4.4. In Section 4.5 we propose new pruning rules based on relaxations [Sch99, BKRS09] of four types: a simple 1D bar relaxation with different stock lengths, a 1D bar relaxation, a 2D slice relaxation and a 1D slice-bar relaxation with forbidden item pairs. The input data for the bar relaxations is obtained from the local partial solution, the information from the constraint propagation procedure, and the relative positions of the items in the container. Section 4.6 reports numerical results and conclusions.

4.2 Modification of the basic algorithm

In order to simplify the following description we introduce operator "$\oplus_3$" as $d \oplus_3 k := (d + k - 1) \pmod 3 + 1$.

Similarly to [CJM08, MSB12a] for OPP-2, here we solve the OPP-3 as three scheduling problems. Let be given sets $A^d := \{A_1^d, \ldots, A_m^d\}$ of activities with $d \in D$ and three types of resources of W^1W^2, W^1W^3, and W^2W^3 units. Each of activities A_i^d with $d \in D$ and $i \in I$ has its time interval $[start_i^d, end_i^d)$, where activity A_i^d can occur. $start_i^d$ designates the earliest point of time where activity A_i^d can start, and end_i^d is the latest point of time where activity A_i^d can end. Each activity $A_i^d \in A^d$ has its level of consumption $w_i^p w_i^q$ of resource W^pW^q with $p := \min\{d \oplus_3 1, d \oplus_3 2\}$ and $q := \max\{d \oplus_3 1, d \oplus_3 2\}$. Activities $A_i^d \in A^d$ have durations w_i^d.

Definition 4.2. *A schedule is called continuous, if in the schedule each activity is not interrupted during the execution, and cumulative, if all activities are consuming the same resource.*

At every discrete point of time the resource capacity is limited by a certain value and must not be exceeded.

Let us now superpose each feasible start time point of activity A_i^d with variables X^d then $x_i^d \in [start_i^d, end_i^d - w_i^d] \cap \mathbb{Z} = [\underline{x}_i^d, \overline{x}_i^d] \cap \mathbb{Z}$, i.e., $[\underline{x}_i^d, \overline{x}_i^d]$ are feasible domains for variables. If now we assume that $\underline{x}_i^d := 0$, $\overline{x}_i^d := W^d - w_i^d$ with $i \in I$ and $d \in D$, all three schedules are cumulative and continuous, and if constraints (4.1) for variables from X^d are satisfied then the model based on three scheduling problems connected through constraints (4.1) is a feasible model of OPP-3.

The formulation of OPP-3 modeled by the three scheduling problems is solved by the branch-and-bound method, i.e., a binary branching tree $T := (V, E)$ is built. Two nodes $u, v \in V$ differ by the local set of branching restrictions. Based on the kind of branching restrictions, various branching strategies are possible.

4.3 Minor modifications

In this section we consider minor modifications of the basic algorithm, which are applied in each node of the branching tree as *raster points* and local preprocessing or only in the root node as *initial preprocessing*.

4.3.1 Raster points

It is often excessive to consider for a variable p each point from its domain $[\underline{p}, \overline{p}] \cap \mathbb{Z}$. For instance, if we have only three activities with durations 4, 7, 9 then a schedule with starting point 5 has never to be considered. The points which are of interest are called *raster points* [Sch08] and are calculated as follows:

$$R^d(N) := \{0 \le x \le N : \ x = \sum_{i \in I} w_i^d a_i, \ a_i \in \{0, 1\}, \ i \in I\}, \quad d \in D.$$

In the book [Sch08], the author proposes an approach of a reduction of the number of raster points by consideration of a *reduced set* of raster points.

$$\widetilde{R}^d(N) := \{\max\{k \in R^d(N) :\ k \leq N - r\} :\ r \in R^d(N)\}, \quad d \in D.$$

In order to obtain raster points which are situated to the left and to the right of a point α, let us consider for each $d \in D$ the following definitions:

$$\underline{R}^d(\alpha, N) := \max\{\beta \in \widetilde{R}^d(N) :\ \beta \leq \alpha\}, \quad \overline{R}^d(\alpha, N) := \min\{\beta \in \widetilde{R}^d(N) :\ \beta \geq \alpha\}$$

Let $\underline{R}^d(\alpha) := \underline{R}^d(\alpha, W^d)$. Further we propose some preprocessing, branching strategies and pruning approaches which use raster points and reduced set of raster points.

4.3.2 Initial preprocessing

The procedure of *initial preprocessing* is performed only once for the root node. The idea of the procedure is to eliminate some symmetrical and equivalent solutions which can satisfy constraints (4.1)-(4.3). The results of the procedure are additional constraints which are appended to model (4.1)-(4.3) or replace some of its constraints. Similar procedure was proposed in [MMBS11] for the 1D contiguous bin packing problem (CBPP-1). For further restrictions refer to papers [BM03, CCM07].

In order to eliminate the solutions which are obtained through the symmetry over the vertical and horizontal lines going over the middle of the container, we apply

$$x^d_{i^*} \leq \left\lfloor \frac{W^d - w^d_{i^*}}{2} \right\rfloor, \quad d \in D,\ i^* := \min\{i \in \mathcal{Q}\},$$

where $\mathcal{Q} := \{i \in I :\ \prod_{k=1}^3 w^k_i = \max\{\prod_{k=1}^3 w^k_i :\ i \in I\}\}$ is the index set of the items with the largest volume.

Let $F^d := \{(i,j) \in I \times I :\ i < j \wedge w^d_i + w^d_j > W^d\}$ be the sets of item pairs which do not fit together in direction $d \in D$. Then instead of (4.1), the following constraints are applied.

$$x^k_i + w^k_i \leq x^k_j \vee x^k_j + w^k_j \leq x^k_i, \quad \text{for at least one } k \in \{d \oplus_3 1, d \oplus_3 2\},\ (i,j) \in F^d;$$
$$x^d_i + w^d_i \leq x^d_j \vee x^d_j + w^d_j \leq x^d_i, \quad \text{for at least one } d,\ (i,j) \in (I \times I) \setminus \bigcup_{k=1}^{3} F^k :\ i < j.$$

If two items i and j with $i < j$ are identical, i.e., $w^d_i = w^d_j$ for all $d \in D$ then we apply the following constraints:

$$x^1_i < x^1_j \vee$$
$$x^1_i = x^1_j \wedge x^2_i < x^2_j \vee$$
$$x^1_i = x^1_j \wedge x^2_i = x^2_j \wedge x^3_i + w^3_i \leq x^3_j, \quad (i,j) \in I \times I :\ i < j,\ w^d_i = w^d_j,\ \forall d \in D.$$

Instead of constraints (4.2), the following constraints are applied:

$$0 \leq x^d_i \leq \underline{R}^d(W^d - w^d_i), \qquad i \in I,\ d \in D. \tag{4.4}$$

If $\exists d \in D, i, j \in I \times I\colon\ i < j \wedge w_i^d + w_j^d > W^d$ then set

$$(i^*, j^*) := \text{argmax}\{\prod_{k=1}^{3} w_i^k + \prod_{k=1}^{3} w_j^k :\ (i,j) \in I \times I :\ i < j,\ w_i^d + w_j^d > W^d\}$$

and apply the following constraints only for the smallest d:

$$x_{i^*}^p + w_{i^*}^p \leq x_{j^*}^p \vee x_{i^*}^q + w_{i^*}^q \leq x_{j^*}^q, \quad p := d \oplus_3 1,\ q := d \oplus_3 2.$$

4.3.3 Local preprocessing

This type of preprocessing is performed for each node of the branching tree. The main idea is to reduce the feasible domain of variables to the smallest possible size. In terms of the constraint programming paradigm the local preprocessing is called *constraint propagation* [Apt03]. Since for the solution of our model we use ILOG CP (see Section 4.6 for further details), the local preprocessing is done automatically for each node of the branching tree after its creation.

4.4 New branching strategies

According to [MSB12a], the branching strategies can be divided into two groups. First group of branching strategies operates with variable domains. The other group branches on mutual positions of items. Here we propose two branching strategies. The first one, FixMin3 fixes item coordinates, the other one, Relations3, branches on mutual positions of items. Refer to Section 4.6.1 for an experimental study of the strategies.

4.4.1 Schedule-or-postpone

Let u be a node[1] of the branching tree $T := (V, E)$. Node u is also called a subproblem. If a value is assigned to a variable then the variable is called fixed and the job is scheduled. Let $\bar{I}^d := \{i \in I :\ \underline{x}_i^d \neq \overline{x}_i^d\}$ for $d \in D$ be index sets of unfixed variables from X^d. Herewith, $I^d := I \backslash \bar{I}^d$ are the index sets of fixed variables.

The ideas of the approach are coming from [CJM08] and were also discussed in [MSB12a]. The transformed strategy is a 3-step approach where the variables from X^1 are fixed first, and the variables from X^2, X^3 are fixed next, see Algorithm 4.1.

An important issue is the selection of a branching variable from unfixed ones, steps 1 and 2. It is based on the following observation. If we pack all large items at first instead of packing all items in an arbitrary order then we can usually faster obtain the inconsistency of the system (4.1)-(4.3), if any, because we do not lose much effort during the allocation of small items.

[1]Note that here and further we omit the subscript notion of node u in the definition of sets in order to simplify the description. Note that all sets, which are considered here and further are defined within node u.

As soon as the variable for branching is selected, it is fixed to the value, which is equal to its lower bound (the first branch) or the lower bound for its domain is increased (the second branch), see step 3.

In order to describe the algorithm in a simpler way let us introduce an ordering relation on the variables, so $x_i^d \prec x_j^d \Leftrightarrow i < j$ for $i, j = 1, \ldots, m$, $d \in D$.

Algorithm 4.1 (FixMin3)**.** *Adaptation of the branching rule for OPP-2 from [CJM08] to OPP-3.*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1) If $\exists d \in D$: $\bar{I}^d \neq \emptyset$ then set $d = \min\{k \in D : \ \bar{I}^k \neq \emptyset\}$, and:

$$P := \{x_i^d \in X^d : \ i \in \bar{I}^d, \ \prod_{k=1}^{3} w_i^k = \max\{\prod_{k=1}^{3} w_i^k : \ i \in \bar{I}^d\}\},$$

and goto step 2, else goto Exit.

(2) Select the variable with the smallest lower bound for its domain and then with the lowest index:

$$p := \min_{\prec}\{p \in \operatorname{argmin}\{\underline{p} : \ p \in P\}\}.$$

(3) Definition of the descendant nodes:

v_1: $\overline{p} := \underline{p}$ (schedule);
v_2: $\underline{p} := \overline{R}^d(\underline{p} + 1)$ (postpone).

The following statement is true.

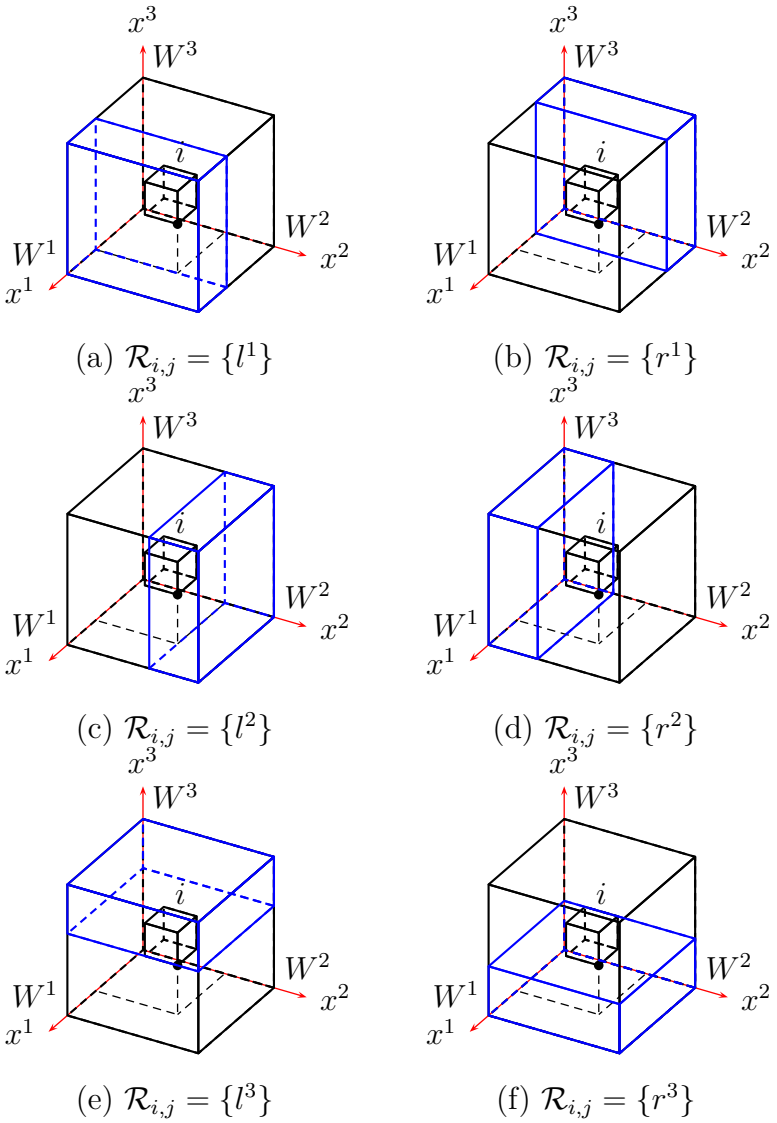
Lemma 4.1. *The depth of the branching tree by* FixMin3 *is $O(m(W^1 + W^2 + W^3))$.*

4.4.2 3-dimensional relations

The branching strategy is oriented on the relation of item pairs (i, j) with $i, j \in I$ and $i < j$. Let $\mathcal{R}_{i,j} := \mathcal{R}_{i,j}(d) \subseteq \{l^k, \bar{l}^k, r^k, \bar{r}^k : \ k \in D\}$ where the notions $l^k, \bar{l}^k, r^k, \bar{r}^k$ imply the following inequalities:

$$\begin{aligned}
l_k \Leftrightarrow &\quad x_i^k + w_i^k \leq x_j^k, \\
\bar{l}_k \Leftrightarrow &\quad x_i^k + w_i^k \geq x_j^k + 1, \\
r_k \Leftrightarrow &\quad x_j^k + w_j^k \leq x_i^k, \\
\bar{r}_k \Leftrightarrow &\quad x_j^k + w_j^k \geq x_i^k + 1,
\end{aligned}$$

Figures 4.1a-4.1f show by the blue colored volumes the feasible locations of an item j, for $\mathcal{R}_{i,j} = \{l^d\}, \{r^d\}$ with $d \in D$, respectively. Now, we associate with each node $u \in V$ the set $\mathcal{R}(u) := \{\mathcal{R}_{i,j} : \ (i, j) \in I \times I \wedge i < j\}$ of relations for the item pairs.



(a) $\mathcal{R}_{i,j} = \{l^1\}$ (b) $\mathcal{R}_{i,j} = \{r^1\}$

(c) $\mathcal{R}_{i,j} = \{l^2\}$ (d) $\mathcal{R}_{i,j} = \{r^2\}$

(e) $\mathcal{R}_{i,j} = \{l^3\}$ (f) $\mathcal{R}_{i,j} = \{r^3\}$

Figure 4.1: Mutual locations of the items i and j. The volumes where the item j can be allocated are marked by the blue color.

In order to prioritize item pairs selected for branching let us introduce a pairs ordering denoted by symbol $\sqsupset$. Suppose $(p,q),(r,k) \in I \times I$: $p < q$ and $r < k$. So, $(p,q) \sqsupset (r,k)$, iff $\prod_{l=1}^{3} w_p^l > \prod_{l=1}^{3} w_r^l \vee (\prod_{l=1}^{3} w_p^l = \prod_{l=1}^{3} w_r^l \wedge \prod_{l=1}^{3} w_q^l > \prod_{l=1}^{3} w_k^l) \vee (\prod_{l=1}^{3} w_p^l = \prod_{l=1}^{3} w_r^l \wedge \prod_{l=1}^{3} w_q^l = \prod_{l=1}^{3} w_k^l \wedge p < r)$.

The main idea of the relations strategy in the 3-dimensional case is to fix a mutual relation for every item pair, see Algorithm 4.2. After a pair is selected, see steps 1, 2, we select a relation to fix for this pair, step 3. At last, when every item pair got a relation, a packing layout (item coordinates) can be computed. If the items lie within the container bounds then the layout is feasible.

In order to simplify the description of the algorithm let us define the set of all item pairs with no fixed l^k or r^k relations:

$$\mathcal{R} := \{R \subseteq \{l^d, r^d, \bar{l}^d, \bar{r}^d : \ d \in D\} : \ R = \emptyset \vee \{l^d, r^d : \ d \in D\} \cap R \neq \emptyset\}.$$

Algorithm 4.2 (RELATIONS3)**.** *Creation of two descendants of a node $u \in V$ according to the relations principle.*
Input data: A node $u \in V$.
Output data: Descendant nodes v_1, v_2.

(1) *If* $\exists \mathcal{R}_{i,j} \in \mathcal{R}(u) : \ \mathcal{R}_{i,j} \in \mathcal{R}$ *then set:*

$$P := \{(i,j) \in I \times I : \ i < j \wedge \mathcal{R}_{i,j} \in \mathcal{R}\},$$

else goto Exit.

(2) *Selection of the pair to branch:*

$$(i,j) := \min_{\sqsupset}\{(i,j) \in P\}.$$

(3) *Definition of the descendant nodes:*
If $l^d \notin R_{i,j} \vee \bar{l}^d \notin R_{i,j}$*:*

$$v_1 : \ \mathcal{R}(v_1) := \mathcal{R}(u) \cup \{l^d\} \cup \{\bar{l}^k, \bar{r}^k : \ k = 1, \ldots, d-1\}$$

$$v_2 : \ \mathcal{R}(v_2) := \mathcal{R}(u) \cup \{\bar{l}^d\} \cup \{\bar{l}^k, \bar{r}^k : \ k = 1, \ldots, d-1\}$$

If $r^d \notin R_{i,j} \vee \bar{r}^d \notin R_{i,j}$*:*

$$v_1 : \ \mathcal{R}(v_1) := \mathcal{R}(u) \cup \begin{cases} \{r^d\} \cup \{\bar{l}^k, \bar{r}^k : \ k = 1, \ldots, d-2\} \cup \{\bar{l}^{d-1}\}, \ \textit{if } d \leq 2; \\ \{l^d\} \cup \{\bar{l}^k, \bar{r}^k : \ k = 1, \ldots, d-1\}, \ \textit{if } d = 3. \end{cases}$$

$$v_2 : \ \mathcal{R}(v_2) := \mathcal{R}(u) \cup \begin{cases} \{\bar{r}^d\} \cup \{\bar{l}^k, \bar{r}^k : \ k = 1, \ldots, d-2\} \cup \{\bar{l}^{d-1}\}, \ \textit{if } d \leq 2; \\ \{r^d\} \cup \{\bar{l}^k, \bar{r}^k : \ k = 1, \ldots, d-1\}, \ \textit{if } d = 3. \end{cases}$$

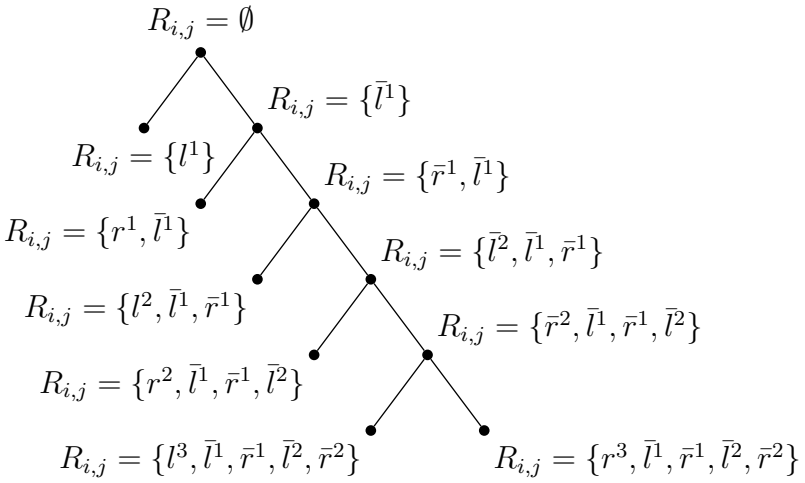


Figure 4.2: Binary branching tree corresponding to RELATIONS3.

Once we have fixed all mutual locations of items, the domain of variables from X^d with $d \in D$ can still contain not just a single value. It depends on the quality of the local preprocessing procedure, see Section 4.3.3. Note that the local preprocessing

procedure is performed for every node after its creation. Several values of a variable from X^d can also be feasible for the model. In our case after all relations are fixed we do not branch variables X^d to a single value.

The following statement is true.

Lemma 4.2. *The depth of the branching tree by* RELATIONS3 *is* $O(m^2)$.

4.5 Advanced constraint propagation

In this section we consider and propose different pruning rules. Some of them were discussed in [Sch99, BKRS09]. Each of them is at least a double relaxation (LP-pair is a triple relaxation). By all of them we relax the 3-dimensionality and go to the 1D case, so 1D relaxation is obtained. Thereafter, we solve the continuous relaxations of the set-partitioning formulation [KZ51, GG61, GG63] of the obtained 1D relaxation.

Let us introduce the notation of a contour and a block-structure. Related but different notions of contour were proposed in [Sch95] for 2D packing layouts. For a node $u \in V$ of the branching tree we assume that items in the d-direction with $d \in D$ are fixed and there are fixed items in the p-direction, $p = \min\{d \oplus_3 1, d \oplus_3 2\}$. Let I_r^d be the set of items whose projection on d-th axis contains a point $r \in [0, W^d)$. Herewith, we can decide whether an item projection intersects a chosen point as $I_r^d = \{i \in I : \ r \in [x_i^d, x_i^d + w_i^d)\}$. Further we introduce $\delta_{r,i}^d$ which indicates whether an item i's coordinate contains point r as follows:

$$\delta_{r,i}^d := \begin{cases} 1, \text{ if } i \in I_r^d; \\ 0, \text{ otherwise;} \end{cases}$$

The items which are fixed in the d-direction and contains point t can be defined as follows:

$$I_r^d(t) := \{i \in I_r^d \cap I^p : \ t \in [x_i^p, x_i^p + w_i^p), \ p := \min\{d \oplus_3 1, d \oplus_3 2\}\},$$

where set I^p is the set of fixed items over the p-direction, which was defined in Section 4.4.1.

Definition 4.3. *The* X_r^d*-contour* C_r^d *with* $d \in D$ *corresponding to a node* $u \in V$ *is the graph of the function*

$$C_r^d(t) := \sum_{i \in I_r^d(t)} w_i^q, \quad t \in [0, W^p), \quad \begin{cases} p := \min\{d \oplus_3 1, d \oplus_3 2\}; \\ q := \max\{d \oplus_3 1, d \oplus_3 2\}. \end{cases}$$

$C_r^d(t)$ is a step function with, in general, some discontinuity points in $(0, W^p)$. Let the sequence $\{\chi_k\}_{k=1}^{s+1}$ contain exactly all the discontinuity points and the border values, such that $0 = \chi_1 < \chi_2 < \ldots < \chi_{s+1} = W^p$, i.e.,

$$\lim_{t \to \chi_k - 0} C_r^d(t) \neq \lim_{t \to \chi_k + 0} C_r^d(t) = C_r^d(\chi_k), \quad k \in \{2, \ldots, s\},$$

where $s+1$ is the number of jump discontinuity points in $(0, W^p)$ plus two border points. For convenience, let

$$C_r^d(W^p) := \lim_{t \to W^p - 0} C_r^d(t).$$

Assigned to an interval $[\chi_k, \chi_{k+1})$ with $k \in S := \{1, \ldots, s\}$ we define a block as the rectangle in $[\chi_k, \chi_{k+1}) \times [0, W^q)$ lying above C_r^d, see Figure 4.3.

Definition 4.4. *The k-th block corresponding to a contour C_r^d is the rectangle $[\chi_k, \chi_{k+1}) \times [C_r^d(\chi_k), W^q)$, denoted by $(\chi_k, \lambda_k, \rho_k)$, where $\lambda_k := W^q - C_r^d(\chi_k)$, and $\rho_k := \chi_{k+1} - \chi_k$.*

In terms of scheduling, the k-th block represents the non-used resource of type q in direction p. Papers [BSM08, MMBS11] define a related notion of slice describing the complete layout in the period $[\chi_k, \chi_{k+1})$. Here a block is a part of a slice.

Definition 4.5. *The sequence of blocks $\{(\chi_k, \lambda_k, \rho_k)\}_{k=1}^s$ is called the block-structure corresponding to a contour C_r^d and is denoted by $S_\parallel$.*

Remark 4.1. *There exists exactly one block-structure for each $d \in D$.*

Further in this section we consider four types of LP-based approaches. The first one, named LP-cont, uses the information concerning the fixed items. The LP-slice approach creates a 2D relaxation where we cut the container into slices. The LP-bar approach creates a 1D bar relaxation where the container is cut into 1D stitches. The LP-pair approach is a double relaxation where at first, 3D container is cut into 2D slices and then these slices are cut into 1D bins. All of the relaxations except LP-cont are tightened with the forbidden item pair sets which are obtained from fixed mutual locations for items in pairs. Herewith, LP-cont is applied with FixMin3 and the others are applied with Relations3.

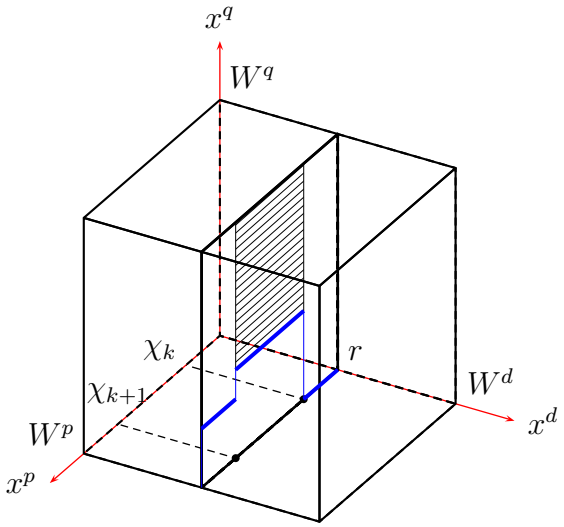


Figure 4.3: Contour of a packing in p-axis for a fixed $d \in D$ and $r \in \widetilde{R}^d(W^d)$.

4.5.1 1D bar relaxation from contour (LP-cont)

Here we propose a pruning rule which is applied with FixMin3. Let $I_r^{d,p} := \{i \in I_r^d : \underline{x}_i^p = \overline{x}_i^p,\ p := \min\{d \oplus_3 1, d \oplus_3 2\}\}$ be the set of fixed items over axis p with fixed $d \in D$. Let us consider the partial solution of a node $u \in V$. Items from $I_r^{d,p}$ give us the information concerning the X_r^d-contour. Based on the X_r^d-contour the corresponding block-structure $S_{\|}$ is build.

The input information for the 1D bar relaxation is constructed as follows. Each block $(\chi_k, \lambda_k, \rho_k)$ with $k \in S$ is considered as a set of bins with length λ_k and quantity ρ_k. In addition to the obtained bins, we also consider one extra type 0 of bins with $\chi_0 := W^p$, $\lambda_0 := W^q$, $\rho_0 := \infty$. In the 2D case each 2D item has a certain geometrical location. Here we relax this condition and consider instead of 2D items 1D items with lengths w_i^q and quantities w_i^p with $i \in \bar{I}_r^{d,p}$, where $\bar{I}_r^{d,p} := I_r^d \backslash I_r^{d,p}$ is the index set of the unfixed items.

The 1D bar relaxation can be described as follows. In order to describe a packing of the bins with obtained items let us introduce packing patterns in the following manner. For each 1D bin of type $k \in S \cup \{0\}$ let J_k denote the index set of all binary vectors $a^{j,k} := (a_1^{j,k}, \ldots, a_m^{j,k}) \in \{0,1\}^m$ with

$$a_i^{j,k} = 0, \quad \forall i \in I_r^{d,p}, \quad \sum_{i \in \bar{I}_r^{d,p}} w_i^q a_i^{j,k} \le \lambda_k, \quad j \in J_k, \tag{4.5}$$

where the i-th component $a_i^{j,k}$ of vector $a^{j,k}$ in the case of $a_i^{j,k} = 1$ indicates the j-th pattern of type k which contains one 1D item of type $i \in I$. Whether item i's x^p-coordinate is fixed, is indicated in the following model by

$$\tilde{\delta}_{r,i}^d := \begin{cases} 0, \text{ if } i \in I_r^{d,p}; \\ 1, \text{ if } i \in \bar{I}_r^{d,p}. \end{cases}$$

The main idea of the approach consists in minimizing the number of used bins of type 0. If at least a small part of that type of bins is used then there exists no packing of residual items $\bar{I}_r^{d,p}$ which fit into the container, and hence, subproblem $u \in V$ is infeasible.

Let us formulate the following continuous relaxation of the set-partitioning model [KZ51, GG61, GG63] of the 1D multiple-capacity bin packing problem[2] (MCBPP-1) on vectors (4.5) and variables $y^{j,k}$ with $j \in J_k$ and $k \in S \cup \{0\}$ which indicate the intensity of usage of packing patterns as follows:

$$y_r^{d,*} = \min \sum_{j \in J_0} y^{j,0}, \quad \text{s.t.} \tag{4.6}$$

$$\sum_{k=0}^{s} \sum_{j \in J_k} a_i^{j,k} y^{j,k} = w_i^p \tilde{\delta}_{r,i}^d, \quad i \in I_r^d; \tag{4.7}$$

$$\sum_{j \in J_k} y^{j,k} \le \rho_k, \quad k \in S; \tag{4.8}$$

$$y^{j,k} \ge 0, \quad k \in S \cup \{0\},\ j \in J_k. \tag{4.9}$$

[2]Usually, the 1D multiple stock size cutting stock problem (MSSCSP-1) [WHS07, AV08] is considered.

This problem is called the 1D bar relaxation.

The formulation (4.6)-(4.9) is an LP problem which is solved by the column generation method [KZ51, GG61, GG63, AV08]. The solution process is started from the initial set of variables (columns), which contains m variables with a large coefficient in the objective function for each constraint (4.7), and initial dual simplex multipliers $d := (d_1, \ldots, d_s)$ determined by a feasible basic solution of (4.6)-(4.9).

The restricted master problem of (4.6)-(4.9) contains variable pools for each type of columns. Each iteration consists of the generation of a column (slave problem) for each pool, its addition into the corresponding pool, and execution of the simplex method on the restricted master problem.

The generation of a column is aimed to maximize the sum of the dual simplex multipliers which is done by the solution of the following 0-1 linear programs:

$$\bar{c}_0 = 1 - \max\{\sum_{i \in \bar{I}_r^{d,p}} d_i a_i : \sum_{i \in \bar{I}_r^{d,p}} w_i^q a_i \leq \lambda_0,\ a_i \in \{0,1\}\}; \tag{4.10}$$

$$\bar{c}_k = -\max\{d_{m+k} + \sum_{i \in \bar{I}_r^{d,p}} d_i a_i : \sum_{i \in \bar{I}_r^{d,p}} w_i^q a_i \leq \lambda_k,\ a_i \in \{0,1\}\}, \quad k \in S. \tag{4.11}$$

Coefficients $\bar{c}_0$, $\bar{c}_k$ are also called *reduced costs*. Each column with a negative reduced cost can improve the value of the objective function and can be added to the pool at each step. But we add only the column with the smallest reduced cost, i.e., $\operatorname{argmin}\{\bar{c}_0, \bar{c}_k : k \in S\}$, since that shows the better solution time. Thus, on each step $s+1$ slave 0-1 linear programs are solved.

The variables which correspond to type 0 of bins have the coefficient in the objective function equal to 1, in contrast to the obtained ones. The column generation process is performed as long as there exists a column which can improve the value of the objective function. That means $-\bar{c}_k > \epsilon$ for some $k \in S \cup \{0\}$ where $\epsilon > 0$ is a small enough constant.

Decision rule LP-cont is applied according to the following lemma.

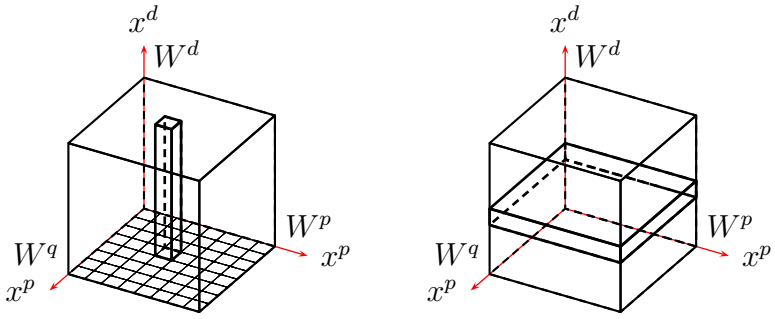
Lemma 4.3. *If $\exists d \in D$ and $r \in \widetilde{R}^d(W^d)$ such that $y_r^{d,*} > \epsilon$, $\epsilon > 0$ then there exists no feasible packing of items $\bar{I}_r^d$ into container with fixed items I^p into container with sizes (W^1, W^2, W^3).*

4.5.2 1D bar relaxation (LP-bar)

Here we consider a pruning rule which is applied with RELATIONS3 and proposed in [Sch99].

The input information for the 1D bar relaxation is constructed as follows. Let us consider the 3-dimensional pattern which is divided into the set of 1D bars with length W^d in the selected direction $d \in D$. So, the projection of the $W^p W^q$ bars has 1×1 unit area on $p0q$ planes with $p := d \oplus_3 1$ and $q := d \oplus_3 2$, see Fig. 4.4a. For the 3-dimensional case the 1D bars must have a fixed 3D allocation. Here we relax this condition and consider instead of 3D items 1D items with length w^d and quantity $w_i^p w_i^q$, where $i \in I$.

The 1D bar relaxation can be described as follows. In order to describe a feasible packing of the 1D bins with the obtained items let us introduce packing patterns in the



(a) 1D bar relaxation (b) 2D slice relaxation

Figure 4.4: 1D bar and 2D slice relaxations. We depict only one 1D bar and one 2D slice in order to simplify the figure.

following manner. For each 1D bin of type d let J^d denote the index set of all binary vectors $a^{d,j} := (a_1^{d,j}, \ldots, a_m^{d,j}) \in \{0,1\}^m$ with

$$\sum_{i \in I} w_i^d a_i^{d,j} \leq W^d, \quad a_f^{d,j} + a_g^{d,j} \leq 1, \quad (f,g) \in F^p \cup F^q, \quad j \in J^d, \tag{4.12}$$

where the i-th component $a_i^{d,j}$ of vector $a^{d,j}$ in the case of $a_i^{d,j} = 1$ indicates the j-th pattern of type d which contains one 1D item of type $i \in I$; F^d is a set of item pairs which do not fit together in the d-direction (F^d is defined in Section 4.3.2). In contrast to (4.5), here the components of the vector do not depend on the fixed items but on the fixed mutual relations.

The main idea of the approach is to minimize the number of used 1D bins in order to pack all the obtained 1D items from I. If this number exceeds the number W^pW^q of the available 1D bins then there exist no packing of items from I with the fixed mutual relations which affect sets F^p and F^q.

Let us formulate the following continuous relaxation of the set-partitioning model of BPP-1 on vectors (4.12) and variables $y^{d,j}$ with $j \in J^d$ which indicate the intensity of usage of packing patterns as follows:

$$y^{d,*} = \sum_{j \in J^d} y^{d,j} \to \min, \text{ s.t. } \sum_{j \in J^d} a_i^{d,j} y^{d,j} = w_i^p w_i^q;\ y_i^{d,j} \geq 0,\ i \in I,\ j \in J^d,\ d \in D. \tag{4.13}$$

The formulations (4.13) are LP problems which are solved by the column generation method. In order to solve the relaxation problems the following 0-1 linear programs are solved:

$$\bar{c}^d = 1 - \max\{\sum_{i \in I} d_i b_i^d : \sum_{i \in I} w_i^d b_i^d \leq W^d,\ b_f^d + b_g^d \leq 1,\ (f,g) \in F^p \cup F^q,$$
$$p := d \oplus_3 1,\ q := d \oplus_3 2,\ b_i^d \in \{0,1\}\}; \tag{4.14}$$

Binary variables b_i denote the fact that item $i \in I$, is used in the d-th pattern. If $-\bar{c}^d \geq \epsilon$ then the current basis is optimal, otherwise a new column is included.

Decision rule LP-bar is applied according to the following lemma.

Lemma 4.4. *If $\exists d \in D$ such that $y^{d,*} - W^p W^q > \epsilon$, $\epsilon > 0$ with $p := d \oplus_3 1$, $q := d \oplus_3 2$ then there exists no feasible packing of items I into container (W^1, W^2, W^3) with fixed relations $\mathcal{R}(u)$.*

4.5.3 2D slice relaxation (LP-slice)

The following pruning rule is applied with RELATIONS3 and proposed in [Sch99]. Here we consider the 3-dimensional pattern which is cut in selected direction $d \in D$ into the set of 2D slices with thickness of 1 unit having $W^p W^q$ area units where $p := d \oplus_3 1$ and $q := d \oplus_3 2$, see Fig. 4.4b. The set of the 3D items is considered as a set of 1D items with length $w_i^p w_i^q$ and quantity w_i^d, where $i \in I$.

In order to describe a feasible packing of the 2D slices with the obtained 1D items we introduce packing patterns in the following manner. For each 2D slice of type d let J^d denote the index set of all binary vectors $a^{d,j} := (a_1^{d,j}, \ldots, a_m^{d,j}) \in \{0,1\}^m$ with

$$\sum_{i \in I} w_i^p w_i^q a_i^{d,j} \leq W^p W^q, \quad a_f^{d,j} + a_g^{d,j} \leq 1, \quad (f,g) \in F^d, \quad j \in J^d, \tag{4.15}$$

where the i-th component $a_i^{d,j}$ of vector $a^{d,j}$ in the case of $a_i^{d,j} = 1$ indicates the j-th pattern of type d which contains one 1D item of type $i \in I$.

Further we omit the condition that each slice has its fixed position and try to minimize the number of used 2D bins while all the obtained from I 1D items are packed. Let us formulate the following continuous relaxation of the set-partitioning model of BPP-1 on vectors (4.15) and variables $y^{d,j}$ with $j \in J^d$ which indicate the intensity of usage of packing patterns as follows:

$$y^{d,*} = \sum_{j \in J^d} y^{d,j} \to \min, \text{ s.t. } \sum_{j \in J^d} a_i^{d,j} y^{d,j} = w_i^d;\ y_i^{d,j} \geq 0,\ i \in I,\ j \in J^d,\ d \in D. \tag{4.16}$$

Formulations (4.16) are LP problems which are solved by the column generation method. In order to solve the relaxation problems the following 0-1 linear programs are solved:

$$\bar{c}^d = 1 - \max\{\sum_{i \in I} d_i b_i^d :\ \sum_{i \in I} w_i^p w_i^q b_i^d \leq W^p W^q,\ b_f^d + b_g^d \leq 1,\ (f,g) \in F^d$$
$$p := d \oplus_3 1,\ q := d \oplus_3 2,\ b_i^d \in \{0,1\}\}, \tag{4.17}$$

where $d := (d_1, \ldots, d_m)$ is the vector of the simplex multipliers.

Decision rule LP-slice is applied according to the following lemma.

Lemma 4.5. *If $\exists d \in D$ such that $y^{d,*} - W^d > \epsilon$, $\epsilon > 0$ then there exists no feasible packing of items I into container (W^1, W^2, W^3) with fixed relations $\mathcal{R}(u)$.*

Remark 4.2. *The experiential study shows that LP-bar prunes non of the subproblems while the solution of the test instances.*

Herewith, we do not present the results of the approaches with LP-bar in Section 4.6.

4.5.4 1D slice-bar relaxation with forbidden pairs (LP-pair)

Here we propose a pruning rule which is applied with RELATIONS3. In the subsequent description we will need extra definitions of item pairs based on different observations.

Let for a fixed $d \in D$ and $r \in \widetilde{R}^d(W^d)$:

$$G_r^{d,k} := \{(i,j) \in I_r^d \times I_r^d : \ i < j, \ (\overline{x}_i^k + w_i^k \leq \underline{x}_j^k \vee \overline{x}_j^k + w_j^k \leq \underline{x}_i^k)\}, \quad k \in \{d \oplus_3 1, d \oplus_3 2\},$$

be the set of item pairs, which do not overlap over k-th axis.

Sets of item pairs which do not overlap in each direction for a fixed $d \in D$ and $r \in \widetilde{R}^d(W^d)$, are defined as follows:

$$F_r^{d,k} := \{(i,j) \in I_r^d \times I_r^d : \ i < j, \ \mathcal{R}_{i,j} \subseteq \{l^k, r^k\}\}, \qquad d \in \{2,3\}.$$

If an item i has an obligatory part which will have a projection over d-axis then it is indicated by:

$$\theta_i^d := \begin{cases} 1, \ \text{if } \underline{x}_i^d + w_i^d - \overline{x}_i^d > 0; \\ 0, \ \text{otherwise}, \end{cases}$$

The set of overlapping item pairs in k-th axis for a fixed $d \in D$, and $r \in \widetilde{R}^d(W^d)$ is defined as follows:

$$P_r^{d,k} := \{(i,j) \in I_r^d \times I_r^d : \ i < j, \ \theta_i^k = 1, \ \underline{x}_j^k + w_j^k > \overline{x}_i^k, \ \underline{x}_i^k + w_i^k > \overline{x}_j^k\},$$

where $k \in \{d \oplus_3 1, d \oplus_3 2\}$.

The input information for the 1D bar relaxations is constructed as follows. For every $d \in D$ and $r \in \widetilde{R}^d(W^d)$ we have a 2D container (W^p, W^q) in the section of a plane $p0q$ going through r, where $p := \min\{d \oplus_3 1, d \oplus_3 2\}$ and $q := \max\{d \oplus_3 1, d \oplus_3 2\}$. This 2D container is considered as a set of 1D bins with length W^q and quantity W^p. Instead of 3D items (w_i^d, w_i^p, w_i^q) with $i \in I_r^d$, which intersect the plane $p0q$ going through r we consider only 2D items (w_i^p, w_i^q). Instead of them we consider 1D items with lengths w_i^q and quantities w_i^p where $i \in I_r^d$. Similarly, we define the 1D bins for the other direction q, i.e., 1D bins with length W^p and quantity W^q, and 1D items with lengths w_i^p and quantities w_i^q.

In order to describe the 1D bar relaxations we define vertical and horizontal patterns. Let for each vertical bin, J^p denote the index set of binary vectors $a^j := (a_1^j, \ldots, a_m^j) \in \{0,1\}^m$ with

$$\sum_{i \in I} w_i^q a_i^j \leq W^q, \quad a_f^j + a_g^j \leq 1, \quad (f,g) \in F_r^{d,p} \cup P_r^{p,q} \cup G_r^{d,p}, \quad j \in J^x, \tag{4.18}$$

and let for horizontal bins, J^q denote the index set of binary vectors $b^j = (b_1^j, \ldots, b_m^j) \in \{0,1\}^m$ with

$$\sum_{i \in I} w_i^p b_i^j \leq W^p, \quad b_f^j + b_g^j \leq 1, \quad (f,g) \in F_r^{d,q} \cup P_r^{q,p} \cup G_r^{d,q}, \quad j \in J^y. \tag{4.19}$$

The similar idea from Section 4.5.1 underlies the following 1D bar relaxations. Let us formulate for $d \in D$ and $r \in \widetilde{R}^d(W^d)$ the following continuous relaxations of the set-partitioning model of MCBPP-1 on vectors (4.18), (4.19) and variables y^j with $j \in J^p$ and z^j with $j \in J^q$ which indicate the intensity of usage of vertical and horizontal packing patterns, respectively:

$$y_r^{d,p,*} = \sum_{j \in J_p} y^j \to \min, \text{ s.t. } \sum_{j \in J_p} a_i^j y^j = w_i^d \delta_{r,i}^p; \; y_i^j \geq 0, \; i \in I, \; j \in J_p^d. \tag{4.20}$$

$$z_r^{d,q,*} = \sum_{j \in J_q} z^j \to \min, \text{ s.t. } \sum_{j \in J_q} b_i^j z^j = w_i^d \delta_{r,i}^q; \; z_i^j \geq 0, \; i \in I, \; j \in J_q^d. \tag{4.21}$$

Formulations (4.20) and (4.21) are LP problems which are solved by the column generation method. In order to solve the relaxation problems the following 0-1 linear programs are solved:

$$\bar{c}_r^{d,p} = 1 - \max\{\sum_{i \in I_r} d_i a_i : \sum_{i \in I_r} w_i^q a_i \leq W^q, \; a_f + a_g \leq 1, \\ (f,g) \in F_r^{d,p} \cup P_r^{p,q} \cup G_r^{d,p}, \; a_i \in \{0,1\}\}; \tag{4.22}$$

$$\bar{c}_r^{d,q} = 1 - \max\{\sum_{i \in I_r} d_i b_i : \sum_{i \in I_r} w_i^p b_i \leq W^p, \; b_f + b_g \leq 1, \\ (f,g) \in F_r^{d,q} \cup P_r^{q,p} \cup G_r^{d,q}, \; a_i \in \{0,1\}\}; \tag{4.23}$$

where $d := (d_1, \ldots, d_m)$ is the vector of the simplex multipliers.

Decision rule LP-pair is applied according to the following lemma.

Lemma 4.6. *If $\exists d \in D$ and $r \in \widetilde{R}^d(W^d)$ such that $y_r^{d,p,*} + z_r^{d,q,*} - W^p - W^q > \epsilon$, $\epsilon > 0$, $p := \min\{d \oplus_3 1, d \oplus_3 2\}$ and $q := \max\{d \oplus_3 1, d \oplus_3 2\}$ then there exists no feasible packing of items I with fixed relations.*

4.6 Numerical study

In this section we discuss numerical experiments for pure OPP-3 instances.

The algorithm was implemented as a single-threaded application in C++ based on Visual Studio 2008, compiler version 9.0.30729, on an AMD Athlon 64 Dual Core 4200+ (2.2 GHz) CPU. IBM ILOG CPLEX 12.1 was used as an LP and ILP solver. ILOG CP 1.6 with ILOG Scheduler 6.8 was used as a constraint programming framework. The test instances, detailed results and source code are available on the CaPaD website[3] and in [MSB12b].

The slave problems (4.10)-(4.11) were solved by the dynamic programming approach with strong bounds [MPT99], implementation of which was taken from the personal website[4] of D. Pisinger. The slave problems (4.20)-(4.21) were solved as the

[3] http://www.math.tu-dresden.de/∼capad
[4] http://www.diku.dk/∼pisinger

0-1 knapsack problem with forbidden pairs of items by our own implementation of a branch-and-bound approach.

Time limit for each instance and method was set to 900 seconds. Here we consider results for the proposed branching strategies and the interval graph algorithm from [BR13]. Note that the algorithms can be compared with respect the percentage of the solved instances and solution time but not the number of nodes. In Tables 4.1–4.5 the number of nodes and time are the mean values over solved instances. From a rational number we take only the integer part without rounding.

4.6.1 Self-generated OPP-3 instances

There is a lack in the literature concerning the OPP-3 instances. The OPP-3 instances [BKRS09, BR13] are to our knowledge the single open published instances. They were generated by similar principles as in [BR13]. Here we test the proposed algorithms on these instances and compare the results with the interval graph algorithm from that paper.

The test package consists of 1260 instances of OPP-3, both infeasible and feasible ones as well. The package is divided into three classes with different maximal items side ratios $r_{\max}$, i.e., $r_{\max} = 1, 3, 20$. Each of these classes is further divided into subclasses according to the waste ratio from 0 to 40 with step 2. Container is a cube with side length 1000. Number of items $m = 15, 20$.

Tables 4.1–4.5 show the number of proven feasible, proven infeasible, and unsolved instances for each waste class. For the solution of the instances from [BR13], the authors limited the time up to 1 minute but took a faster computer. We limit the time by 5 min, so the comparison remains fair.

Since it does not make sense to compare the number of nodes, the efficiency indicators for the algorithms are the number (or ratio) of proven instances and the overall mean time for the solution.

4.7 Conclusions

Here we proposed constraint programming approaches for 3-dimensional orthogonal packing problems and adapted linear programming-based pruning rules of different types into the constraint propagation process of the constraint programming.

The main theoretical and experimental observations are the following.

- The relations strategy is very effective for instances with the maximal relation of item sides greater than 1, so not cubes.
- On average and in particular the relations strategy is stronger than the algorithm from [BR13] even without using LP-based pruning rules with the constraint propagation.
- The schedule-or-postpone strategy is very bad at finding a feasible solution, if there exists at least one.

- The 1D bar relaxation from a contour increases the efficiency of the schedule-or-postpone strategy but the overall efficiency remains not sufficient.
- The 2D slice relaxation is very week relaxation, so it prunes non of the subproblems while the solution of the test instances.
- The 1D slice-bar relaxation with forbidden pairs effects instances with a greater items sizes ratio.

4.8 Acknowledgments

We thank François Clautiaux for the kindly provided code of the original algorithm; David Pisinger for the provided code for the solution of 0-1 knapsack problems. We appreciate the Academic Initiative of IBM which enables many researchers all over the world to compare their methods using state-of-the-art IBM ILOG Optimization Software.

Table 4.1: Self-generated instances: $W = H = 1000$, $r_{\max} = 1, 3, 20$, number of instances is 630 for each $m = 15, 20$. Column w.% indicates the waste ratio (%), fe the amount of proven feasible instances, in the amount of proven infeasible instances, un the amount of unsolved instances, n the number of nodes, and t the total time for the solution (in seconds).

		Algorithm from [BR13]										RELATIONS3									
		$m = 15$					$m = 20$					$m = 15$					$m = 20$				
$r_{\max}$	w.%	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t
1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	2	0	0	9	1	739914	21
	2	0	10	0	1	0	0	10	0	1	0	0	10	0	22349	1	0	10	0	1432273	40
	4	0	10	0	1	0	0	10	0	1	0	0	10	0	2	0	0	8	2	2	0
	6	0	10	0	1	0	0	10	0	1	0	0	10	0	2	0	0	9	1	9720029	257
	8	0	10	0	1	0	0	10	0	1	0	0	10	0	364865	10	0	9	1	1505353	39
	10	0	10	0	1	0	0	10	0	1	0	0	10	0	989541	25	0	7	3	4615989	121
	12	0	10	0	1	0	0	10	0	1	0	0	10	0	3427735	85	0	8	2	1581705	46
	14	0	10	0	1	0	0	10	0	1	0	0	10	0	2505817	61	0	6	4	2951354	77
	16	0	10	0	1	0	0	10	0	1	0	0	10	0	2	0	0	4	6	13130204	349
	18	0	10	0	1	0	0	9	1	1	0	0	10	0	2	0	0	5	5	7570159	199
	20	0	10	0	1	0	0	10	0	1	0	0	10	0	3308180	80	0	7	3	2	0
	22	0	9	1	1	0	0	9	1	1	0	0	8	2	8096733	196	1	3	6	7807724	216
	24	0	10	0	1	0	0	6	4	1	0	0	8	2	2	0	2	3	5	5282384	152
	26	1	9	0	8256	16	0	6	4	1	0	1	8	1	3393703	83	2	0	8	2779	0
	28	0	10	0	1	0	0	4	6	1	0	0	9	1	6880317	167	3	2	5	209	0
	30	1	7	2	8728	16	0	8	2	1	0	1	6	3	1802348	44	0	2	8	2	0
	32	1	7	2	4103	7	0	5	5	1	0	3	5	2	4308676	105	4	0	6	6058	0
	34	4	4	2	24708	45	0	4	6	1	0	6	3	1	46676	1	6	2	2	614993	21
	36	4	5	1	35214	68	0	2	8	1	0	5	3	2	124	0	8	1	1	2686	0
	38	4	5	1	14768	26	0	0	10	-	-	5	4	1	133	0	10	0	0	373	0
	40	6	4	0	21999	38	2	3	5	93325	63	6	4	0	124	0	6	3	1	261	0
		21	180	9	5609	10	2	156	52	4667	3	27	168	15	1673683	41	42	98	70	2712593	73
3	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	18	0
	2	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	38	0
	4	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	712	0
	6	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	3240	0
	8	0	10	0	1	0	0	10	0	1	0	0	10	0	2	0	0	10	0	785601	22
	10	0	10	0	1	0	0	8	2	1	0	0	10	0	37	0	0	10	0	339575	9
	12	0	10	0	5504	7	0	5	5	1	0	0	10	0	43	0	0	10	0	1236393	31
	14	0	10	0	1	0	0	4	6	1	0	0	10	0	3	0	0	8	2	37375	1
	16	0	10	0	1	0	0	3	7	1	0	0	10	0	8	0	1	9	0	651706	17
	18	0	10	0	1491	1	0	4	6	1	0	0	10	0	1622	0	2	8	0	2049036	54
	20	0	5	5	223	0	0	2	8	1	0	2	8	0	986320	22	6	4	0	48010	1
	22	0	6	4	1	0	0	2	8	1	0	0	10	0	5063	0	6	3	1	295161	8
	24	0	3	7	1	0	0	0	10	-	-	3	7	0	45388	1	9	1	0	916731	26
	26	2	4	4	31195	50	0	0	10	-	-	5	5	0	4546	0	9	1	0	1826858	52
	28	2	3	5	8617	12	0	0	10	-	-	6	4	0	6345	0	10	0	0	2788	0
	30	4	2	4	37616	73	0	1	9	1	0	7	3	0	2431	0	9	1	0	292	0
	32	4	2	4	39973	70	0	1	9	1	0	8	2	0	171	0	9	1	0	1317	0
	34	10	0	0	19298	30	0	0	10	-	-	10	0	0	152	0	10	0	0	329	0
	36	7	2	1	13635	21	0	0	10	-	-	8	2	0	272	0	10	0	0	327	0
	38	9	0	1	47905	96	0	0	10	-	-	10	0	0	16708	0	10	0	0	321	0
	40	10	0	0	12282	21	1	0	9	57122	25	10	0	0	179	0	10	0	0	330	0
		48	127	35	10369	18	1	80	129	3809	2	69	141	0	50919	1	101	106	3	390293	11
20	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	2	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	4	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	6	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	7	0
	8	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	10	0	10	0	10673	13	0	10	0	1	0	0	10	0	1	0	0	10	0	8	0
	12	0	10	0	1	0	0	8	2	1	0	0	10	0	1	0	0	10	0	88	0
	14	0	10	0	8994	10	0	6	4	1	0	0	10	0	2	0	0	10	0	585	0
	16	0	10	0	1090	1	0	6	4	1	0	0	10	0	8	0	0	10	0	40	0
	18	0	8	2	12175	16	0	6	4	1	0	0	10	0	308	0	0	10	0	23	0
	20	0	8	2	2	0	0	5	5	1	0	0	10	0	988	0	1	8	1	61612	2
	22	0	8	2	1	0	0	4	6	1	0	0	10	0	120	0	4	6	0	47949	2
	24	0	8	2	1	0	0	7	3	1	0	0	10	0	58	0	1	9	0	303462	9
	26	2	4	4	13273	15	0	5	5	1	0	3	7	0	1983	0	3	7	0	46207	2
	28	0	6	4	10	0	0	3	7	1	0	1	9	0	3775	0	6	4	0	14729	0
	30	0	7	3	1	0	0	2	8	1	0	2	8	0	1484	0	8	2	0	77171	2
	32	2	2	6	46266	85	0	4	6	1	0	7	3	0	619	0	5	5	0	21454	1
	34	2	4	4	47807	68	0	2	8	1	0	6	4	0	87	0	8	1	1	671	0
	36	7	2	1	31620	50	0	0	10	-	-	8	2	0	92	0	10	0	0	4180	0
	38	4	5	1	12613	18	0	1	9	1	0	5	5	0	1598	0	9	1	0	44280	1
	40	5	4	1	16643	27	0	3	7	1	0	6	4	0	75	0	7	3	0	315	0
		22	156	32	9580	14	0	122	88	1	0	38	172	0	534	0	62	146	2	29656	1
		91	463	76	8519	14	3	358	269	2736	2	134	481	15	575045	14	205	350	75	1044181	28

Table 4.2: Comparison of the approaches ($m = 15$), $r_{\max} = 1, 3$.

		Algorithm [BR13]					Relations3					Relations3+LP-pair					Relations3+LP-bar					FixMin3					FixMin3+LP-cont				
$r_{\max}$	w.%	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t
1	0	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	2	0	10	0	1	0	0	10	0	22349	1	0	10	0	22336	12	0	10	0	1	0	0	10	0	1031	0	0	10	0	881	3
	4	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	6	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	8	0	10	0	1	0	0	10	0	364865	10	0	9	1	18799	30	0	10	0	27740	4	0	9	1	275078	17	0	8	2	1	0
	10	0	10	0	1	0	0	10	0	989541	25	0	8	2	1	0	0	10	0	989541	103	0	9	1	993848	57	0	8	2	1	0
	12	0	10	0	1	0	0	10	0	3427735	85	0	7	3	1	0	0	8	2	754147	67	0	7	3	1	0	0	7	3	1	0
	14	0	10	0	1	0	0	10	0	2505817	61	0	8	2	1	0	0	9	1	583396	51	0	8	2	1	0	0	8	2	1	0
	16	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	1	0
	18	0	10	0	1	0	0	10	0	2	0	0	10	0	2	0	0	10	0	2	0	0	10	0	1	0	0	10	0	1	0
	20	0	10	0	1	0	0	10	0	3308180	80	0	8	2	1	0	0	9	1	655918	39	0	8	2	1	0	0	8	2	1	0
	22	0	9	1	1	0	0	8	2	8096733	196	0	4	6	1	0	0	7	3	4433651	315	0	4	6	1	0	0	4	6	1	0
	24	0	10	0	1	0	0	8	2	2	0	0	8	2	2	0	0	8	2	2	0	0	8	2	1	0	0	8	2	1	0
	26	1	9	0	8256	16	1	8	1	3393703	83	1	6	3	33	0	1	7	2	630653	52	0	6	4	1	0	0	6	4	1	0
	28	0	10	0	1	0	0	9	1	6880317	167	0	6	4	2	0	0	6	4	2	0	0	6	4	1	0	0	6	4	1	0
	30	1	7	2	8728	16	1	6	3	1802348	44	1	5	4	37	0	1	5	4	37	0	0	5	5	1	0	0	5	5	1	0
	32	1	7	2	4103	7	3	5	2	4308676	105	3	4	3	93	0	3	4	3	93	0	0	4	6	1	0	0	4	6	1	0
	34	4	4	2	24708	45	6	3	1	46676	1	6	3	1	46676	31	6	3	1	46676	4	0	3	7	1	0	0	3	7	1	0
	36	4	5	1	35214	68	5	3	2	124	0	5	3	2	123	0	5	3	2	124	0	0	3	7	1	0	0	3	7	1	0
	38	4	5	1	14768	26	5	4	1	133	0	5	4	1	132	0	5	4	1	133	0	0	4	6	1	0	0	4	6	1	0
	40	6	4	0	21999	38	6	4	0	124	0	6	4	0	124	0	6	4	0	124	0	0	4	6	1	0	0	4	6	1	0
		21	180	9	5609	10	27	168	15	1673683	41	27	147	36	4208	4	27	157	26	386774	30	0	148	62	60475	4	0	146	64	43	0
3	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	2	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	4	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	6	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	8	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	2	0	0	10	0	1	0	0	10	0	1	0
	10	0	10	0	1	0	0	10	0	37	0	0	10	0	35	0	0	10	0	36	0	0	9	1	700664	34	0	10	0	10523	72
	12	0	10	0	5504	7	0	10	0	43	0	0	10	0	18	0	0	10	0	43	0	0	8	2	1	0	0	9	1	1	0
	14	0	10	0	1	0	0	10	0	3	0	0	10	0	1	0	0	10	0	3	0	0	7	3	2	0	0	10	0	1	0
	16	0	10	0	1	0	0	10	0	8	0	0	10	0	2	0	0	10	0	8	0	0	7	3	10380	0	0	10	0	2	0
	18	0	10	0	1491	1	0	10	0	1622	0	0	10	0	1618	1	0	10	0	1621	0	0	6	4	1	0	0	7	3	1	0
	20	0	5	5	223	0	2	8	0	986320	22	1	8	1	7503	8	1	8	1	7505	1	0	5	5	2	0	0	7	3	2029	24

The table is continued on the next page.

Table 4.3: Comparison of the approaches ($m = 15$), $r_{\max} = 3, 20$. Continuation.

		Algorithm [BR13]					Relations3					Relations3+LP-pair					Relations3+LP-bar					FixMin3					FixMin3+LP-cont				
$r_{\max}$	w.%	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t
	22	0	6	4	1	0	0	10	0	5063	0	0	10	0	4735	9	0	10	0	5063	1	0	8	2	12063	0	0	8	2	12063	35
	24	0	3	7	1	0	3	7	0	45388	1	3	7	0	40067	62	3	7	0	45388	3	0	2	8	1	0	0	3	7	1	0
	26	2	4	4	31195	50	5	5	0	4546	0	5	5	0	4482	5	5	5	0	4546	1	0	6	4	1	0	0	6	4	1	0
	28	2	3	5	8617	12	6	4	0	6345	0	6	4	0	6335	9	6	4	0	6345	1	0	2	8	1	0	0	3	7	1	0
	30	4	2	4	37616	73	7	3	0	2431	0	7	3	0	2430	1	7	3	0	2431	0	0	1	9	1	0	0	2	8	1	0
	32	4	2	4	39973	70	8	2	0	171	0	8	2	0	170	0	8	2	0	171	0	0	1	9	1	0	0	2	8	1	0
	34	10	0	0	19298	30	10	0	0	152	0	10	0	0	150	0	10	0	0	152	0	0	0	10	-	-	0	0	10	-	-
	36	7	2	1	13635	21	8	2	0	272	0	8	2	0	271	0	8	2	0	272	0	0	1	9	1	0	1	2	7	820	0
	38	9	0	1	47905	96	10	0	0	16708	0	10	0	0	16706	58	10	0	0	16708	3	0	0	10	-	-	0	0	10	-	-
	40	10	0	0	12282	21	10	0	0	179	0	10	0	0	179	0	10	0	0	179	0	0	0	10	-	-	0	0	10	-	-
		48	127	35	10369	18	69	141	0	50919	1	68	141	1	4034	7	68	141	1	4308	1	0	113	97	40174	2	1	129	80	1414	7
20	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	2	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	4	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	6	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	8	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	10	0	10	0	10673	13	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	12	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	14	0	10	0	8994	10	0	10	0	2	0	0	10	0	2	0	0	10	0	2	0	0	10	0	3	0	0	10	0	2	0
	16	0	10	0	1090	1	0	10	0	8	0	0	10	0	7	0	0	10	0	8	0	0	9	1	3	0	0	10	0	5	0
	18	0	8	2	12175	16	0	10	0	308	0	0	10	0	293	1	0	10	0	308	0	0	7	3	1280	0	0	9	1	997	8
	20	0	8	2	2	0	0	10	0	988	0	0	10	0	929	2	0	10	0	988	0	0	8	2	1	0	0	9	1	26	0
	22	0	8	2	1	0	0	10	0	120	0	0	10	0	14	0	0	10	0	120	0	0	7	3	1	0	0	8	2	5	0
	24	0	8	2	1	0	0	10	0	58	0	0	10	0	17	0	0	10	0	58	0	0	6	4	1	0	0	9	1	1	0
	26	2	4	4	13273	15	3	7	0	1983	0	3	7	0	1787	3	3	7	0	1983	0	0	3	7	1	0	1	4	5	46973	3
	28	0	6	4	10	0	1	9	0	3775	0	1	9	0	3767	2	1	9	0	3775	0	0	4	6	2433	0	0	8	2	1217	6
	30	0	7	3	1	0	2	8	0	1484	0	2	8	0	1467	3	2	8	0	1484	0	0	4	6	1	0	0	7	3	4	0
	32	2	2	6	46266	85	7	3	0	619	0	7	3	0	585	1	7	3	0	619	0	0	2	8	1	0	0	3	7	591	6
	34	2	4	4	47807	68	6	4	0	87	0	6	4	0	85	0	6	4	0	87	0	1	2	7	107	0	0	4	6	7	0
	36	7	2	1	31620	50	8	2	0	92	0	8	2	0	78	1	8	2	0	92	0	0	0	10	-	-	1	2	7	8063	1
	38	4	5	1	12613	18	5	5	0	1598	0	5	5	0	60	0	5	5	0	1598	0	0	3	7	4	0	1	4	5	6160	159
	40	5	4	1	16643	27	6	4	0	75	0	6	4	0	63	0	6	4	0	75	0	0	2	8	1	0	0	4	6	1	0
		22	156	32	9580	14	38	172	0	534	0	38	172	0	436	1	38	172	0	534	0	1	137	72	192	0	3	161	46	3050	9
		91	463	76	8519	14	134	481	15	575045	14	133	460	37	2893	4	133	470	27	130539	10	1	398	231	33846	2	4	436	190	1507	5

Table 4.4: Comparison of the approaches ($m = 20$), $r_{\max} = 1, 3$.

		Algorithm [BR13]					Relations3					Relations3+LP-pair					Relations3+LP-bar					FixMin3					FixMin3+LP-cont				
$r_{\max}$	w.%	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t
1	0	0	10	0	1	0	0	9	1	739914	21	0	8	2	1	0	0	10	0	1	0	0	8	2	1	0	0	8	2	1	0
	2	0	10	0	1	0	0	10	0	1432273	40	0	8	2	1	0	0	10	0	1	0	0	8	2	1	0	0	8	2	1	0
	4	0	10	0	1	0	0	8	2	2	0	0	8	2	1	0	0	10	0	1	0	0	8	2	1	0	0	8	2	1	0
	6	0	10	0	1	0	0	9	1	9720029	257	0	3	7	2	0	0	8	2	329235	46	0	3	7	1	0	0	3	7	1	0
	8	0	10	0	1	0	0	9	1	1505353	39	0	8	2	2	0	0	9	1	2	0	0	8	2	1	0	0	8	2	1	0
	10	0	10	0	1	0	0	7	3	4615989	121	0	6	4	2	0	0	8	2	2	0	0	6	4	1	0	0	6	4	1	0
	12	0	10	0	1	0	0	8	2	1581705	46	0	7	3	1	0	0	7	3	2	0	0	7	3	1	0	0	7	3	1	0
	14	0	10	0	1	0	0	6	4	2951354	77	0	5	5	2	0	0	5	5	2	0	0	5	5	1	0	0	5	5	1	0
	16	0	10	0	1	0	0	4	6	13130204	349	0	1	9	2	0	0	2	8	1476112	77	0	1	9	1	0	0	1	9	1	0
	18	0	9	1	1	0	0	5	5	7570159	199	0	3	7	2	0	0	3	7	2	0	0	3	7	1	0	0	3	7	1	0
	20	0	10	0	1	0	0	7	3	2	0	0	7	3	2	0	0	7	3	2	0	0	7	3	1	0	0	7	3	1	0
	22	0	9	1	1	0	1	3	6	7807724	216	0	2	8	2	0	1	2	7	324894	41	0	2	8	1	0	0	2	8	1	0
	24	0	6	4	1	0	2	3	5	5282384	152	0	3	7	2	0	0	3	7	2	0	0	3	7	1	0	0	3	7	1	0
	26	0	6	4	1	0	2	0	8	2779	0	2	0	8	2779	2	2	0	8	2779	0	0	0	10	-	-	0	0	10	-	-
	28	0	4	6	1	0	3	2	5	209	0	3	2	5	209	1	3	2	5	209	0	0	2	8	1	0	0	2	8	1	0
	30	0	8	2	1	0	0	2	8	2	0	0	2	8	2	0	0	2	8	2	0	0	2	8	1	0	0	2	8	1	0
	32	0	5	5	1	0	4	0	6	6058	0	4	0	6	6058	6	4	0	6	6058	1	0	0	10	-	-	0	0	10	-	-
	34	0	4	6	1	0	6	2	2	614993	21	6	2	2	614993	70	6	2	2	614993	22	0	2	8	1	0	0	2	8	1	0
	36	0	2	8	1	0	8	1	1	2686	0	8	1	1	2686	1	8	1	1	2686	0	0	1	9	1	0	0	1	9	1	0
	38	0	0	10	-	-	10	0	0	373	0	10	0	0	373	1	10	0	0	373	0	0	0	10	-	-	0	0	10	-	-
	40	2	3	5	93325	63	6	3	1	261	0	6	3	1	261	0	6	3	1	261	0	0	3	7	1	0	0	3	7	1	0
		2	156	52	4667	3	42	98	70	2712593	73	39	79	92	29875	4	40	94	76	131315	9	0	79	131	1	0	0	79	131	1	0
3	0	0	10	0	1	0	0	10	0	18	0	0	10	0	16	0	0	10	0	1	0	0	10	0	1515338	67	0	10	0	219	1
	2	0	10	0	1	0	0	10	0	38	0	0	10	0	36	0	0	10	0	1	0	0	9	1	252579	15	0	9	1	3970	6
	4	0	10	0	1	0	0	10	0	712	0	0	10	0	696	2	0	10	0	648	0	0	9	1	182384	15	0	8	2	2952	38
	6	0	10	0	1	0	0	10	0	3240	0	0	10	0	2987	8	0	10	0	3223	1	0	6	4	604379	23	0	7	3	1	0
	8	0	10	0	1	0	0	10	0	785601	22	0	9	1	1560	3	0	10	0	1566	0	0	7	3	1	0	0	7	3	1	0
	10	0	8	2	1	0	0	10	0	339575	9	0	9	1	8728	32	0	10	0	338355	63	0	3	7	1	0	0	5	5	1	0
	12	0	5	5	1	0	0	10	0	1236393	31	0	9	1	4862	15	0	9	1	4947	1	0	3	7	9734	0	0	5	5	22	1
	14	0	4	6	1	0	0	8	2	37375	1	0	8	2	37144	91	0	8	2	37375	5	0	3	7	1	0	0	3	7	1	0
	16	0	3	7	1	0	1	9	0	651706	17	0	6	4	11306	25	1	9	0	651706	129	0	3	7	1	0	0	4	6	2	0
	18	0	4	6	1	0	2	8	0	2049036	54	0	5	5	6154	19	1	7	2	534886	68	0	3	7	1	0	0	5	5	727	11
	20	0	2	8	1	0	6	4	0	48010	1	6	4	0	91006	14	6	4	0	48010	5	0	1	9	1	0	0	2	8	589	15

The table is continued on the next page.

Table 4.5: Comparison of the approaches ($m = 20$), $r_{\max} = 3, 20$. Continuation.

		Algorithm [BR13]					Relations3					Relations3+LP-pair					Relations3+LP-bar					FixMin3					FixMin3+LP-cont				
$r_{\max}$	w.%	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t	fe	in	un	n	t
	22	0	2	8	1	0	6	3	1	295161	8	4	3	3	323126	94	6	3	1	295161	27	0	2	8	1	0	0	3	7	1	0
	24	0	0	10	-	-	9	1	0	916731	26	7	1	2	17073	1	9	1	0	916731	96	0	1	9	1	0	0	1	9	1	0
	26	0	0	10	-	-	9	1	0	1826858	52	8	1	1	55229	5	8	1	1	19182	3	0	3	7	1	0	0	3	7	1	0
	28	0	0	10	-	-	10	0	0	2788	0	10	0	0	9482	1	10	0	0	2788	0	0	2	8	1	0	0	2	8	1	0
	30	0	1	9	1	0	9	1	0	292	0	9	1	0	305	0	9	1	0	292	0	0	0	10	-	-	0	1	9	1	0
	32	0	1	9	1	0	9	1	0	1317	0	9	1	0	20022	1	9	1	0	1317	0	0	2	8	70	0	0	2	8	1	0
	34	0	0	10	-	-	10	0	0	329	0	10	0	0	334	0	10	0	0	329	0	0	0	10	-	-	0	0	10	-	-
	36	0	0	10	-	-	10	0	0	327	0	10	0	0	331	0	10	0	0	327	0	0	0	10	-	-	0	0	10	-	-
	38	0	0	10	-	-	10	0	0	321	0	10	0	0	328	0	10	0	0	321	0	0	0	10	-	-	0	0	10	-	-
	40	1	0	9	57122	25	10	0	0	330	0	10	0	0	336	0	10	0	0	330	0	0	0	10	-	-	0	0	10	-	-
		1	80	129	3809	2	101	106	3	390293	11	93	97	20	28146	15	99	104	7	136071	19	0	67	143	160281	7	0	77	133	499	4
20	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	2	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	4	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	3	0	0	10	0	1	0
	6	0	10	0	1	0	0	10	0	7	0	0	10	0	1	0	0	10	0	5	0	0	10	0	544	0	0	10	0	1	0
	8	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0	0	10	0	1	0
	10	0	10	0	1	0	0	10	0	8	0	0	10	0	6	0	0	10	0	5	0	0	7	3	1409	0	0	8	2	1019	48
	12	0	8	2	1	0	0	10	0	88	0	0	10	0	99	0	0	10	0	88	0	0	10	0	44735	3	0	9	1	1	0
	14	0	6	4	1	0	0	10	0	585	0	0	10	0	655	0	0	10	0	585	0	0	6	4	6	0	0	7	3	122	6
	16	0	6	4	1	0	0	10	0	40	0	0	10	0	36	0	0	10	0	40	0	0	7	3	71	0	0	9	1	26	1
	18	0	6	4	1	0	0	10	0	23	0	0	10	0	22	0	0	10	0	23	0	0	7	3	62	0	0	7	3	1	0
	20	0	5	5	1	0	1	8	1	61612	2	1	8	1	198123	8	1	8	1	61612	5	0	2	8	1	0	0	6	4	801	68
	22	0	4	6	1	0	4	6	0	47949	2	4	6	0	65568	8	4	6	0	47949	5	0	4	6	4	0	0	7	3	396	13
	24	0	7	3	1	0	1	9	0	303462	9	0	9	1	451	0	0	9	1	409	0	0	3	7	1	0	0	5	5	1	0
	26	0	5	5	1	0	3	7	0	46207	2	2	7	1	5445	1	3	7	0	46207	7	0	3	7	1	0	0	5	5	1	0
	28	0	3	7	1	0	6	4	0	14729	0	6	4	0	34678	2	6	4	0	14729	1	0	2	8	1	0	0	4	6	1	0
	30	0	2	8	1	0	8	2	0	77171	2	8	2	0	136835	6	8	2	0	77171	4	0	2	8	1	0	0	3	7	1	0
	32	0	4	6	1	0	5	5	0	21454	1	5	5	0	29269	1	5	5	0	21454	3	0	0	10	-	-	0	3	7	1	0
	34	0	2	8	1	0	8	1	1	671	0	8	2	0	639	0	8	1	1	671	0	0	3	7	1	0	0	5	5	1	0
	36	0	0	10	-	-	10	0	0	4180	0	10	0	0	5952	1	10	0	0	4180	1	0	0	10	-	-	0	0	10	-	-
	38	0	1	9	1	0	9	1	0	44280	1	9	1	0	103436	5	9	1	0	44280	2	0	0	10	-	-	0	1	9	1	0
	40	0	3	7	1	0	7	3	0	315	0	7	3	0	304	0	7	3	0	315	0	0	1	9	1	0	0	2	8	1	0
		0	122	88	1	0	62	146	2	29656	1	60	147	3	27691	2	61	146	3	15225	1	0	107	103	2602	0	0	131	79	119	7
		3	358	269	2736	2	205	350	75	1044181	28	192	323	115	28571	7	200	344	86	94204	10	0	253	377	50218	2	0	287	343	198	4

Summary and Outlook

The thesis is devoted to the $\mathcal{NP}$-hard problems of higher-dimensional orthogonal packing and related problems. The main problems we solve here are the two- and three-dimensional strip packing and feasibility, and the one-dimensional contiguous bin packing.

Given an $\mathcal{NP}$-hard problem of integer linear programming, it is extremely important to find the facets of the convex hull of its integer solutions. In the unlikely case, if the full description of the convex hull is found, then the problem becomes polynomially solvable. Even if some facets of the convex hull are found, it helps to tighten the linear programming relaxation, which leads to the better stability, efficiency, and performance characteristics of the solution methods. Thus, finding the facets is significant, both from theoretical and practical perspectives. One successful attempt in this direction of a search for facets for a particular model of the two-dimensional strip packing and hence feasibility problems is done in the thesis. We proposed a new integer linear programming model and found two classes of facets under appropriate assumptions. As predicted, the new branch-and-cut approach has better stability, efficiency, and performance characteristics.

In the discrete optimization the development of models with tighter linear programming bounds has a crucial significance, since it impacts the stability, efficiency and performance characteristics of the solution methods. If possible, Dantzig-Wolfe decomposition is a classical tool to make an integer linear programming model stronger. But, it leads to a different formulation where sometimes the properties of the solved problem are difficult to deliver. In the thesis, we considered this decomposition for the one-dimensional contiguous bin packing problem and tackled the difficulty related to the heterogeneity and contiguity constraints. As a result, we came up with new branch-and-price approaches, which have a very good efficiency, and have a great potential for the further development and propagation to higher-dimensions.

Nonlinear modeling of the higher-dimensional packing process is more natural and less complicated. It requires an appropriate modeling tool. Recently, the leading role of such a tool for discrete optimization problems has been acquiring constraint programming. Being a synergy, it inherits the results in mathematics, operations research, and computer science. One of a few weaknesses of constraint programming is a week relaxation bound. This we accomplished to excel in the thesis by tightening the constraint propagation with the multilevel relaxations – smaller-dimensional relaxation of the solved problem first, Dantzig-Wolfe decomposition next, and linear programming relaxation at the end. We proposed new constraint programming approaches for the two- and three-dimensional strip packing and feasibility problems, which are distin-

guished by different solution search strategies and a tightened constraint propagation, yielding very good numerical results.

Concerning the thesis, the following research directions seem to be of great interest:

- What further facets exist? This question is of highly interest from the theoretical and practical perspectives. However, we must admit that the number of facet classes can be polynomially unbounded.
- The propagation of the branch-and-price approach to higher-dimensions can be possibly done by branchings. Tackling all dimensions in one model leads to nonlinearity in the constraint set.

The higher-dimensional orthogonal packing and related problems considered in the thesis remain still very hard, despite our achieved results. It is explained by the fact that they are $\mathcal{NP}$-hard. Therefore, the following two issues are relevant. From practice, the effectiveness of a solution method for a problem is always a trade-off between the generality of the method and a deep investigation of the concrete problem structure. From theory, we still do not know whether $\mathcal{P} \neq \mathcal{NP}$, but in both resolutions of this problem the modeling issues and exact approaches will remain up-to-date.

Bibliography

[Apt03] K. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003. 71, 112

[AV08] C. Alves and J. M. Valério de Carvalho. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers and Operations Research*, 35(4):1315–1328, 2008. 84, 85, 118, 119

[AVPT09] R. Alvarez-Valdes, F. Parreño, and J. Tamarit. A branch and bound algorithm for the strip packing problem. *OR Spectrum*, 31:431–459, 2009. 10.1007/s00291-008-0128-5. 65, 108

[BB07] R. Baldacci and M. Boschetti. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *European Journal of Operational Research*, 183(3):1136–1149, December 2007. 8, 65, 66, 108

[BCDP11] N. Beldiceanu, M. Carlsson, S. Demassey, and E. Poder. New filtering for the cumulative constraint in the context of non-overlapping rectangles. *Annals of Operations Research*, 184:27–50, 2011. 10.1007/s10479-010-0731-0. 66, 75, 108

[Bea85] J. Beasley. Bounds for two-dimensional cutting. *Journal of the Operational Research Society*, 36(1):71–74, 1985. 8, 66, 108

[Bel10] G. Belov. Imposing non-preemptiveness in resource-constrained problems using linear programming and the consecutive-ones property. In *Tagungsband 19. Workshop für Diskrete Optimierung*, pages 9–12, 2010. 49

[BKRS09] G. Belov, V. Kartak, H. Rohling, and G. Scheithauer. One-dimensional relaxations and LP bounds for orthogonal packing. *International Transactions in Operational Research*, 16(6):745–766, 2009. 8, 66, 67, 68, 108, 109, 116, 124

[BKRS13] G. Belov, V. Kartak, H. Rohling, and G. Scheithauer. Conservative scales in packing problems. *OR Spectrum*, 35(2):505–542, 2013. 67, 100, 109

[BL76] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335 – 379, 1976. 47, 49

[BM03] M. Boschetti and A. Mingozzi. The two-dimensional finite bin packing problem. part I: New lower bounds for the oriented case. *4OR: A Quarterly Journal of Operations Research*, 1:27–42, 2003. 10.1007/s10288-002-0005-z. 71, 111

[BR13] G. Belov and H. Rohling. LP bounds in an interval-graph algorithm for orthogonal-packing feasibility. *Operations Research*, 61(2):483–497, 2013. 67, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 109, 124, 126, 127, 128, 129, 130, 139, 140

[BSM08] G. Belov, G. Scheithauer, and E. Mukhacheva. One-dimensional heuristics adapted for two-dimensional rectangular strip packing. *Journal of the Operational Research Society*, 59(10):823–832(10), June 2008. 72, 117

[CAVdC10a] F. Clautiaux, C. Alves, and J. Valério de Carvalho. A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179:317–342, 2010. 67, 109

[CAVdC10b] F. Clautiaux, C. Alves, and J. Valério de Carvalho. A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179:317–342, 2010. 10.1007/s10479-008-0453-8. 67

[CCM07] J. Carlier, F. Clautiaux, and A. Moukrim. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Computers and Operations Research*, 34(8):2223–2250, 2007. 67, 71, 111

[CJCM08] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers and Operations Research*, 35(3):944–959, 2008. 65

[CJM08] F. Clautiaux, A. Jouglet, and A. Moukrim. A new graph-theoretical model for k-dimensional guillotine-cutting problems. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 43–54. Springer Berlin Heidelberg, 2008. 5, 38, 39, 63, 65, 66, 68, 69, 98, 99, 100, 102, 107, 108, 110, 112, 113, 139

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Mass, Cambridge, 1990. 56, 57

[CMM03] R. Conway, W. Maxwell, and L. Miller. *Theory of Scheduling*. Dover Books on Computer Science Series. Dover, 2003. 42

[Dom09] M. Dom. *Recognition, generation, and application of binary matrices with the consecutive-ones property*. PhD thesis, University of Jena, 2009. 49

[DW60] G. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. 42

[FS01] S. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91:11–31, 2001. 10.1007/s101070100243. 67

[FS04a] S. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29(2):353–368, 2004. 65, 67, 108, 109

[FS04b] S. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60(2):311–329, 2004. 67, 109

[FSvdV07] S. Fekete, J. Schepers, and J. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587, 2007. 7

[GG61] P. Gilmore and R. Gomory. A linear programming approach to the cutting-stock problem (Part I). *Operations Research*, 9:849–859, 1961. 42, 67, 84, 109, 116, 118, 119

[GG63] P. Gilmore and R. Gomory. A linear programming approach to the cutting-stock problem (Part II). *Operations Research*, 11:863–888, 1963. 42, 67, 84, 85, 109, 116, 118, 119

[Hif98] M. Hifi. Exact algorithms for the guillotine strip cutting/packing problem. *Computers and Operations Research*, 25(11):925–940, 1998. 65, 108

[HNS08] J. Hardin, G. Nemhauser, and M. Savelsbergh. Strong valid inequalities for the resource-constrained scheduling problem with uniform resource requirements. *Discrete Optimization*, 5(1):19 – 35, 2008. 8, 38, 45

[Hop00] E. Hopper. *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. PhD thesis, Cardiff University, 2000. 38, 40, 102, 106, 139, 140

[HT00] E. Hopper and B. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1):34–57, 2000. 38, 40, 102, 106, 139, 140

[Kan39] L. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, July 1960. (translation, firstly appeared in Russian in 1939). 6

[KIN+09] M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, and H. Nagamochi. Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research*, 198(1):73–83, 2009. 65, 108

[KMP10] R. Korf, M. Moffitt, and M. Pollack. Optimal rectangle packing. *Annals of Operations Research*, 179:261–295, 2010. 10.1007/s10479-008-0463-6. 66, 108

[KZ51] L. Kantorovich and V. Zalgaller. *Calculation of rational cutting of stock (in Russian)*. Lenizdat, Leningrad, 1951. 67, 84, 85, 109, 116, 118, 119

[LMV02] A. Lodi, S. Martello, and D. Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123:379–396, 2002. 5

[Lue83] G. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In *Foundations of Computer Science'83*, pages 289–297, 1983. 67

[MAVdC10] R. Macedo, C. Alves, and J. Valério de Carvalho. Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers and Operations Research*, 37:991–1001, June 2010. 5, 65, 107

[MMBS11] M. Mesyagutov, E. Mukhacheva, G. Belov, and G. Scheithauer. Packing of one-dimensional bins with contiguous selection of identical items: An exact method of optimal solution. *Automation and Remote Control*, 72:141–159, 2011. 10.1134/S0005117911010127. v, 8, 45, 71, 72, 73, 74, 111, 117

[MMV03] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip packing problem. *INFORMS Journal on Computing*, 15(3):310–319, 2003. 27, 38, 40, 73, 139

[MPT99] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45:414–423, 1999. 37, 98, 123

[MSB12a] M. Mesyagutov, G. Scheithauer, and G. Belov. LP bounds in various constraint programming approaches for orthogonal packing. *Computers and Operations Research*, 39(10):2425–2438, 2012. v, 108, 109, 110, 112

[MSB12b] M. Mesyagutov, G. Scheithauer, and G. Belov. New constraint programming approaches for 3D orthogonal packing. Technical report, Preprint MATH-NM-01-2012, Technische Universität Dresden, 2012. v, 123

[MSB13a] M. Mesyagutov, G. Scheithauer, and G. Belov. A new branch-and-cut method for the strip packing problem. Technical report, Preprint MATH-NM-04-2013, Technische Universität Dresden, 2013. v, 42, 62

[MSB13b] M. Mesyagutov, G. Scheithauer, and G. Belov. New branch-and-price methods for the 1D contiguous bin packing problem. Technical report, Preprint MATH-NM-06-2013, Technische Universität Dresden, 2013. v, 61

[NW88] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, 1988. 8, 24, 43

[Pad00] M. Padberg. Packing small boxes into a big box. *Mathematical Methods of Operations Research*, 52(1):1–21, 2000. 8, 66, 94, 108

[PS07] D. Pisinger and M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS J. on Computing*, 19:36–51, January 2007. 65, 66, 108

[Sch95] G. Scheithauer. Equivalence and dominance for problems of optimal packing of rectangles. *Ricerca Operativa*, 83(27):3–34, 1995. 72, 73, 74, 116

[Sch99] G. Scheithauer. LP-based bounds for the container and multi-container loading problem. *International Transactions in Operational Research*, 6(2):199–213, 1999. 67, 68, 109, 116, 119, 121

[Sch08] G. Scheithauer. *Zuschnitt- und Packungsoptimierung: Problemstellungen, Modellierungstechniken, Lösungsmethoden*. Teubner Verlag, Wiesbaden, 2008. 32, 70, 110, 111

[Sim08] H. Simonis. Search strategies for rectangle packing. In *of Lecture Notes in Computer Science*, pages 52–66. Springer, 2008. 66, 68, 75, 77, 103, 108

[SO08] H. Simonis and B. O'Sullivan. Search strategies for rectangle packing. In *Proceedings of the 14th international conference on Principles and Practice of Constraint Programming*, CP '08, pages 52–66, Berlin, Heidelberg, 2008. Springer-Verlag. 66, 68, 75, 103, 108

[Tuc72] A. Tucker. A structure theorem for the consecutive 1's property. *Journal of Combinatorial Theory Series B*, 12:153–162, 1972. 44, 45, 49

[WHS07] G. Wäscher, H. Hausner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, December 2007. 84, 118

[Wol98] L. Wolsey. *Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Chichester, 1998. 24, 30, 31

List of Tables

List of Figures

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt ubernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ahnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Die vorgelegte Dissertation habe ich am Institut für Numerische Mathematik der Technischen Universitët Dresden unter der wissenschaftlichen Betreuung von Herrn Prof. Dr. rer. nat. Andreas Fischer und Herrn Dr. rer. nat. Guntram Scheithauer angefertigt.

Assurance

I affirm that I have written this dissertation without any inadmissible help from any third person and without recourse to any aids other than cited; all sources are clearly referenced. The dissertation has never been submitted in this or similar form before, neither in Germany nor in any foreign country.

I have written this dissertation at the Institute of Numerical Mathematics, Dresden University of Technology, under the scientific supervision of Prof. Dr. rer. nat. Andreas Fischer and Dr. rer. nat. Guntram Scheithauer.

Marat A. Mesyagutov
Dresden, August 31, 2013