

**International Series in  
Operations Research & Management Science**

**Guntram Scheithauer**

# **Introduction to Cutting and Packing Optimization**

**Problems, Modeling Approaches,  
Solution Methods**



 Springer

# **International Series in Operations Research & Management Science**

Volume 263

## **Series Editor**

Camille C. Price  
Stephen F. Austin State University, TX, USA

## **Associate Series Editor**

Joe Zhu  
Worcester Polytechnic Institute, MA, USA

## **Founding Series Editor**

Frederick S. Hillier  
Stanford University, CA, USA

More information about this series at <http://www.springer.com/series/6161>

Guntram Scheithauer

# Introduction to Cutting and Packing Optimization

Problems, Modeling Approaches,  
Solution Methods



Springer

Guntram Scheithauer  
Institute for Numerical Mathematics  
TU Dresden  
Dresden, Germany

ISSN 0884-8289                   ISSN 2214-7934 (electronic)  
International Series in Operations Research & Management Science  
ISBN 978-3-319-64402-8       ISBN 978-3-319-64403-5 (eBook)  
DOI 10.1007/978-3-319-64403-5

Library of Congress Control Number: 2017951441

© Springer International Publishing AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature  
The registered company is Springer International Publishing AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

The terms *allocation*, *packing*, and *cutting problem* comprise a wide variety of theoretical and practical problems. Although various formulations with different denotations appear frequently, there exist strong relations between them so that solution methods for cutting problems can also be applied for allocation and packing problems, and vice versa.

For example, on the one hand, any (two-dimensional) jigsaw puzzle can be considered as packing or allocation problem: *Is it possible to pack all given puzzle pieces into a predefined region? And, if yes, find a corresponding pattern.* On the other hand, the related cutting problem could be: *How to dissect (cut) a predefined region so that a set of desired pieces is obtained?*

In this book, we in particular deal with allocation, packing, and cutting problems in which an optimization component is of importance. In general, we differentiate between two kinds of problem statements which are formulated here as allocation problems:

- *Computation of an optimal (allocation) pattern*

In this first problem type, a region (container) and a set of smaller objects (pieces, items) are given, and a profit coefficient is assigned to each object. Then the task is to find a subset of the objects with maximal total profit such that all items of it can be placed simultaneously within the region while meeting some allocation constraints.

- *Computation of an optimal combination of (allocation) patterns*

In this case, a sufficiently large number of containers, each having a cost coefficient, and a set of smaller objects are given. Find a subset of the containers with minimal total costs such that all objects can be assigned to the chosen containers regarding related allocation constraints.

Problems of the first type aim, in general, to *maximize material utilization* whereas those of the second type have the *minimization of material usage* as objective.

Within the last decades, a very large number of investigations have been presented concerning most diversified applications and corresponding research work. In fact, it is impossible to encompass the wide variety of problem statements which

can be formulated as allocation, packing, or cutting problem. Therefore, attempts to classify all these topics are useful to obtain a better overview on related problem formulations and relationships between them. Important contributions in this respect are done in Dyckhoff [H. Dyckhoff, A typology of cutting and packing problems. *Eur. J. Oper. Res.* **44**(2), 145–159 (1990)], Dyckhoff and Finke [H. Dyckhoff, U. Finke, *Cutting and Packing in Production and Distribution* (Physica, Heidelberg, 1992)], and Wäscher and H. Haußner [G. Wäscher, H. Haußner, H. Schumann, An improved typology of cutting and packing problems. *Eur. J. Oper. Res.* **183**, 1109–1130 (2007)].

However, the concern of this book consists in giving an introduction to the treatment of cutting and packing problems. Therefore, we will consider several one-, two-, and three-dimensional basic problems, some of their formulations, related modeling issues, and solution approaches. In particular, we will address the following optimization problems arising in the field of cutting and packing:

- knapsack problems,
- cutting stock and bin packing problems,
- feasibility problems,
- optimal guillotine patterns of rectangular pieces,
- non-guillotine patterns of rectangular pieces,
- packing of rectangles into a strip,
- consideration of quality constraints,
- pallet and multi-pallet loading problems,
- container and multi-container loading problems,
- allocation of polygonal objects,
- circle and sphere packing.

As formulation variant of a considered problem we will optionally choose a description of it as an allocation, packing, or cutting problem. A translation into another formulation is obvious, in general. Frequently, the presentation of modeling issues and solution approaches is supplemented by some examples and exercises. Corresponding solutions are provided, too.

This book is addressed to everyone who is dealing with allocation, packing, or cutting of regular or irregular objects, and to those who intend to deal with such problems. In particular, this monograph is directed at students of mathematics, optimization, business or techno mathematics, operations research, and engineering. Moreover, our aim is to provide fundamental knowledge to researchers and practitioners which have to solve real-world allocation, packing, or cutting problems. Based on the presented modeling methods and the proposed solution strategies as well as numerous references to related problems, the manuscript should enable the reader to successfully tackle further new and more complex cutting and packing problems.

This book is a result of a long-time work on the field of cutting and packing, and of several lectures on this topic. It constitutes a major extension of the earlier German version [G. Scheithauer, *Zuschnitt- und Packungsoptimierung* (Vieweg + Teubner, Wiesbaden, 2008)]. The ingredients of the book are based on many

theoretically oriented investigations and experience acquired in numerous practical projects.

I am deeply grateful to Johannes Terno who introduced me to the fields of optimization and cutting and packing. My sincere thanks go to many researchers and practitioners who were and/or are working together with me on a plurality of very interesting topics within that area. In particular, I would like to thank again Jürgen Rietz for his help to finish the German predecessor of this manuscript, for which John Martinovic gave very valuable support. Not least, I thank my dear wife Monika for her understanding and active help.

Dresden, Germany  
May 2017

Guntram Scheithauer

# Contents

|          |   |    |
|----------|---|----|
| <b>1</b> | <b>Modeling</b>                                   | 1  |
| 1.1      | Set-Theoretical Models                            | 1  |
| 1.2      | Representation of Objects                         | 5  |
| 1.3      | Representation of Patterns                        | 7  |
| 1.4      | Approximation and Internal Description of Objects | 8  |
| 1.5      | A Nonlinear Optimization Model, $\Phi$ -Functions | 9  |
| 1.6      | Integer Linear Programming Models                 | 12 |
| 1.7      | Further Opportunities of Modeling                 | 14 |
| 1.8      | Exercises   | 14 |
| 1.9      | Solutions   | 15 |
|          | References  | 17 |
| <b>2</b> | <b>Knapsack Problems</b>                          | 19 |
| 2.1      | Problem Statement                                 | 19 |
| 2.2      | Algorithm of Gilmore and Gomory                   | 20 |
| 2.3      | Longest Path Method                               | 23 |
| 2.4      | Branch-and-Bound Algorithms                       | 25 |
| 2.5      | Periodicity and Dominance of Solutions            | 29 |
| 2.6      | Knapsack Problems with Upper Bounds               | 32 |
| 2.7      | Sets of Potential Allocation Points               | 34 |
| 2.8      | Exercises   | 38 |
| 2.9      | Solutions   | 40 |
|          | References  | 45 |
| <b>3</b> | <b>One-Dimensional Bin Packing</b>                | 47 |
| 3.1      | Problem Statement and Modeling                    | 47 |
| 3.2      | Lower Bounds                                      | 49 |
| 3.2.1    | Natural Bounds                                    | 49 |
| 3.2.2    | Combinatorial Bounds                              | 52 |
| 3.2.3    | Dual Feasible Functions                           | 54 |
| 3.2.4    | LP-Based Lower Bound                              | 58 |

|          |  |            |
|----------|--|------------|
| 3.3      | Reduction Methods and Equivalence of Instances .....                               | 58         |
| 3.4      | Heuristics .....   | 60         |
| 3.5      | Performance Results for 1BPP Heuristics.....                                       | 62         |
| 3.5.1    | Worst-Case Performance .....   | 63         |
| 3.5.2    | Average-Case Performance .....   | 67         |
| 3.6      | Exact Approaches and Extensions.....   | 67         |
| 3.7      | Exercises .....  | 68         |
| 3.8      | Solutions .....  | 69         |
|          | References .....   | 71         |
| <b>4</b> | <b>One-Dimensional Cutting Stock .....</b>   | <b>73</b>  |
| 4.1      | Problem Statement and Notations .....  | 74         |
| 4.2      | The Gilmore/Gomory Model .....   | 75         |
| 4.3      | Solving the Continuous Relaxation .....  | 78         |
| 4.3.1    | The Revised Simplex Method.....  | 78         |
| 4.3.2    | The Farley Bound.....  | 82         |
| 4.3.3    | Column Generation .....  | 83         |
| 4.4      | Getting Integer Solutions, Heuristics .....  | 88         |
| 4.5      | A Branch-and-Bound Approach .....  | 90         |
| 4.6      | Lower Bounds .....   | 92         |
| 4.7      | Equivalence of Instances .....   | 94         |
| 4.8      | Alternative Models .....   | 96         |
| 4.8.1    | A Nonlinear Model .....  | 96         |
| 4.8.2    | Kantorovich-Type Models .....  | 97         |
| 4.8.3    | The One-Cut Model .....  | 100        |
| 4.8.4    | The Arcflow Model .....  | 101        |
| 4.9      | Neighboring Problems .....   | 103        |
| 4.10     | The Cutting Stock Problem with Multiple Stock Lengths .....                        | 105        |
| 4.11     | Integer Round Up and Modified Integer Round Up Property .....                      | 107        |
| 4.11.1   | Definitions .....  | 107        |
| 4.11.2   | Transformations .....  | 109        |
| 4.11.3   | Subproblems Possessing the MIRUP .....   | 109        |
| 4.11.4   | Further Relaxations .....  | 113        |
| 4.11.5   | MIRUP and Higher-Dimensional Cutting Stock<br>Problems .....                       | 115        |
| 4.11.6   | Resume .....   | 115        |
| 4.12     | Exercises .....  | 116        |
| 4.13     | Solutions .....  | 117        |
|          | References .....   | 121        |
| <b>5</b> | <b>Orthogonal Packing Feasibility, Two-Dimensional Knapsack<br/>Problems .....</b> | <b>123</b> |
| 5.1      | Nonlinear Basic Models .....   | 124        |
| 5.2      | Integer Linear Programming Models .....  | 125        |
| 5.2.1    | The Beasley-Type Model.....  | 125        |
| 5.2.2    | The Padberg-Type Model .....   | 127        |

|          |   |            |
|----------|---|------------|
| 5.3      | Necessary Conditions for Feasibility .....          | 129        |
| 5.3.1    | Natural Criteria for Feasibility .....              | 129        |
| 5.3.2    | The Bar Relaxation .....                            | 130        |
| 5.3.3    | The Contiguous Relaxation .....                     | 134        |
| 5.4      | The Interval-Graph Approach .....                   | 138        |
| 5.4.1    | The Interval-Graph Model and a Simplification ..... | 138        |
| 5.4.2    | The Algorithm of Fekete and Schepers .....          | 140        |
| 5.5      | Sufficient Conditions for Feasibility .....         | 143        |
| 5.5.1    | Rectangular Pieces .....                            | 143        |
| 5.5.2    | Quadratic Pieces .....                              | 144        |
| 5.6      | A Constructive Branch-and-Bound Algorithm .....     | 145        |
| 5.6.1    | The Contour Concept .....                           | 145        |
| 5.6.2    | A Branch-and-Bound Algorithm .....                  | 147        |
| 5.6.3    | Upper Bounds .....                                  | 148        |
| 5.6.4    | Equivalence and Dominance .....                     | 151        |
| 5.7      | Further Approaches for the OPP .....                | 152        |
| 5.8      | Exercises .....                                     | 153        |
| 5.9      | Solutions .....                                     | 154        |
|          | References .....                                    | 155        |
| <b>6</b> | <b>Optimal Guillotine Cutting .....</b>             | <b>157</b> |
| 6.1      | Problem Statements .....                            | 157        |
| 6.2      | General Guillotine Cutting .....                    | 158        |
| 6.3      | Two-Stage Guillotine Cutting .....                  | 164        |
| 6.4      | Three-Stage Guillotine Cutting .....                | 165        |
| 6.5      | Further Pattern Types .....                         | 170        |
| 6.6      | Bounded Guillotine Cutting .....                    | 171        |
| 6.6.1    | The Algorithm of Wang .....                         | 171        |
| 6.6.2    | ILP Models for Bounded 2-Stage Cutting .....        | 172        |
| 6.7      | Guillotine Cutting Stock Problems .....             | 175        |
| 6.7.1    | Column Generation Approach .....                    | 175        |
| 6.7.2    | One-Cut ILP Model of the 2- and 3-Stage 2CSP .....  | 176        |
| 6.7.3    | Extensions .....                                    | 177        |
| 6.8      | Three-Dimensional Guillotine Cutting .....          | 177        |
| 6.9      | Exercises .....                                     | 179        |
| 6.10     | Solutions .....                                     | 179        |
|          | References .....                                    | 180        |
| <b>7</b> | <b>Packing Rectangles into a Strip .....</b>        | <b>183</b> |
| 7.1      | Integer Linear Programming Models .....             | 184        |
| 7.1.1    | A Beasley-Type Model .....                          | 184        |
| 7.1.2    | A Linear Mixed-Integer Model .....                  | 185        |
| 7.2      | Lower Bounds .....                                  | 186        |
| 7.2.1    | Natural Lower Bounds .....                          | 186        |
| 7.2.2    | Combinatorial Bounds .....                          | 189        |
| 7.2.3    | LP Based Bounds .....                               | 190        |

|          |  |            |
|----------|--|------------|
| 7.3      | Heuristics for the Strip Packing Problem .....           | 195        |
| 7.3.1    | Heuristics for the Offline Strip Packing Problem .....   | 195        |
| 7.3.2    | Performance Results .....                                | 198        |
| 7.3.3    | Shelf Algorithms for Online Problems .....               | 203        |
| 7.4      | Local Search and Metaheuristics .....                    | 207        |
| 7.4.1    | Local Search .....                                       | 207        |
| 7.4.2    | Metaheuristics .....                                     | 208        |
| 7.5      | A Branch-and-Bound Algorithm .....                       | 211        |
| 7.6      | Guillotine Strip Packing .....                           | 214        |
| 7.6.1    | Quality of Guillotine Patterns .....                     | 215        |
| 7.6.2    | ILP Models for $k$ -Stage Guillotine Strip Packing ..... | 218        |
| 7.7      | Exercises .....  | 221        |
| 7.8      | Solutions .....  | 222        |
|          | References .....   | 225        |
| <b>8</b> | <b>Two-Dimensional Bin Packing .....</b>                 | <b>227</b> |
| 8.1      | Problem Statement and Modeling .....                     | 227        |
| 8.1.1    | Non-Guillotine Patterns .....                            | 229        |
| 8.1.2    | Two-Stage Guillotine Patterns .....                      | 230        |
| 8.1.3    | Three-Stage Guillotine Patterns .....                    | 232        |
| 8.2      | Basic Results .....                                      | 234        |
| 8.3      | Lower Bounds .....                                       | 237        |
| 8.4      | Solution Approaches .....                                | 240        |
| 8.5      | Exercises .....  | 241        |
| 8.6      | Solutions .....  | 242        |
|          | References .....   | 243        |
| <b>9</b> | <b>Quality Restrictions .....</b>                        | <b>245</b> |
| 9.1      | Cutting of Variable Lengths .....                        | 245        |
| 9.1.1    | Problem Formulation .....                                | 246        |
| 9.1.2    | Modeling .....   | 247        |
| 9.1.3    | Optimal Value Function .....                             | 248        |
| 9.1.4    | Upper Bounds and Solution Strategy .....                 | 250        |
| 9.1.5    | Allocating a Single Piece .....                          | 254        |
| 9.1.6    | Solution Approaches .....                                | 258        |
| 9.1.7    | Example .....  | 260        |
| 9.2      | Defective or Forbidden Regions .....                     | 264        |
| 9.2.1    | Problem Formulation .....                                | 264        |
| 9.2.2    | Basic Recursion .....                                    | 266        |
| 9.2.3    | Improvements .....                                       | 267        |
| 9.2.4    | Examples and Extensions .....                            | 270        |
| 9.3      | Exercises .....  | 272        |
| 9.4      | Solutions .....  | 274        |
|          | References .....   | 277        |

|   |     |
|---|-----|
| <b>10 Pallet Loading .....</b>                                    | 279 |
| 10.1 The Standard Pallet Loading Problem .....                    | 279 |
| 10.1.1 Problem Statement .....                                    | 280 |
| 10.1.2 Equivalence of PLP Instances .....                         | 280 |
| 10.1.3 Upper Bounds .....   | 282 |
| 10.2 The G4 Heuristic .....                                       | 284 |
| 10.2.1 Notations, Block-Heuristics .....                          | 284 |
| 10.2.2 The G4-Structure .....                                     | 286 |
| 10.2.3 The Basic Recursion .....                                  | 287 |
| 10.2.4 Improvements .....   | 289 |
| 10.3 The Guillotine Pallet Loading Problem .....                  | 291 |
| 10.3.1 Basic Model and Algorithm .....                            | 291 |
| 10.3.2 Definition of Subproblems .....                            | 294 |
| 10.3.3 Properties of $E_0$ and $E_1$ .....                        | 296 |
| 10.3.4 Algorithm for Subproblem 1 .....                           | 299 |
| 10.4 The Pallet Loading Problem with Several Item Types .....     | 302 |
| 10.5 The Multi Pallet Loading Problem .....                       | 305 |
| 10.5.1 Problem Statement .....                                    | 305 |
| 10.5.2 Solution Strategy .....                                    | 306 |
| 10.5.3 The MPLP Algorithm .....                                   | 307 |
| 10.5.4 The Distributing Procedure .....                           | 308 |
| 10.5.5 The Loading Procedure .....                                | 309 |
| 10.5.6 Alternative Loading Procedure .....                        | 311 |
| 10.5.7 Examples .....   | 311 |
| 10.6 Exercises .....  | 313 |
| 10.7 Solutions .....  | 313 |
| References .....  | 315 |
| <b>11 Container Loading .....</b>                                 | 317 |
| 11.1 Problem Statements .....                                     | 317 |
| 11.2 The Container Loading Problem .....                          | 318 |
| 11.2.1 Integer Linear Programming Model .....                     | 318 |
| 11.2.2 The Basic Algorithm .....                                  | 321 |
| 11.2.3 The Contour Concept .....                                  | 323 |
| 11.2.4 Further Heuristics .....                                   | 327 |
| 11.3 Multi Container Loading and Three-Dimensional Bin Packing .. | 328 |
| 11.4 Three-Dimensional Strip Packing .....                        | 329 |
| 11.5 LP Bounds .....  | 330 |
| 11.5.1 The Bar Relaxation of the CLP .....                        | 330 |
| 11.5.2 The Bar Relaxation of the MCLP .....                       | 336 |
| 11.5.3 A Layer Relaxation for the Container Loading Problem ..... | 338 |
| 11.6 Exercises .....  | 340 |
| 11.7 Solutions .....  | 341 |
| References .....  | 343 |

|              |  |     |
|--------------|--|-----|
| <b>12</b>    | <b>Packing of Polygonal Pieces</b>                   | 345 |
| 12.1         | Problem Statement and Modeling                       | 345 |
| 12.1.1       | Strip Packing  | 345 |
| 12.1.2       | Related Problems                                     | 347 |
| 12.2         | Construction of $\Phi$ -Functions                    | 348 |
| 12.2.1       | Convex Polygons                                      | 348 |
| 12.2.2       | Non-convex Polygons                                  | 355 |
| 12.2.3       | Rotation of Objects                                  | 358 |
| 12.2.4       | Normalized $\Phi$ -Functions, $\rho$ -Dense Patterns | 360 |
| 12.3         | Computation of All Allocation Points                 | 362 |
| 12.3.1       | A Model with Binary Variables                        | 362 |
| 12.3.2       | The Basic Algorithm                                  | 363 |
| 12.3.3       | The Rectangular Case                                 | 364 |
| 12.3.4       | Example  | 365 |
| 12.4         | Algorithms for Strip Packing Problems                | 369 |
| 12.4.1       | Sequential Approaches                                | 370 |
| 12.4.2       | Improvement Methods                                  | 371 |
| 12.4.3       | Metaheuristics                                       | 372 |
| 12.4.4       | Further Aspects                                      | 372 |
| 12.5         | Exercises  | 373 |
| 12.6         | Solutions  | 373 |
|              | References   | 381 |
| <b>13</b>    | <b>Circle and Sphere Packing</b>                     | 385 |
| 13.1         | Problem Statements                                   | 385 |
| 13.2         | Packing Circles into a Circle                        | 386 |
| 13.2.1       | Modeling   | 386 |
| 13.2.2       | Computing a Local Minimum Solution                   | 389 |
| 13.2.3       | Overcoming a Local Minimum                           | 390 |
| 13.2.4       | Further Heuristics                                   | 392 |
| 13.3         | Strip Packing  | 392 |
| 13.4         | Packing Identical Circles into a Circle              | 394 |
| 13.5         | Packing Circles into a Square                        | 395 |
| 13.5.1       | Modeling   | 396 |
| 13.5.2       | Lower Bounds   | 396 |
| 13.5.3       | Upper Bounds   | 397 |
| 13.5.4       | Optimal Patterns                                     | 398 |
| 13.6         | Packing Identical Circles into a Rectangle           | 398 |
| 13.7         | Sphere Packing                                       | 399 |
| 13.8         | Exercises  | 400 |
| 13.9         | Solutions  | 401 |
|              | References   | 404 |
| <b>Index</b> |  | 407 |

# Acronyms and Notations

## Acronyms

|            |  |
|------------|--|
| BF, BFD(H) | Best Fit, Best Fit Decreasing (Height)   |
| BL         | Bottom Left                              |
| BPP        | Bin Packing Problem                      |
| b&b        | Branch-and-bound                         |
| CLP        | Container Loading Problem                |
| CSP        | Cutting Stock Problem                    |
| C&P        | Cutting and Packing                      |
| DP         | Dynamic Programming                      |
| FF, FFD(H) | First Fit, First Fit Decreasing (Height) |
| FFS        | First Fit Shelf                          |
| G4         | Generalized 4-Block PLP heuristic        |
| GA         | Genetic Algorithm                        |
| HS         | Harmonic Shelf                           |
| ILP        | Integer Linear Programming               |
| IP         | Integer Property                         |
| IRUP       | Integer Round-Up Property                |
| KP         | Knapsack Problem                         |
| LP         | Linear Programming                       |
| LS         | Local Search                             |
| MILP       | Mixed-Integer Linear Programming         |
| MIRUP      | Modified Integer Round-Up Property       |
| M4         | G4 heuristic with at most 4 item types   |
| M8         | G4 heuristic with at most 8 item types   |
| NF, NFD(H) | Next Fit, Next Fit Decreasing (Height)   |
| NFS        | Next Fit Shelf                           |
| OPP        | Orthogonal Packing Feasibility Problem   |
| PLP        | Pallet Loading Problem                   |
| SA         | Simulated Annealing                      |

|            |  |
|------------|--|
| SPP        | Strip Packing Problem                    |
| s.t.       | Subject to                               |
| SVC        | Sequential Value Correction              |
| TS         | Tabu Search                              |
| WF, WFD(H) | Worst Fit, Worst Fit Decreasing (Height) |

## Notations

|  |  |
|--|--|
| $\mathbb{B} = \{0, 1\}$                          | Set of binary numbers                                |
| $\mathbb{N} = \{1, 2, 3, \dots\}$                | Set of natural numbers                               |
| $\mathbb{Z}_+ = \{0, 1, 2, \dots\}$              | Set of nonnegative integers                          |
| $\mathbb{Z} = \{0, \pm 1, \pm 2, \dots\}$        | Set of integers                                      |
| $\mathbb{R}^m$                                   | $m$ -dimensional Euclidean space, $m \in \mathbb{N}$ |
| $\mathbb{R}_> = \{x \in \mathbb{R} : x > 0\}$    |  |
| $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$ |  |
| $ I  = \text{card}(I)$                           | Cardinality of a finite set $I$                      |
| $[x] = \max\{y \in \mathbb{Z} : y \leq x\}$      | Largest integer not larger than $x$ (rounding down)  |
| $[x] = \min\{y \in \mathbb{Z} : y \geq x\}$      | Smallest integer not smaller than $x$ (rounding up)  |
| $\text{conv}(S)$                                 | Convex hull of set $S$                               |
| $\text{cl}(S)$                                   | Closure of set $S$                                   |
| $\text{int}(S)$                                  | Interior of set $S$                                  |
| $I_n$  | Identity matrix of dimension $n \times n$            |
| $e^i = (0, \dots, 0, 1, 0, \dots, 0)^\top$       | $i$ -th unit vector of appropriate dimension         |
| $n! = 1 \cdot 2 \cdots n$                        | Factorial  |
| $(x)_+ = \max\{0, x\}$                           |  |
| $\vee$   | Logical or   |
| $\wedge$   | Logical and  |
| $O(x)$   | Landau symbol; something is proportional to $x$      |

# List of Figures

|           |   |     |
|-----------|---|-----|
| Fig. 1.1  | Translation of triangle $O$ by vector $u$ .....   | 2   |
| Fig. 1.2  | Rotation and translation of object (triangle) $O$ .....   | 3   |
| Fig. 1.3  | Representation of a polygon .....   | 6   |
| Fig. 1.4  | Example 1.2 .....   | 7   |
| Fig. 2.1  | Algorithm of Gilmore and Gomory .....   | 21  |
| Fig. 2.2  | Algorithm of Gilmore and Gomory with index information .....  | 22  |
| Fig. 2.3  | Algorithm of Gilmore and Gomory: identification<br>of a solution .....  | 23  |
| Fig. 2.4  | Longest Path Method .....   | 24  |
| Fig. 2.5  | Partition into $\alpha_k + 1$ subproblems .....   | 26  |
| Fig. 2.6  | Branching tree: the entries within the ovals correspond<br>to $k/y$ . The upper bounds are given with respect to the<br>initial problem .....   | 28  |
| Fig. 2.7  | General Branch-and-Bound algorithm .....  | 28  |
| Fig. 2.8  | Alternative concept: partition into two subproblems<br>$(\tilde{y} = y - \alpha_k a_k, \alpha_{k+1} = \lfloor \tilde{y}/a_{k+1} \rfloor)$ ..... | 29  |
| Fig. 2.9  | Lexicographic Longest Path Method .....   | 31  |
| Fig. 2.10 | Modified Longest Path Method .....  | 34  |
| Fig. 2.11 | The Backward Dynamic Programming (BDP) algorithm .....  | 36  |
| Fig. 2.12 | The Forward Dynamic Programming (FDP) algorithm .....   | 36  |
| Fig. 2.13 | Algorithm to compute all solutions .....  | 41  |
| Fig. 3.1  | Dual feasible functions $y = u^{(k)}(x)$ for $k = 1, \dots, 4$ .....  | 55  |
| Fig. 3.2  | Dual feasible function $y = U^{(\varepsilon)}(x)$ .....   | 55  |
| Fig. 3.3  | Dual feasible functions $y = \phi^{(\varepsilon)}(x)$ .....   | 56  |
| Fig. 4.1  | General step of the Revised Simplex Method .....  | 80  |
| Fig. 4.2  | Column generation for solving problem (4.8) .....   | 85  |
| Fig. 4.3  | Branch-and-bound algorithm for the 1CSP .....   | 91  |
| Fig. 5.1  | Horizontal bar relaxation for rectangle packing .....   | 130 |
| Fig. 5.2  | A 3D container and the 1 <sup>st</sup> bar relaxation (along $W_1$ ) .....  | 135 |

|           |  |     |
|-----------|--|-----|
| Fig. 5.3  | The horizontal contiguous bar relaxation does not provide an OPP solution in general .....               | 136 |
| Fig. 5.4  | An exemplary packing pattern, the complement intersection graphs and their transitive orientations ..... | 139 |
| Fig. 5.5  | Example of a feasible graph system where $G_1$ is non-interval .....                                     | 140 |
| Fig. 5.6  | A ‘degenerate’ 2OPP example from [13] .....  | 140 |
| Fig. 5.7  | The branching schema: enumerating by edges .....   | 141 |
| Fig. 5.8  | Monotone packing according to the contour concept .....  | 147 |
| Fig. 5.9  | Branch-and-bound algorithm for the 2OPP .....  | 148 |
| Fig. 5.10 | Horizontal and vertical bar relaxation .....   | 150 |
| Fig. 6.1  | Guillotine and non-guillotine cutting .....  | 158 |
| Fig. 6.2  | Applying a vertical or horizontal guillotine cut .....   | 159 |
| Fig. 6.3  | Optimal patterns for $L \times W = 160 \times 105$ and $L \times W = 160 \times 100$ .....               | 161 |
| Fig. 6.4  | Exact and non-exact 2-stage guillotine cutting patterns with horizontal G-cuts in the first stage .....  | 164 |
| Fig. 6.5  | A 3-stage guillotine cutting pattern, exact case, vertical G-cuts in the first stage .....               | 165 |
| Fig. 6.6  | A 3-stage guillotine cutting pattern, non-exact case .....   | 166 |
| Fig. 6.7  | Optimal 3-stage patterns of Example 6.3 .....  | 169 |
| Fig. 6.8  | 1-group cutting pattern .....  | 170 |
| Fig. 6.9  | 2- and 3-block patterns .....  | 170 |
| Fig. 6.10 | A schema of the Algorithm of Wang .....  | 172 |
| Fig. 6.11 | To the one-cut model of 2- and 3-stage guillotine cutting .....  | 177 |
| Fig. 6.12 | Three-dimensional guillotine patterns .....  | 178 |
| Fig. 7.1  | Item $1 \times 2$ fits in a pattern for the contiguous problem of height 4, but not for the SPP .....    | 194 |
| Fig. 7.2  | BL-packing for piece sequences $\pi = (1, 2, 3, 4, 5, 6, 7)$ and $\pi = (7, 6, 5, 4, 3, 2, 1)$ .....     | 196 |
| Fig. 7.3  | $BL_0$ -patterns for sequences $\pi = (1, 2, 3, 4, 5, 6, 7)$ and $\pi = (7, 6, 5, 4, 3, 2, 1)$ .....     | 197 |
| Fig. 7.4  | Patterns obtained by the NFDH and the FFDH heuristic .....   | 198 |
| Fig. 7.5  | To the proof of Theorem 7.9 (a), pattern obtained for $k = 3$ .....                                      | 199 |
| Fig. 7.6  | To the proof of Theorem 7.9 (c), pattern obtained for $k = 4$ .....                                      | 200 |
| Fig. 7.7  | To the proof of Theorem 7.11 (b) .....   | 202 |
| Fig. 7.8  | NFS and FFS patterns .....   | 204 |
| Fig. 7.9  | Normalized monotone packing pattern used in the contour approach .....                                   | 212 |
| Fig. 7.10 | Branch-and-bound algorithm for the SPP .....   | 212 |
| Fig. 7.11 | 2-, 3-, and 5-stage guillotine cutting patterns .....  | 215 |
| Fig. 7.12 | Exact 2-stage, non-exact 2-stage, and general 3-stage G-patterns .....                                   | 219 |

|            |   |     |
|------------|---|-----|
| Fig. 7.13  | Example with $z^{b-hor} < z^{c-hor} < H^*$  | 225 |
| Fig. 9.1   | Finger-joining technology   | 246 |
| Fig. 9.2   | Optimal value function of Example 9.1   | 249 |
| Fig. 9.3   | Bounding function $ub_1$ of Example 9.2   | 252 |
| Fig. 9.4   | Bounding function $ub_2$ of Example 9.2   | 253 |
| Fig. 9.5   | Yield function $h$ of Example 9.3   | 254 |
| Fig. 9.6   | Possible situations which can occur during the update of $h$                                      | 256 |
| Fig. 9.7   | A sketch of a branch-and-bound algorithm  | 258 |
| Fig. 9.8   | A sketch of an FDP algorithm  | 259 |
| Fig. 9.9   | Yield functions $h_1, \dots, h_5$   | 261 |
| Fig. 9.10  | Description of defects by <i>rectangles</i>   | 265 |
| Fig. 9.11  | Exact 3-stage guillotine cutting  | 266 |
| Fig. 9.12  | Definition of set $Y_d$   | 269 |
| Fig. 9.13  | Cutting pattern of Example 9.4  | 271 |
| Fig. 9.14  | A more general pattern of a strip   | 272 |
| Fig. 9.15  | Cutting pattern of Example 9.5  | 273 |
| Fig. 9.16  | Optimal value functions of Exercise 9.3 (a)–(c)   | 274 |
| Fig. 9.17  | Optimal value functions of Exercise 9.4 (a)–(c)   | 275 |
| Fig. 9.18  | Bounding functions $ub_1$ and $ub_2$ of Exercise 9.5 (a)  | 276 |
| Fig. 9.19  | Bounding functions $ub_1$ and $ub_2$ of Exercise 9.5 (b)  | 277 |
| Fig. 10.1  | Optimal pattern of the PLP instances (40, 25, 7, 3) and (52, 33, 9, 4)                            | 281 |
| Fig. 10.2  | Optimal 3-block pattern for (13, 11, 3, 2) and optimal 4-block pattern for (17, 11, 3, 2)         | 285 |
| Fig. 10.3  | Optimal 5-block pattern for (44, 37, 7, 5) and optimal 7-block pattern for (37, 22, 7, 3)         | 285 |
| Fig. 10.4  | Optimal 9-block pattern for (68, 52, 10, 7) and optimal diagonal-block pattern for (27, 25, 7, 3) | 286 |
| Fig. 10.5  | Notations used in the G4-recursion  | 288 |
| Fig. 10.6  | Optimal patterns of the instances (57, 44, 12, 5) and (56, 52, 12, 5)                             | 289 |
| Fig. 10.7  | The G4 heuristic for the PLP  | 290 |
| Fig. 10.8  | Two types of $\ell$ -strips. (a) vertical $\ell$ -strip. (b) horizontal $\ell$ -strip             | 292 |
| Fig. 10.9  | Principal shape of a block- $\ell$ -pattern   | 292 |
| Fig. 10.10 | Partition of $L \times W$ in three subregions   | 296 |
| Fig. 10.11 | Modulo function $E(y) = (5 + 4y) \bmod 23$  | 297 |
| Fig. 10.12 | Optimal patterns of instance (40, 25, 7, 3) for the GPLP  | 301 |
| Fig. 10.13 | 4-block-dissections of a pallet   | 302 |

|            |  |     |
|------------|--|-----|
| Fig. 10.14 | Patterns of Example 10.4 regarding various additional restrictions. (a) $v = 992, 336$ . (b) $v = 990, 198$ , $t = 3$ . (c) $v = 983, 988$ , $t = 2$ . (d) $v = 989, 952$ , same type in II and IV. (e) $v = 982, 154$ , $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$ . (f) $v = 963, 382$ , $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$ , $u = (0, 0, 0, 0, 0, 11)^\top$ . (g) $v = 949, 136$ , $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$ , $u = (0, 0, 0, 8, 0, 11)^\top$ . (h) $v = 990, 198$ , with sequencing condition ..... | 304 |
| Fig. 10.15 | The MPLP algorithm .....   | 307 |
| Fig. 10.16 | Distributing Procedure for the MPLP algorithm .....  | 308 |
| Fig. 10.17 | Loading Procedure for the MPLP algorithm .....   | 310 |
| Fig. 10.18 | Layer-wise packing in Example 10.5 .....   | 312 |
| Fig. 10.19 | Patterns of Example 10.6 .....   | 313 |
| Fig. 10.20 | G4-structure of a 5-block pattern .....  | 314 |
| Fig. 10.21 | G4-structure of a 7-block pattern .....  | 314 |
| Fig. 11.1  | Six possible orientations of a box .....   | 319 |
| Fig. 11.2  | Relation $R_i <_R R_j$ .....   | 325 |
| Fig. 11.3  | Non-monotone and monotone contour of Example 11.1 .....  | 325 |
| Fig. 11.4  | Counterexample for an exclusive application of monotone contours .....   | 326 |
| Fig. 11.5  | Packing pattern of Example 11.3 .....  | 332 |
| Fig. 12.1  | Polygons considered in Example 12.1 .....  | 349 |
| Fig. 12.2  | Mutual positions of two polygons .....   | 350 |
| Fig. 12.3  | Placements related to Example 12.2 .....   | 351 |
| Fig. 12.4  | Hodographs related to Example 12.2 .....   | 351 |
| Fig. 12.5  | Packing regions $U_i$ , $i \in \{1, 2, 3\}$ , related to Example 12.3 .....  | 353 |
| Fig. 12.6  | Some patterns belonging to Example 12.3 .....  | 353 |
| Fig. 12.7  | Allocation regions $U_{12}(u_1)$ of Example 12.3 for $u_1 = (0, 0), (0.5, 1)$ , and $(1, 2)$ .....   | 354 |
| Fig. 12.8  | Patterns of Example 12.4 with $m = 4$ .....  | 354 |
| Fig. 12.9  | Rotation ranges in Example 12.5 .....  | 360 |
| Fig. 12.10 | Normalized $\Phi$ -function and its approximation .....  | 361 |
| Fig. 12.11 | Basic algorithm to obtain all allocation points .....  | 364 |
| Fig. 12.12 | Polygon $P$ and region $R$ .....   | 366 |
| Fig. 12.13 | Some possible placements .....   | 369 |
| Fig. 12.14 | Set of all allocation points .....   | 369 |
| Fig. 12.15 | Polygons of Exercise 12.1 .....  | 374 |
| Fig. 12.16 | No-fit polygons of Exercise 12.1 .....   | 374 |
| Fig. 12.17 | Examples of Exercise 12.4 .....  | 376 |
| Fig. 12.18 | Example-polygon of Exercise 12.5 .....   | 376 |
| Fig. 12.19 | Example-polygons of Exercise 12.9 .....  | 380 |
| Fig. 13.1  | Non-convexity of $G$ .....   | 388 |
| Fig. 13.2  | Non-convexity of feasible region .....   | 393 |

|           |   |     |
|-----------|---|-----|
| Fig. 13.3 | Feasible region and extreme points .....      | 393 |
| Fig. 13.4 | Optimal patterns for $n \leq 7$ .....         | 395 |
| Fig. 13.5 | Optimal patterns for $8 \leq n \leq 12$ ..... | 395 |
| Fig. 13.6 | Regular patterns .....                        | 399 |
| Fig. 13.7 | Notations to Exercise 13.4 .....              | 403 |

# List of Tables

|            |   |     |
|------------|---|-----|
| Table 2.1  | Algorithm of Gilmore and Gomory: optimal values $F(k, y)$ of Example 2.1 .....  | 22  |
| Table 2.2  | Longest Path Method: optimal values $f(y) = v(y)$ of Example 2.2 .....          | 25  |
| Table 2.3  | Potential allocation points of Example 2.4 .....                                | 38  |
| Table 2.4  | Algorithm of Gilmore and Gomory: optimal values $F(k, y)$ to Exercise 2.4 ..... | 42  |
| Table 2.5  | Longest Path Method applied to Exercise 2.4 .....                               | 42  |
| Table 2.6  | Lexicographic Longest Path Method applied to Exercise 2.13 .....                | 44  |
| Table 4.1  | Input data of Example 4.2 .....   | 81  |
| Table 6.1  | Input data of Example 6.1 .....   | 161 |
| Table 6.2  | Calculations according to recursion (6.3) .....                                 | 161 |
| Table 6.3  | Calculations according to recursion (6.4) .....                                 | 163 |
| Table 6.4  | Input data of Example 6.3 .....   | 168 |
| Table 6.5  | Recursions (6.8) and (6.10) with $\tilde{S} = S(\ell, L)$ .....                 | 169 |
| Table 6.6  | Recursions (6.8) and (6.10) with $\tilde{S} = S^{red}(\ell, L)$ .....           | 169 |
| Table 7.1  | Input data of Example 7.2 .....   | 196 |
| Table 7.2  | Input data of Example 7.4 .....   | 198 |
| Table 7.3  | Input data of Example 7.6 .....   | 204 |
| Table 9.1  | Yield functions $h_1, \dots, h_5$ .....   | 262 |
| Table 9.2  | Input data of the piece types of Example 9.4 .....                              | 270 |
| Table 9.3  | Input data of defects of Example 9.4 .....                                      | 270 |
| Table 9.4  | Description of defects of Example 9.4 using $\psi$ and $\omega$ .....           | 271 |
| Table 9.5  | Input data of piece types of Example 9.5 .....                                  | 272 |
| Table 9.6  | Input data of defects of Example 9.5 .....                                      | 272 |
| Table 10.1 | Comparison of computational times for the 206 instances of COVER II .....       | 291 |

|            |  |     |
|------------|--|-----|
| Table 10.2 | Comparison of variants in dependence on $L$ .....                    | 291 |
| Table 10.3 | Input data of Example 10.4 .....                                     | 303 |
| Table 10.4 | Input data of Example 10.5 .....                                     | 311 |
| Table 10.5 | Input data of Example 10.6 and patterns obtained .....               | 312 |
| Table 11.1 | Six possible orientations of a cuboid .....                          | 319 |
| Table 12.1 | Input data of non-convex polygon $P$ .....                           | 365 |
| Table 12.2 | Input data of non-convex regions $R$ .....                           | 366 |
| Table 12.3 | Functions $f_{ijl}$ for $j = 1, \dots, 4$ and $i \in \{1, 2\}$ ..... | 366 |
| Table 12.4 | Functions $f_{ijl}$ for $j \geq 5$ and $i \in \{1, 2\}$ .....        | 367 |
| Table 13.1 | Maximal minimum distance of $n$ points within a unit circle .....    | 395 |
| Table 13.2 | Maximal minimum distance of $n$ points within a unit square .....    | 399 |

# Chapter 1

## Modeling

Cutting and packing problems can be modeled in various ways which enables us to apply different solution approaches. Within this introductory chapter we introduce frequently used models by means of examples. Related solution approaches will be considered within the subsequent chapters.

One of the essential aspects for modeling cutting and packing problems consists in the way of representing the objects which have to be placed. Although only three-dimensional objects appear in reality, the terms of *one-dimensional cutting problem* or *two-dimensional packing problem* are typically used depending on the number of parameters required to describe the problem in a sufficiently clear manner and to formulate an appropriate mathematical model.

### 1.1 Set-Theoretical Models

For one-dimensional cutting and packing problems as well as for higher-dimensional problems with objects having a regular structure (e.g. rectangles, boxes), only a small number of input data are needed to describe an object (e.g. length, width, height). In contrast, in the textile and other industries where non-rectangular pieces have to be produced, such a short description is not possible, in general. For that reason, we consider a very general allocation problem at first.

The region  $B$  into which the objects have to be placed, is always assumed to be a closed and bounded subset of the Euclidean space  $\mathbb{R}^d$  where  $d \in \mathbb{N} = \{1, 2, 3, \dots\}$ . In general, we have  $d \in \{1, 2, 3\}$ . In this case, any object (or piece, item)  $O_i$ ,  $i \in I$ , to be placed is also represented by a non-empty compact set of  $\mathbb{R}^d$ . Here, and in the following,  $I$  denotes an appropriate index set of objects which have to be placed.

In order to model real objects we always assume that every object  $O_i$  fulfills the relation

$$\text{cl}(\text{int}(O_i)) = O_i.$$

As usual,  $\text{cl}(S)$  denotes the *closure* and  $\text{int}(S)$  the *interior* of set  $S$ . In general, we assume that the object  $O_i$  is given in *normalized position* which is defined by

$$\min\{x_k : x = (x_1, \dots, x_d)^\top \in O_i\} = 0 \quad \text{for all } k = 1, \dots, d.$$

We assign a *reference point* to each object which has to be placed. For an object in normalized position we identify its reference point with the origin of  $\mathbb{R}^d$ , in general.

Let us consider a cutting or packing problem where rotation of objects is not permitted and the objective is to maximize the material utilization. In that case, we have to determine a subset  $I^* \subseteq I$  of objects as well as *translation vectors*  $u_i \in \mathbb{R}^d$ ,  $i \in I^*$ , such that, for all  $i \in I^*$ , the translated objects  $O_i$  (translated by vectors  $u_i$ ) satisfy all allocation constraints. Let

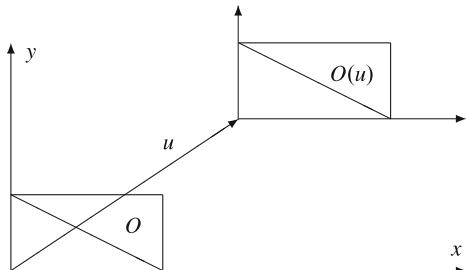
$$O_i(u_i) := u_i + O_i := \{v \in \mathbb{R}^d : v = u_i + u, u \in O_i\}$$

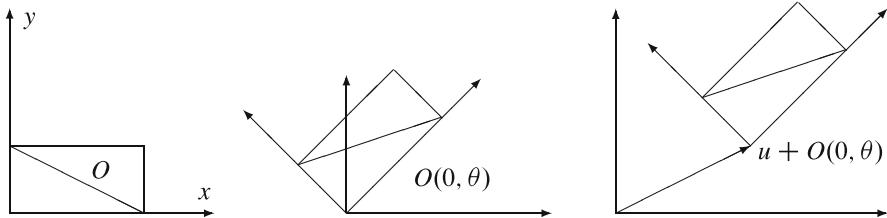
denote the *translation* of object  $O_i$  by vector  $u_i$ . Hence, vector  $u_i$  also indicates the position of the translated reference point of  $O_i$ . We call point  $u_i$  the *allocation point* of the object. The translation of an object is depicted in Fig. 1.1.

If *rotation* of objects is permitted, additional  $d'$ -dimensional vectors  $\theta_i \in \Theta_i \subseteq \mathbb{R}^{d'}$  of rotation angles are searched in the allocation problem ( $d' = 1$  if  $d = 2$ ;  $d' = 3$  if  $d = 3$ ).

*Remark* In many cases, the set of possible rotation angles  $\theta_i$  is strongly restricted or even a finite one. For instance, when cutting or packing rectangular pieces mostly only rotation by  $90^\circ$  is allowed. In the case of orthogonal packing of boxes at most six different axis-parallel orientations are possible. Moreover, in the textile industry the texture of the material has to be regarded, in general, and therefore can lead to restricted ranges of rotation angles.

**Fig. 1.1** Translation of triangle  $O$  by vector  $u$





**Fig. 1.2** Rotation and translation of object (triangle)  $O$

In the two-dimensional case, the rotation of an object by the angle  $\theta$  can be described by means of the *rotation matrix*

$$D_\theta := \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}.$$

The point set which results through rotation of object  $O_i(u_i)$  by angle  $\theta_i$  and center  $u_i$  of rotation, is the following:

$$D(O_i(u_i), \theta_i) := \{v \in \mathbb{R}^2 : v = u_i + D_{\theta_i}w, w \in O_i(0)\}.$$

The rotation and translation of an object is depicted in Fig. 1.2

The following allocation constraints have to be fulfilled in the majority of cutting and packing problems with only very few exceptions:

- **Containment condition**

The condition that an allocated object  $O_i$ , i.e.  $i \in I^*$ , which is translated by vector  $u_i$  and rotated by angle  $\theta_i$ , is completely contained within the region  $B$ , can easily be described in a set-theoretical manner as follows:

$$D(O_i(u_i), \theta_i) \subseteq B, \quad \forall i \in I^*.$$

- **Non-overlapping condition**

In the placement problems considered in this book, the allocation of the objects  $O_i$  and  $O_j$  has to be done such that no overlap appears:

$$\text{int}(D(O_i(u_i), \theta_i)) \cap D(O_j(u_j), \theta_j) = \emptyset, \quad \forall i \neq j, i, j \in I^*.$$

*Remark* In some applications a minimum distance between two allocated objects has to be guaranteed, e.g., a saw kerf when cutting wood. The consideration of a saw kerf is addressed in Exercise 1.1. In general, we do not regard minimum distances.

Let  $c_i$  denote the *value* (or profit coefficient) of object  $O_i$  ( $c_i \in \mathbb{R}_+$ ). In the case of maximizing the material utilization we can take, for instance, the length, the area, or the volume of object  $O_i$  as  $c_i$ . Herewith, we obtain the following set-theoretical model which is frequently called *generalized 0/1 Knapsack Problem*.

### Set-theoretical model of maximizing the material utilization

$$\max \sum_{i \in I^*} c_i \quad \text{s.t.} \quad (1.1)$$

$$D(O_i(u_i), \theta_i) \subseteq B, \quad \forall i \in I^*, \quad (1.2)$$

$$\text{int}(D(O_i(u_i), \theta_i)) \cap D(O_j(u_j), \theta_j) = \emptyset, \quad \forall i \neq j, i, j \in I^*, \quad (1.3)$$

$$I^* \subseteq I. \quad (1.4)$$

Obviously, we are looking for a subset  $I^*$  of the available objects due to restriction (1.4). According to conditions (1.2) and (1.3) all objects of  $I^*$  can be placed without overlap simultaneously within region  $B$  provided that they are rotated by angles  $\theta_i$  and positioned at allocated points  $u_i, i \in I^*$ . Due to target function (1.1), the material utilization becomes maximal if the objects of  $I^*$  are chosen.

In order to formulate a set-theoretical model for the second type of cutting and packing problems, the minimization of material usage, we suppose that several regions  $B_j$  with values  $c_j, j \in J$ , and objects  $O_i, i \in I$ , all to be produced, are given. Now the task is to assign each object to one of the regions such that all placement conditions are met. That means that we have to identify a subset  $J^*$  of  $J$  and index sets  $I_j \subset I$  for  $j \in J^*$  with  $\cup_{j \in J^*} I_j = I$ , and to find translation vectors  $u_i$  and rotation angles  $\theta_i$  for all  $i \in I$ .

### Set-theoretical model of minimizing the material usage

$$\min \sum_{j \in J^*} c_j \quad \text{s.t.} \quad (1.5)$$

$$D(O_i(u_i), \theta_i) \subseteq B_j, \quad \forall i \in I_j \forall j \in J^*, \quad (1.6)$$

$$\text{int}(D(O_i(u_i), \theta_i)) \cap D(O_k(u_k), \theta_k) = \emptyset, \quad \forall i \neq k, i, k \in I_j, \quad \forall j \in J^*, \quad (1.7)$$

$$\bigcup_{j \in J^*} I_j = I, \quad (1.8)$$

$$I_j \subseteq I \quad \forall j \in J^*, \quad J^* \subseteq J. \quad (1.9)$$

Because of restriction (1.9) an index set  $J^*$  of available regions is searched as well as an assignment of all objects to the selected regions [conditions (1.6) and (1.8)]. Restriction (1.7) guarantees the feasibility of allocation.

Both models, (1.1)–(1.4) and (1.5)–(1.9), are formulated in a very general manner and therefore, are less applicable for solving real tasks. In general, a more detailed description of the real problem is needed. The representation of the region(s)  $B$  ( $B_j$ ) and objects  $O_i$  has to provide sufficient structural information to obtain more suitable formulations of mathematical models and to enable the computation of feasible or even optimal solutions.

**Definition 1.1** A feasible placement of objects within a region  $B$  is called *packing* or *cutting pattern*, or *arrangement*.

A pattern has to indicate which pieces are placed, as well as their allocation points, and their angles of rotation, if necessary. Nevertheless, for certain cutting and packing problems we will use some modifications which imply fewer, but sufficiently many information.

## 1.2 Representation of Objects

In order to describe objects and regions, in general, we apply the following principles. We define

- non-connected objects as union of connected objects;
- non-convex objects as union of convex objects (if possible);
- convex polygonal objects as intersection of a finite number of half-planes, i.e., as solution set of a system of inequalities;
- regular objects as a tuple of characteristic parameters.

As usual, we characterize a (connected) one-dimensional object by its *length*, a rectangle by its *length* and *width* (or *width* and *height*), a box (orthogonal parallelepiped) by its *length*, *width* and *height*, a circle and a ball (sphere) by its radius, etc.

For a convex object  $O$  we assume a representation of the form

$$O = \{x \in \mathbb{R}^d : g_i(x) \leq 0, i \in I(O)\}$$

where  $I(O)$  denotes a finite index set and  $g_i$  is a convex function for all  $i \in I(O)$ .

If  $O$  is a bounded polyhedral set, i.e., a polytope (if  $d = 3$ ) or a (convex) polygon (if  $d = 2$ ), then a representation using its extreme points (corners) is possible due to Minkowski's Theorem (see e.g., [4], p. 96). Let  $v_j \in O, j \in J(O)$ , denote the extreme points of  $O$  where  $J(O)$  is an appropriate index set which obviously depends on  $O$ . Then object  $O$  can also be represented in the form

$$O = \text{conv}\{v_j : j \in J(O)\}$$

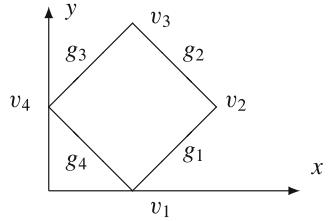
where

$$\text{conv}\{v_j : j \in J(O)\} := \{x \in \mathbb{R}^d : x = \sum_{j \in J(O)} \lambda_j v_j, \sum_{j \in J(O)} \lambda_j = 1, \lambda_j \geq 0, j \in J(O)\}$$

denotes the *convex hull* of the (finite) point set  $\{v_j : j \in J(O)\}$ .

*Example 1.1* In the Euclidean space  $\mathbb{R}^2$  the relationship between the two representations of a polygon is obvious. Without loss of generality, let  $v_i, i \in I := \{1, \dots, n\}$ ,

**Fig. 1.3** Representation of a polygon



denote the extreme points of the (convex) polygon  $O$  numbered counterclockwise. Then the corresponding half-planes  $\{x \in \mathbb{R}^2 : g_i(x) \leq 0\}$  are determined by  $g_i(x) := a_i^\top(x - v_i)$  where the normal vector  $a_i \in \mathbb{R}^2$  of  $g_i$  is given by  $a_i^\top(v_{i+1} - v_i) = 0$  and  $a_i^\top(v_k - v_i) \leq 0$  for all  $k \in I$  ( $i = 1, \dots, n$ ,  $v_{n+1} := v_1$ ). Additionally, it is mostly demanded that the length (norm) of  $a_i$  is equal to one, i.e., the Hesse normal form is used. Contrariwise, given the half-planes with a successive numbering, we obtain an extreme point  $v_i$  as the intersection point of two consecutive bounding lines, i.e. of  $g_{i-1}(x) = 0$  and  $g_i(x) = 0$  ( $g_0 := g_n$ ). Figure 1.3 illustrates this relationship. ■

A further opportunity to represent a convex object is given by

$$O = \{x \in \mathbb{R}^d : f(x) \leq 0\} \quad \text{with} \quad f(x) := \max\{g_i(x) : i \in I(O)\}$$

where the  $g_i$  are assumed to be convex functions. This form enables us in a simple way to represent an object which is the union of (finitely many) convex sets. Let  $O_k$ ,  $k \in K$ , denote convex sets and let

$$O := \bigcup_{k \in K} O_k \quad \text{with} \quad O_k = \{x \in \mathbb{R}^d : f_k(x) \leq 0\}, \quad k \in K.$$

Then we have

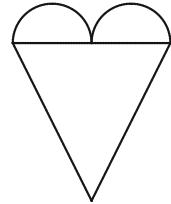
$$O = \{x \in \mathbb{R}^d : f(x) \leq 0\} \quad \text{with} \quad f(x) := \min\{f_k(x) : k \in K\}.$$

If object  $O_k$  ( $k \in K$ ) is defined by functions  $g_{ik}$ ,  $i \in I(O_k)$  or in the form  $f_k(x) := \max\{g_{ik}(x) : i \in I(O_k)\}$ , respectively, then we obtain the following representation of the, in general, non-convex object  $O$ :

$$O = \{x \in \mathbb{R}^d : \min_{k \in K} \max_{i \in I(O_k)} g_{ik}(x) \leq 0\}.$$

*Example 1.2* Consider the three convex regions,

$$\begin{aligned} O_1 &:= \{(x, y) \in \mathbb{R}^2 : f_1(x, y) := \max\{y, 2x - y - 4, -2x - y - 4\} \leq 0\}, \\ O_2 &:= \{(x, y) \in \mathbb{R}^2 : f_2(x, y) := \max\{-y, (x - 1)^2 + y^2 - 1\} \leq 0\}, \\ O_3 &:= \{(x, y) \in \mathbb{R}^2 : f_3(x, y) := \max\{-y, (x + 1)^2 + y^2 - 1\} \leq 0\}, \end{aligned}$$

**Fig. 1.4** Example 1.2

and the composed object  $O := O_1 \cup O_2 \cup O_3$  (Fig. 1.4). It is easy to see that  $O$  can also be described in the form

$$O := \{(x, y) \in \mathbb{R}^2 : f(x, y) := \min\{f_1(x, y), f_2(x, y), f_3(x, y)\} \leq 0\}. \quad \blacksquare$$

It is to notice that the representation of objects as shown above can lead, in general, to numerical difficulties which can be caused by the non-differentiability of the min- and max-function. Moreover, the verification of non-overlapping of two placed objects can become difficult.

### 1.3 Representation of Patterns

A (cutting or packing) pattern, according to Definition 1.1, is an arrangement of objects within a certain region. In order to give a complete description of it, all of its characteristic parameters must be contained. In general, the representation of an arrangement has to imply which objects are placed, as well as their particular allocation points and rotation angles.

In various cutting and packing problems which are considered in the following, such a complete description of a pattern is not needed. Instead, a considerably smaller number of characteristic data is sufficient. This can happen, in particular, in cutting and packing problems with homogeneous material, for example when cutting chip-boards, glass or steel sheets, or when packing pallets or containers. In contrast, if non-homogeneous material is available, such a complete information has to be provided by a pattern, in general.

For example, let us consider the (one-dimensional) 0/1 *knapsack problem* (Chap. 2): find a subset  $I^* \subseteq I$  of  $m$  available objects, represented by  $i \in I = \{1, \dots, m\}$ , such that a given knapsack can be filled best possible. In order to project the knapsack problem onto a single dimension we characterize each object  $i \in I$  by its volume  $v_i$  and the knapsack by its volume  $V$ . This enables us to represent a feasible arrangement (pattern) by a *nonnegative  $m$ -dimensional binary vector*  $a = (a_1, \dots, a_m)^\top \in \mathbb{B}^m$  where  $a_i = 1$  indicates that piece  $i$  is chosen, i.e.  $i \in I^*$ . Vector  $a \in \mathbb{B}^m$  represents a feasible pattern if and only if

$$\sum_{i \in I} v_i a_i \leq V. \quad (1.10)$$

Because of the reduced information contained within such a vector  $a \in \mathbb{B}^m$ , the reconstruction of a complete arrangement is not longer possible since different arrangements with the same objects lead to the same vector  $a$ . It is obvious, that a concrete sequence of placing the objects, and therefore the respective allocation points, are less important since every permuted sequence of all pieces  $i \in I^*$  yields a feasible arrangement.

Nevertheless, this loss of information leads to an essential simplification with respect to the mathematical modeling and solution methods. In general, a vector  $a \in \mathbb{B}^m$  which fulfills condition (1.10) represents not only a finite number of arrangements determined by different sequences of the placed objects, but actually an infinite one if  $\sum_{i \in I} v_i a_i < V$ . In this case, the allocation points of the objects can vary within a total range of length  $V - \sum_{i \in I} v_i a_i$ .

In many cases, the number of feasible patterns to be regarded in a corresponding mathematical model can be restricted to a finite one by means of normalized patterns [3]. For instance, a feasible pattern of the 0/1 knapsack problem represented by index set  $I^* = \{i_1, \dots, i_{|I^*|}\}$ , is called *normalized* if the allocation points  $x_{i_j}$  are determined by  $x_{i_1} = 0$  and  $x_{i_j} = \sum_{k=1}^{j-1} v_{i_k}$  for  $j = 2, \dots, |I^*|$ . That means that the objects are placed directly side by side. Potential unused space is located at the end. Frequently, the notation *left-justified* is used in the same sense.

Restricting to normalized patterns and applying the concept of potential sets of allocation points (see Sect. 2.7) can possibly lead to a further reduction of the number of feasible patterns, and hence of variables, in various models of cutting and packing problems.

In case of modeling more complex problems where, besides the containment and non-overlapping constraints, further restriction have to be regarded, one has to take care whether a reduced representation of arrangements is appropriate or not.

## 1.4 Approximation and Internal Description of Objects

Objects with curvilinear boundary, which occur for instance in the textile industry where a designer creates the objects, can hardly be handled by computers, in particular when arrangements should be computed quickly. At least, a description of the curvilinear boundary by functions is mostly helpful.

In order to approximate arbitrary curves, (*quadratic* or *cubic*) *splines* (special composed polynomial functions) or even *step functions* (piecewise constant) are applied. We refer to [8] for details. A representation of objects in a pixel-wise manner depends on the scaling and, therefore, can be very expensive for higher accuracy.

Moreover, besides the approximation error, one has to estimate possible rounding errors occurring during the numerical computations, in particular when objects are rotated and/or the allocation points have to be within a given lattice (which is sometimes required).

Furthermore, the effort to compute the union or the intersection of two objects is strongly dependent on the accuracy of approximation.

An important aspect when approximating real objects is that structural properties (such as convexity) are conserved, or an outer-approximation can be guaranteed. This is needed to ensure that a pattern of the approximated objects induces a feasible pattern of the original objects.

## 1.5 A Nonlinear Optimization Model, $\Phi$ -Functions

We consider a cutting or packing problem where all objects  $O_i$ ,  $i \in I = \{1, \dots, m\}$ , have to be placed within a region  $B$  of minimal size. In order to get a more practicable model in comparison to the set-theoretical model (1.5)–(1.9) (minimizing the material usage), all restrictions have to be formulated in an analytical form. These are, on the one hand, the non-overlapping conditions, and on the other hand, the containment conditions.

In the following, we describe the concept of  $\Phi$ -functions which was proposed by Stoyan [5] in 1983. A  $\Phi$ -function characterizes the mutual position and interaction of two objects  $O_i(u_i)$  and  $O_j(u_j)$  of  $\mathbb{R}^d$  where  $u_i$  and  $u_j$  denote the allocation points (translation vectors) of them. For simplicity, here we do not consider rotations of the objects.

**Definition 1.2** A continuous function  $\Phi_{ij}(u_i, u_j) : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ , which possesses the following properties, is called  $\Phi$ -function of two objects  $O_i$  and  $O_j$ :

$$\Phi_{ij}(u_i, u_j) = \begin{cases} > 0, & \text{if } O_i(u_i) \cap O_j(u_j) = \emptyset, \\ = 0, & \text{if } \text{int}(O_i(u_i)) \cap O_j(u_j) = \emptyset \text{ and } O_i(u_i) \cap O_j(u_j) \neq \emptyset, \\ < 0, & \text{if } \text{int}(O_i(u_i)) \cap O_j(u_j) \neq \emptyset. \end{cases}$$

It is obvious, the definition of a  $\Phi$ -function of two objects which can be translated and rotated, can be done in a similar way (see Chap. 12). Due to the definition, we have that two objects  $O_i(u_i)$  and  $O_j(u_j)$  do not overlap if and only if  $\Phi_{ij}(u_i, u_j) \geq 0$  holds.

Moreover, the condition that object  $O_i$  is completely contained in region  $B$  can be modeled by means of a  $\Phi$ -function, too. More precisely, this can be done by means of a  $\Phi$ -function of object  $O_i$  and the complementary region  $B_0$  of the interior  $\text{int}(B)$  of  $B$ , i.e.,

$$B_0 := \text{cl}(\mathbb{R}^d \setminus B) = \mathbb{R}^d \setminus \text{int}(B).$$

In the considered problem of minimizing the material usage, set  $B_0$  depends on one or several parameters (i.e. its dimensions). We denote these parameters by  $p \in \mathbb{R}^q$ . In the case of packing rectangles into a strip of given width and minimal length, we have only one parameter, the length of the strip, i.e.  $q = 1$ . However, in that case

where a rectangular region  $B$  with minimal area is searched, two parameters exist: the length and the width of  $B$ , i.e.  $p \in \mathbb{R}^2$ . Without loss of generality, we can assume that  $B_0$  is not translated and has its reference and allocation point at  $0 \in \mathbb{R}^2$ . Then the non-overlapping of  $B_0$  and object  $O_i$ , i.e. the containment of  $O_i$  within  $B$ , can be modeled in the form  $\phi_i(u_i, p) := \Phi_i(u_i, 0, p) \geq 0$  where  $\Phi_i$  is a  $\Phi$ -function of  $B_0$  and  $O_i$ .

Let

$$u = (u_1, \dots, u_m)^\top \in \mathbb{R}^{d \cdot m}$$

denote the vector of all translation vectors, and let  $F : \mathbb{R}^{dm+q} \rightarrow \mathbb{R}$  define a corresponding target function. Then we obtain the following model:

### Nonlinear model for minimizing the material usage

$$\min F(u, p) \quad \text{s.t.} \quad (u, p) \in \Omega \quad \text{where} \quad (1.11)$$

$$\Omega = \{(u, p) \in \mathbb{R}^{dm+q} : \Phi_{ij}(u_i, u_j) \geq 0, 1 \leq i < j \leq m, \phi_i(u_i, p) \geq 0, i = 1, \dots, m\}. \quad (1.12)$$

Set  $\Omega$  represents the set of feasible arrangements. In general, the functions  $\Phi_{ij}$  and  $\phi_i$  are nonlinear. Properties of  $\Omega$  and  $\Phi$ -functions are investigated in more detail in Chap. 12.

*Example 1.3 (Packing of Circles)* We consider the *Circle Packing Problem*. The task is to find a circle  $C$  of minimal radius  $R$  and center  $(0,0)$  such that  $m$  circles  $C_i$  with radii  $r_i, i \in I = \{1, \dots, m\}$  can be placed completely within  $C$  without overlap. Furthermore, a corresponding pattern has to be found.

Let  $\rho(u, v) := \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$  denote the (Euclidean) distance between two points  $u = (u_1, u_2) \in \mathbb{R}^2$  and  $v = (v_1, v_2) \in \mathbb{R}^2$ . Then circle  $C_i(u_i)$  is completely contained in  $C$  if and only if  $\rho(u_i, 0) \leq R - r_i$  where  $u_i \in \mathbb{R}^2$  denotes the center of the translated object  $C_i(u_i)$ . Hence,  $\Phi_i(u_i, R) := R - r_i - \rho(u_i, 0)$  is an appropriate  $\Phi$ -function of  $C_i$  and  $C_0 := \text{cl}(\mathbb{R}^2 \setminus C)$ .

Furthermore, two circles  $C_i$  and  $C_j$  with  $i \neq j$  do not overlap if and only if  $\rho(u_i, u_j) \geq r_i + r_j$ . Consequently,  $\Phi_{ij}(u_i, u_j) := \rho(u_i, u_j) - r_i - r_j$  is an appropriate  $\Phi$ -function of  $C_i$  and  $C_j$ . Summarizing, we obtain the following nonlinear model according to (1.11) and (1.12):

$$\min R \quad \text{s.t.} \quad \phi_i(u_i, R) \geq 0, i \in I, \quad \Phi_{ij}(u_i, u_j) \geq 0, i \neq j, i, j \in I. \quad \blacksquare$$

Next we consider the problem of maximizing the material utilization. In that case, the region  $B$  is given. Instead of the unknown dimensions of  $B$ , now we have decision variables as parameters. Variable  $a_i \in \{0, 1\}$  indicates whether object  $i$  is placed (then  $a_i = 1$ ) or not (then  $a_i = 0$ ),  $i \in I$ . We denote the value (profit) of object  $i \in I$  by  $c_i$ . According to (1.1)–(1.4) we obtain the following model in which rotation of objects is not considered:

### Nonlinear model for maximizing material utilization

$$\max \sum_{i \in I^*} c_i = \max \sum_{i \in I} c_i a_i \quad \text{s.t.} \quad (u, a) \in \Omega \quad \text{where}$$

$$\begin{aligned} \Omega = \{(u, a) \in \mathbb{R}^{dm} \times \mathbb{B}^m : \\ \Phi_{ij}(u_i, u_j) \geq 0, \forall i \neq j : a_i a_j = 1, \phi_i(u_i) \geq 0, \forall i : a_i = 1\} \end{aligned}$$

where  $a = (a_1, \dots, a_m)^\top \in \mathbb{B}^m$ . Besides the allocation constraints we have, because of the  $m$  binary variables, some additional difficulties for solution approaches. Which of the  $2^m$  subsets  $I^* \subseteq I$  is an optimal one?

In order to construct a  $\Phi$ -function  $\Phi_{ij}$  for two objects  $O_i$  and  $O_j$  with  $i \neq j$  we assume that they are given as follows:

$$O_i = \{x \in \mathbb{R}^d : f_i(x) \leq 0\} \quad \text{and} \quad O_j = \{x \in \mathbb{R}^d : f_j(x) \leq 0\}$$

Evidently, we have  $O_i(u_i) \cap \text{int}(O_j(u_j)) = \emptyset$  if and only if  $x \in O_i(u_i)$  implies  $x \notin \text{int}(O_j(u_j))$ , or equivalently, if

$$x - u_i \in O_i \Rightarrow x - u_j \notin \text{int}(O_j).$$

Hence, if  $f_i(x - u_i) \leq 0$  holds, then  $f_j(x - u_j) \geq 0$  has to be fulfilled. This issue can be modeled by

$$\max\{f_i(x - u_i), f_j(x - u_j)\} \geq 0 \quad \forall x \in \mathbb{R}^d$$

since  $f_i(x - u_i) > 0$  and  $f_j(x - u_j) > 0$  for all  $x \notin O_i(u_i) \cup O_j(u_j)$ . Consequently, a corresponding  $\Phi$ -function is defined by

$$\Phi_{ij}(u_i, u_j) := \min_{x \in \mathbb{R}^d} \max\{f_i(x - u_i), f_j(x - u_j)\}.$$

Since the mutual position of the two objects is determined by  $u := u_j - u_i$ , a  $\Phi$ -function can also be represented as follows:

$$\Phi_{ij}(u_i, u_j) := \phi_{ij}(u_j - u_i) \quad \text{with} \quad \phi_{ij}(u) := \min_{x \in \mathbb{R}^d} \max\{f_i(x), f_j(x - u)\}.$$

In general, we have to guarantee that some regularity condition is fulfilled by the objects so that the minimum over  $\mathbb{R}^d$  can be computed. For example, if  $O_i$  and  $O_j$  are convex polygons, then the computation of the minimum can be reduced to a finite number of test-points (see Chap. 12).

In a similar way, we can obtain a  $\Phi$ -function for  $O_i$  and  $B_0 := \mathbb{R}^d \setminus \text{int}(B)$ . For more detailed information related to  $\Phi$ -functions of two- and three-dimensional objects we refer to [1, 6, 7].

Within the related literature, frequently some specific notations are used for the set of difference vectors  $u := u_j - u_i$ . The set

$$\{u \in \mathbb{R}^d : \Phi_{ij}(0, u) = 0\},$$

where  $\Phi_{ij}$  denotes a  $\Phi$ -function of two (real) objects  $O_i$  and  $O_j$ , is called *hodograph* [8]. Moreover, set

$$U_{ij} = \{u \in \mathbb{R}^d : \Phi_{ij}(0, u) \leq 0\}$$

is congruent to the *Minkowski sum*

$$O_i \oplus (-O_j) := \{u \in \mathbb{R}^d : u = v - w, v \in O_i, w \in O_j\},$$

and we have

$$O_i \oplus (-O_j) = \{u \in \mathbb{R}^d : \Phi_{ij}(u) \leq 0\}.$$

In this context, the notation *no-fit polygon* is widespread. In case of  $u \in \text{int}(U_{ij})$  the two objects  $O_i(0)$  and  $O_j(u)$  overlap each other. That means that they do not form a feasible arrangement. The notations introduced above, frequently appear in the description of numerous algorithms to arrange non-regular objects.

## 1.6 Integer Linear Programming Models

In this section, we exemplarily address a problem of maximizing the material utilization and one of minimizing the material usage. Such problems occur in many industrial branches, for instance, in wood and steel cutting. Related solution approaches can be found in Chaps. 2 and 4.

Suppose that pieces of lengths  $\ell_i$ ,  $i \in I := \{1, \dots, m\}$ , have to be obtained from raw material of length  $L$ , e.g., from round timber or logs. In case of maximizing the material utilization a single optimal pattern has to be found which exploits the material in a best possible way. In a problem of minimizing the material usage the target is to find the minimum number of copies of the raw material which is needed to fulfill the order demands, i.e., to produce  $b_i$  pieces of length  $\ell_i$  for each  $i \in I$ . Obviously, besides the minimum number one is also interested in the patterns related to the minimum.

An integer linear programming (ILP) model of the one-dimensional cutting problem of maximizing the material utilization can be stated as follows: let  $c_i$  denote the yield (profit) of a piece of length  $\ell_i$  for  $i \in I$  if obtained from the raw material. For the one-dimensional cutting problem, a vector  $a = (a_1, \dots, a_m)^\top \in \mathbb{Z}_+^m$

represents a feasible pattern if and only if

$$\sum_{i \in I} \ell_i a_i \leq L,$$

i.e., if the total sum of piece lengths, to be cut according to pattern  $a$ , does not exceed the length of the raw material. In order to obtain the maximal yield from a single item of raw material we have to compute which pieces should be cut. Let  $a_i \in \mathbb{Z}_+$  ( $i \in I$ ) denote the number of copies of piece type  $i$  which are obtained. Then a model of optimal round timber cutting is as follows:

### ILP model of maximizing the material utilization

$$\begin{aligned} \max \quad & \sum_{i \in I} c_i a_i && \text{s.t.} \\ & \sum_{i \in I} \ell_i a_i \leq L, \quad a_i \in \mathbb{Z}_+, \quad i \in I. \end{aligned}$$

According to the objective function, the material utilization obtained by an optimal pattern is maximal with respect to the yield coefficients  $c_i$ . Setting  $c_i = \ell_i$  for all  $i \in I$  we receive a maximal material utilization in the proper sense. Solution approaches for this kind of problems are described in Chap. 2.

Next we consider the problem of minimizing the material usage in the one-dimensional case. That problem is known as *one-dimensional Cutting Stock Problem* or *one-dimensional Bin Packing Problem*. Here we need to consider various different patterns. Let the nonnegative integer vector  $a^j = (a_{1j}, \dots, a_{mj})^\top \in \mathbb{Z}_+^m$ , represent pattern  $j$ . The coefficient  $a_{ij}$  indicates how many copies of piece type  $i$  (i.e. the length  $\ell_i$ ) are obtained when pattern  $j$  is applied once. Let  $J$  denote an appropriate index set of all feasible patterns.

In order to formulate an ILP model we define an *integer variable*  $x_j$  for each pattern  $a^j$  which indicates how often  $a^j$  is applied in the solution. In this way, we obtain a general, pattern-oriented model of the one-dimensional cutting stock problem:

### ILP model for minimizing the material usage

$$\begin{aligned} \min \quad & \sum_{j \in J} x_j && \text{s.t.} \\ & \sum_{j \in J} a_{ij} x_j \geq b_i, \quad i \in I, \quad x_j \in \mathbb{Z}_+, \quad j \in J. \end{aligned}$$

The two principal difficulties for solving real instances consist in finding integer solutions, and in the, in general, extreme high number of variables. More detailed investigations are provided in Chap. 4.

## 1.7 Further Opportunities of Modeling

Besides the approaches to model cutting and packing problems as discussed in the previous sections, there exist further possibilities. We restrict ourselves to the so-called *permutation model*.

In the *two-dimensional orthogonal strip packing problem* (2SPP, Chap. 7)  $m$  rectangles and a strip of known width are given. The task is to find an arrangement of all  $m$  rectangular pieces within the strip such that the occupied height of it is minimal. It is well-known that this problem is *NP-hard* [2], i.e., it is very likely that there does not exist any exact algorithm for this problem which finds a proved optimal solution in polynomial time. Therefore, numerous heuristics have been developed. Some of them construct a feasible pattern based on a given sequence of the rectangles in the way that one rectangle is packed after another. Consequently, the following optimization problem is of interest:

### Permutation model for the strip packing problem

Find a permutation  $\pi^*$  of  $\{1, \dots, m\}$  such that the chosen heuristic computes a pattern of minimal height.

Strategies for (approximately) solving such problems, frequently called *meta-heuristics*, are addressed in Sect. 7.4.2.

Besides the task to find  $\pi^*$  there arise further important questions concerning the performance of such an approach. For example, for the BL heuristic (Sect. 7.3) it is known that there exist instances of the 2SPP for which no sequence of rectangles enables the BL heuristic to construct an optimal pattern.

It is to notice that also *graph-theoretical models* are used to formulate cutting and packing problems mathematically. We consider examples of that kind in Sects. 2.3 and 4.8.

## 1.8 Exercises

**Exercise 1.1** Consider the round timber cutting problem. How can a positive (saw) kerf be easily regarded within a related model?

**Exercise 1.2** Show that  $D(O_i(u_i), \theta_i) = D(O_i(0), \theta_i)(u_i)$ . holds

**Exercise 1.3** Consider the case  $d = 1$  and defect-free homogeneous raw material. Formulate a general model of maximizing the material utilization. How can the 0/1 knapsack problem be obtained?

**Exercise 1.4** Consider pieces of length  $\ell_1 = 17$  and  $\ell_2 = 21$ , and raw material of length  $L = 136$ . Which pattern maximizes the material utilization? Which solution appears if additionally pieces of length  $\ell_3 = 23$  and  $\ell_4 = 24$  can be cut?

**Exercise 1.5** Which object is represented by

$$\{(x_1, x_2) : x_1^2 + x_2^2 - 25 \leq 0, -4(x_1 + 1)^2 - 9x_2^2 + 36 \leq 0, (x_1 - 2)^2 + x_2^2 - 16 \leq 0\}?$$

**Exercise 1.6** Logical operators can also be applied to represent objects. Which non-convex object is described by the term  $(f_1 \wedge f_2 \wedge f_3) \vee (f_1 \wedge f_2 \wedge f_4)$  where  $f_1(x, y) = (x - 3)^2 + y^2 \leq 36, f_2(x, y) = (x + 3)^2 + y^2 \leq 36, f_3(x, y) = (x - 1)^2 + (y + 3)^2 \leq 25, f_4(x, y) = (x + 1)^2 + (y + 3)^2 \leq 25$ ?

**Exercise 1.7** Without loss of generality, in the two-dimensional case we can use the convention that an object always lies on the *left* side of a bounding straight line  $\overline{PQ}$  which goes from point  $P$  to point  $Q$ . Which object is represented by the sequence  $(0, 0), (1, 1), (2, 0), (2, 3), (1, 2), (0, 3), (0, 0)$  of corner points? Give a description by means of convex half-planes.

**Exercise 1.8** Compute the Minkowski sum of the two objects

$$O_1 = \{(x, y) : 0 \leq x \leq 2, 0 \leq y \leq 3\} \text{ and } O_2 = \{(x, y) : y \geq 0, y \leq 2x, y \leq 4 - 2x\}.$$

**Exercise 1.9** Consider a rectangular container  $C$  of given length  $L$  and width  $W$ . Which minimal height  $H$  of the container is needed to pack all boxes of size  $\ell_i \times w_i \times h_i, i \in I := \{1, \dots, m\}$ ? For the sake of simplicity, we assume that rotation of the boxes is not allowed and that they have to be packed with the given orientation. Furthermore, let

$$C(H) := \{(x, y, z) : 0 \leq x \leq L, 0 \leq y \leq W, 0 \leq z \leq H\},$$

$$O_i := \{(x, y, z) : 0 \leq x \leq \ell_i, 0 \leq y \leq w_i, 0 \leq z \leq h_i\}, \quad i \in I.$$

Find functions  $f_{ik}(u_i, h) : \mathbb{R}^4 \rightarrow \mathbb{R}, k = 1, \dots, 6, i \in I$  such that

$$\Phi_i(u_i, h) := \min\{f_{ik}(u_i, h) : k = 1, \dots, 6\}$$

is a  $\Phi$ -function of  $O_i$  and  $\text{cl}(\mathbb{R}^3 \setminus C(H))$ . Define a  $\Phi$ -function of  $O_i$  and  $O_j, i, j \in I, i \neq j$ .

## 1.9 Solutions

**To Exercise 1.1** Let  $L$  denote the length of the material,  $\ell_i, i \in I$ , the lengths of the piece types, and  $\Delta > 0$  the size of the (saw) kerf. If, according to a pattern  $a = (a_1, \dots, a_m)^\top \in \mathbb{Z}_+^m$ , exactly  $k = \sum_{i \in I} a_i$  pieces are cut then a total waste of size  $(k-1)\Delta$  has to be regarded, i.e., pattern  $a$  has to fulfill the condition  $\sum_{i \in I} \ell_i a_i \leq L - (k-1)\Delta$  to be feasible. This inequality holds if and only if  $\sum_{i \in I} (\ell_i + \Delta)a_i \leq L + \Delta$  is satisfied.

Hence, if we replace the length  $L$  of the material by  $L + \Delta$  and the length  $\ell_i$  of piece type  $i$  by  $\ell_i + \Delta$ ,  $i \in I$ , then we obtain an instance of the one-dimensional cutting stock problem where the kerf is regarded.

**To Exercise 1.2** Since the center of rotation and the reference point coincide, the translation and rotation are commutative in the given sense.

**To Exercise 1.3** In the one-dimensional case, each piece  $O_i$  and the region  $B$  can be described by the interval  $[0, \ell_i]$ , and  $[0, b]$ , respectively. Obviously, rotations do not have to be applied. Hence, we have

$$\max_{i \in I^*} c_i \quad \text{s. t.}$$

$$O_i(u_i) \subseteq B \Leftrightarrow 0 \leq u_i, u_i + \ell_i \leq b, \quad i \in I^*,$$

$$\text{int}(O_i(u_i)) \cap O_j(u_j) = \emptyset \Leftrightarrow (u_i + \ell_i \leq u_j) \vee (u_j + \ell_j \leq u_i), \quad \forall i \neq j, i, j \in I^*,$$

$$I^* \subseteq I.$$

Since defect-free homogeneous material is assumed, the real positions  $u_i$  are less important. If we define 0/1 variables  $x_i$  by  $x_i := 1$  if  $i \in I^*$  and  $x_i := 0$  for  $i \in I \setminus I^*$ , then for each solution  $(u_i, i \in I^*)$  there exists a solution of the 0/1 knapsack problem

$$\max_{i \in I} c_i x_i \quad \text{s. t.} \quad \sum_{i \in I} \ell_i x_i \leq b, \quad x_i \in \{0, 1\}, \quad i \in I.$$

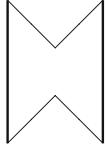
Using an indirect proof one can show that a solution of the 0/1 knapsack problem corresponds to a solution of the problem mentioned above.

**To Exercise 1.4** We obtain for two piece types the unique optimal pattern  $a = (8, 0)^\top$ , and for four piece types five optimal patterns:  $a^1 = (8, 0, 0, 0)^\top$ ,  $a^2 = (4, 1, 1, 1)^\top$ ,  $a^3 = (1, 0, 1, 4)^\top$ ,  $a^4 = (0, 2, 2, 2)^\top$ ,  $a^5 = (0, 1, 5, 0)^\top$ .

**To Exercise 1.5** Intersection of two circles and the complement of an ellipse.

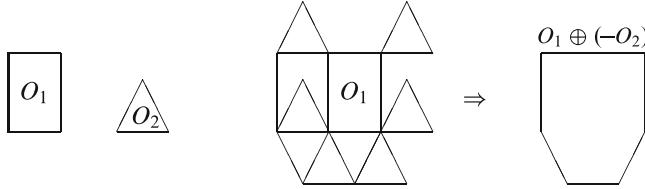
**To Exercise 1.6** Union of intersections of three circles each.

**To Exercise 1.7** We obtain the following object and its representation:



$$\begin{aligned} O &= \{(x, y) : x \geq 0, x \leq y \leq 3 - x\} \\ &\cup \{(x, y) : x \leq 2, 2 - x \leq y \leq 1 + x\} \\ &= \{(x, y) : 0 \leq x \leq 1, x \leq y \leq 3 - x\} \\ &\cup \{(x, y) : 1 \leq x \leq 2, 2 - x \leq y \leq 1 + x\} \end{aligned}$$

**To Exercise 1.8** The curve of the reference point (lower left corner) of  $O_2$ , obtained when shifting  $O_2$  around  $O_1$ , encloses the Minkowski sum.



**To Exercise 1.9** Let  $u_i = (x_i, y_i, z_i)^\top$ ,  $i \in I$ , and  $u = (x, y, z)^\top$ . Because of

$$O_i(u_i) \subseteq C(h) \Leftrightarrow x_i \geq 0, y_i \geq 0, z_i \geq 0, x_i + \ell_i \leq L, y_i + w_i \leq W, z_i + h_i \leq h$$

we obtain with

$$\begin{aligned} f_{i1}(u, h) &:= x, f_{i2}(u, h) := y, f_{i3}(u, h) := z, \\ f_{i4}(u, h) &:= L - x - \ell_i, f_{i5}(u, h) := W - y - w_i, f_{i6}(u, h) := h - z - h_i \end{aligned}$$

the desired  $\Phi$ -function  $\Phi_i(u_i, h) = \min\{f_{ik}(u_i, h) : k = 1, \dots, 6\}$  for  $O_i$  and  $\text{cl}(\mathbb{R}^3 \setminus C(h))$ . Because of

$$\begin{aligned} O_i(u_i) \cap \text{int}(O_j(u_j)) &= \emptyset \Leftrightarrow \\ (x_i + \ell_i \leq x_j) \vee (y_i + w_i \leq y_j) \vee (z_i + h_i \leq z_j) \\ \vee (x_j + \ell_j \leq x_i) \vee (y_j + w_j \leq y_i) \vee (z_j + h_j \leq z_i) \end{aligned}$$

we define

$$\begin{aligned} g_1^{ij}(u_i, u_j) &:= x_j - x_i - \ell_i, g_2^{ij}(u_i, u_j) := y_j - y_i - w_i, g_3^{ij}(u_i, u_j) := z_j - z_i - h_i, \\ g_4^{ij}(u_i, u_j) &:= x_i - x_j - \ell_j, g_5^{ij}(u_i, u_j) := y_i - y_j - w_j, g_6^{ij}(u_i, u_j) := z_i - z_j - h_j, \end{aligned}$$

and obtain a  $\Phi$ -function of the form  $\Phi_{ij}(u_i, u_j) = \min\{g_k^{ij}(u_i, u_j) : k = 1, \dots, 6\}$  for  $O_i$  and  $O_j$ ,  $i \neq j$ ,  $i, j \in I$ .

## References

1. J.A. Bennell, G. Scheithauer, Y. Stoyan, T. Romanova, Tools of mathematical modeling of arbitrary object packing problems. Ann. OR **179**, 343–368 (2010)
2. M.R. Garey, D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
3. J.C. Herz, Recursive computational procedure for two-dimensional stock cutting. IBM J. Res. Dev. **16**, 462–469 (1972)

4. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988)
5. Y.G. Stoyan, Mathematical methods for geometric design, in *Advances in CAD/CAM, Proceedings of PROLAMAT 82, Leningrad*, Amsterdam, ed. by T.M.R. Ellis, O.J. Semenkoc (1983), pp. 67–86
6. Y.G. Stoyan, G. Scheithauer, N. Gil, T. Romanova,  $\phi$ -function for complex 2d-objects. *4OR* **2**(1), 69–84 (2002)
7. Y.G. Stoyan, J. Terno, G. Scheithauer, N. Gil, T. Romanova, Construction of a  $\phi$ -function for two convex polytopes. *Appl. Math.* **29**, 199–218 (2002)
8. J. Terno, R. Lindemann, G. Scheithauer, *Zuschnittprobleme und ihre praktische Lösung* (Verlag Harri Deutsch, Thun und Frankfurt/Main, 1987)

# Chapter 2

## Knapsack Problems

The *Knapsack Problem* is a linear integer programming problem with only one constraint which defines a relation between the variables. In this respect, it is the ‘simplest’ integer optimization problem. Since the knapsack problem already possesses essential difficulties of integer programming, it is subject of numerous investigations. It is well-known that the knapsack problem belongs to the class of *NP-hard* problems [4], i.e., with high probability there does not exist any algorithm for the knapsack problem which computes an optimal solution in polynomial time. For a detailed study of various variants of the knapsack problem we refer to [7] and [8].

Within this chapter we present basic techniques to solve the knapsack problem which often can be used in solution approaches for other cutting and packing problems.

### 2.1 Problem Statement

A common verbal formulation of the (standard) *Knapsack Problem* is as follows: let a knapsack of volume (capacity)  $b$  and  $m$  types of pieces with volume  $a_i$  and value  $c_i$ ,  $i \in I = \{1, \dots, m\}$  be given. Find such a set of pieces whose total volume does not exceed  $b$  and which maximizes the total value. Another formulation results if, e.g., the coefficients  $a_i$  represent the weight of a piece of type  $i$  and  $b$  the maximal admissible load. Note that several pieces of the same type can be packed. Therefore, we assume  $a_i \neq a_j$  for all  $i \neq j$  in the following.

With respect to cutting and packing problems, the problem of round timber cutting, considered in Exercise 1.1, is equivalent to a knapsack problem where  $b$  is the timber length,  $a_i$ ,  $i \in I$ , are the piece lengths, and  $c_i$ ,  $i \in I$ , the piece values.

Let the nonnegative integer variable  $x_i$  denote the number of copies of piece type  $i$  which are used to fill the knapsack, i.e.  $x_i \in \mathbb{Z}_+$  for all  $i \in I$ . Then we obtain the following formulation of the (standard) knapsack problem:

$$f(b) := \max \left\{ \sum_{i \in I} c_i x_i : \sum_{i \in I} a_i x_i \leq b, x_i \in \mathbb{Z}_+, i \in I \right\}. \quad (2.1)$$

For the sake of simplicity, we always assume that  $c_i > 0$ ,  $0 < a_i \leq b$ ,  $a_i \neq a_j$  for all  $i \neq j$ ,  $i, j \in I$  hold, and that all input data are integers. For a more general case, see Exercise 2.1. Using  $a = (a_1, \dots, a_m)^\top$ ,  $c = (c_1, \dots, c_m)^\top$ , and  $x = (x_1, \dots, x_m)^\top$  the knapsack problem (2.1) can be written in the compact form

$$f(b) := \max \{c^\top x : a^\top x \leq b, x \in \mathbb{Z}_+^m\}.$$

In the following we denote, for short, a knapsack problem with input data  $a, c, b$  by  $\text{KP}(c, a, b)$ , or likewise, by  $\text{KP}(b)$ . The latter notation is used to indicate the dependence on parameter  $b$  while the coefficients  $a_i$  and  $c_i$ ,  $i \in I$ , are maintained. Hence,  $f(y)$  represents the optimal value of  $\text{KP}(y)$  for any  $y \in \mathbb{Z}_+$ .

The important special case where  $x_i \in \{0, 1\}$  is demanded for all  $i \in I$ , is known as *0/1 Knapsack Problem*. Efficient algorithms for the standard and the 0/1 knapsack problem and for related problems are listed in [7] and [8].

In the following we present some basic algorithms for solving the (standard) knapsack problem. Their principles are applied in various solution approaches for other cutting and packing problems.

## 2.2 Algorithm of Gilmore and Gomory

The algorithm proposed by Gilmore and Gomory [5] is based on the principle of *Dynamic Programming* (DP, see e.g. [9]). For that reason we define the optimal value  $F(k, y)$  of the reduced knapsack problem with the piece types  $1, \dots, k$ , where  $k \in \{1, \dots, m\}$ , and the volume  $y$  of the knapsack,  $y \in \{0, 1, \dots, b\}$ :

$$F(k, y) := \max_{x_1, \dots, x_k} \left\{ \sum_{i=1}^k c_i x_i : \sum_{i=1}^k a_i x_i \leq y, x_i \in \mathbb{Z}_+, i \in I_k \right\}$$

where  $I_k := \{1, \dots, k\}$ . Obviously, for  $k = 1$  we have  $F(1, y) = c_1 \lfloor y/a_1 \rfloor$  for all  $y = 0, 1, \dots, b$ , as well as  $F(k, 0) = 0$  for  $k = 1, \dots, m$ . The symbol  $\lfloor q \rfloor$  denotes the largest integer not larger than  $q \in \mathbb{R}$ , i.e.,  $q$  is rounded down to the nearest integer. Hence, in order to solve the knapsack problem (2.1) we have to compute  $F(m, b)$  since  $f(b) = F(m, b)$ . For the reduced knapsack problem  $F(k, y)$ , let  $\alpha_k = \alpha_k(y) := \lfloor y/a_k \rfloor$ . We consider the  $\alpha_k + 1$  values which variable  $x_k$  can take

in a feasible solution of  $F(k, y)$  and obtain, because of the linearity of the objective function and the restriction,

$$\begin{aligned} F(k, y) &= \max_{j=0, \dots, a_k} \max_{x_1, \dots, x_{k-1}} \left\{ jc_k + \sum_{i=1}^{k-1} c_i x_i : \sum_{i=1}^{k-1} a_i x_i \leq y - ja_k, x_i \in \mathbb{Z}_+, i \in I_{k-1} \right\} \\ &= \max_{j=0, \dots, a_k} \left\{ jc_k + \max_{x_1, \dots, x_{k-1}} \left\{ \sum_{i=1}^{k-1} c_i x_i : \sum_{i=1}^{k-1} a_i x_i \leq y - ja_k, x_i \in \mathbb{Z}_+, i \in I_{k-1} \right\} \right\}. \end{aligned}$$

Herewith we obtain a first recurrence formula for  $y = 0, 1, \dots, b$  and  $k = 2, \dots, m$ :

$$F(k, y) = \max \{ jc_k + F(k-1, y - ja_k) : j = 0, \dots, \lfloor y/a_k \rfloor \}.$$

If we consider the two cases

$$x_k = 0 \quad \vee \quad x_k \geq 1,$$

then another recursion results in a similar way for  $y \geq a_k$ :

$$F(k, y) = \max \{ F(k-1, y), c_k + F(k, y - a_k) \}. \quad (2.2)$$

Formula (2.2) constitutes the basis of the Algorithm of Gilmore and Gomory which is depicted in Fig. 2.1. It is easy to see, the algorithm computes  $F(m, b)$  in  $O(mb)$  time where  $O(\cdot)$  denotes the Landau symbol. That means, there exists a number  $\rho$  which is independent on the input data, such that the number of computational operations is bounded above by  $\rho mb$ . Algorithms whose worst-case performance can be estimated in this form (i.e., is bounded by a polynomial of the input data) are called *pseudo-polynomial*.

*Example 2.1* We consider the following instance of the knapsack problem

$$\begin{aligned} \max \quad & 5x_1 + 10x_2 + 12x_3 + 6x_4 \quad \text{s.t.} \\ & 4x_1 + 7x_2 + 9x_3 + 5x_4 \leq 15, \quad x_i \in \mathbb{Z}_+, \quad i = 1, 2, 3, 4. \end{aligned}$$

### Algorithm of Gilmore and Gomory

Input:  $c, a, b$

Output:  $F(m, y)$  for  $y = 0, 1, \dots, b$

- (1) Set  $F(1, y) := c_1 \lfloor y/a_1 \rfloor$  for  $y = 0, 1, \dots, b$ .
- (2) For  $k = 2, \dots, m$ :
- (3)     for  $y = 0, \dots, a_k - 1$  set  $F(k, y) := F(k-1, y)$ ,
- (4)     for  $y = a_k, \dots, b$  set
- (5)          $F(k, y) := \max \{ F(k-1, y), F(k, y - a_k) + c_k \}$ .

**Fig. 2.1** Algorithm of Gilmore and Gomory

**Table 2.1** Algorithm of Gilmore and Gomory: optimal values  $F(k, y)$  of Example 2.1

| $k \setminus y$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|-----------------|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| 1               | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 5  | 10 | 10 | 10 | 10 | 15 | 15 | 15 | 15 |
| 2               | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 10 | 10 | 10 | 10 | 15 | 15 | 15 | 20 | 20 |
| 3               | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 10 | 10 | 12 | 12 | 15 | 15 | 17 | 20 | 20 |
| 4               | 0 | 0 | 0 | 0 | 5 | 6 | 6 | 10 | 10 | 12 | 12 | 15 | 16 | 17 | 20 | 20 |

### Algorithm of Gilmore and Gomory with index information

Input:  $c, a, b$

Output:  $F(m, y)$  for  $y = 0, 1, \dots, b$ . and solution  $x$  of  $F(m, b)$

- (1) Set  $F(1, y) := c_1 \lfloor y/a_1 \rfloor$  for  $y = 0, 1, \dots, b$ ,
- (2) set  $i(y) := 0$  for  $y = 1, \dots, a_1 - 1$  and  $i(y) := 1$  for  $y = a_1, \dots, b$ .
- (3) For  $k = 2, \dots, m$ :
- (4) for  $y = 0, \dots, a_k - 1$  set  $F(k, y) := F(k - 1, y)$ ,
- (5) for  $y = a_k, \dots, b$ :
- (6) if  $F(k - 1, y) < F(k, y - a_k) + c_k$ , then set
- (7)  $F(k, y) := F(k, y - a_k) + c_k$ ,  $i(y) := k$ ,
- (8) otherwise set  $F(k, y) := F(k - 1, y)$ .
- (9) Set  $x_i := 0$  for  $i = 1, \dots, m$  and  $y := b$ .
- (10) While  $i(y) > 0$  set  $i := i(y)$ ,  $x_i := x_i + 1$ ,  $y := y - a_i$ .

**Fig. 2.2** Algorithm of Gilmore and Gomory with index information

The computation of  $F(k, y)$  is shown in Table 2.1. Therein, the optimal values  $F(k, y)$  are calculated row-wise. We obtain  $f(15) = F(4, 15) = 20$  as optimal value of the knapsack problem. ■

Since the Algorithm of Gilmore and Gomory is based on the principle of dynamic programming, each value  $F(k, y)$  constitutes the optimal value of the reduced knapsack problem  $F(k, y)$ . In that case, when only a solution of  $F(m, b)$  has to be computed, and the values  $F(k, y)$  are not of interest for  $k < m$ , then it is not necessary to store a table with  $m$  rows and  $b$  columns. Since the computation of  $F(k, y)$  for  $k > 1$  according to formula (2.2) requires only a function value of the previous stage (row)  $k - 1$ , namely  $F(k - 1, y)$ , and one of the current stage  $k$ , the values of the stages  $j = 1, \dots, k - 2$  are not needed any longer. Moreover, the value  $F(k, y)$  can be stored on the place of  $F(k - 1, y)$  since the latter value is not used in the ongoing computations. Consequently, a vector of lengths  $b$  is sufficient as memory space to perform the recursion. In particular, at last  $f(y)$  is obtained for all  $y = 1, \dots, b$ .

Besides the optimal values  $F(m, y)$ , mostly a corresponding solution  $x \in \mathbb{Z}_+^m$  is of interest. Such a solution can be obtained by means of index information stored during the computation of  $F(m, b)$ . A respective modification of the Algorithm of Gilmore and Gomory is shown in Fig. 2.2.

**Algorithm of Gilmore and Gomory: identification of a solution**

Input:  $c, a, b, F(m, y)$  for  $y = 0, 1, \dots, b$

Output: solution  $x$  of  $F(m, b)$

- (6) Set  $y := b, x_i := 0, i = 1, \dots, m, i := 1$ .
- (7) While  $F(m, y) > 0$ : if  $F(m, y - a_i) + c_i = F(m, y)$ ,
- (8)     then set  $x_i := x_i + 1, y := y - a_i$ ,
- (9)     otherwise set  $i := i + 1$ .

**Fig. 2.3** Algorithm of Gilmore and Gomory: identification of a solution

An alternative to obtain a solution  $x$  related to  $F(m, b)$  does not require to store index information. In the so-called *Phase 2* of dynamic programming one either uses index information as above, or applies some further computations based on the now available optimal values  $f(y)$ . This variant is presented in Fig. 2.3 as additional steps which have to be performed after the termination of the Algorithm of Gilmore and Gomory.

Consider again Example 2.1. We obtain  $x_1 = 2, x_2 = 1, x_3 = 0, x_4 = 0$ . Further, or even all, solutions can be obtained by simple modifications of the proposed approaches, see Exercise 2.2.

## 2.3 Longest Path Method

The solution approach considered here corresponds to the computation of a longest path in a related directed graph. Similar to the Algorithm of Gilmore and Gomory, the solution strategy is based on dynamic programming, but the computations now are done in the opposite direction.

Corresponding to the knapsack problem  $\text{KP}(c, a, b)$  we construct a *directed graph*  $G = (V, E)$  with vertex set  $V = \{0, 1, \dots, b\}$ . The set  $E$  of arcs (directed edges) is defined by

$$E := \{(j, k) \in V \times V : \exists i \in I \text{ with } j + a_i = k\}.$$

We assign the value (length of the arc)  $c_i$  to arc  $(j, k) \in E$  if  $j + a_i = k$ .

In case of  $\min\{a_i : i \in I\} > 1$  we additionally can add arcs  $(y, y + 1)$  for  $y = 0, 1, \dots, b - 1$  with value (length) 0 to ensure that a path in  $G$  from vertex 0 to vertex  $b$  exists. At appropriate places we will point out whether this is necessary or not. Actually, we go on without it.

In order to compute a solution of the knapsack problem (2.1) a longest path in  $G$  has to be found, i.e., a path with maximal total value.

According to the Algorithm of Ford and Moore for determining a longest path in a directed graph (see e.g. [1]), let  $v(y)$  denote the length (the value) of the longest,

**Longest Path Method**Input:  $c, a, b$ Output: optimal value  $v^*$  and  $v(y)$  for  $y = 0, \dots, b$ 

- (1) Set  $v(y) := 0$  for  $y = 0, \dots, b$ , set  $y := 0$ .
- (2) While  $y + \min\{a_i : i \in I\} \leq b$ :
- (3)     For  $i = 1, \dots, m$ : if  $y + a_i \leq b$ ,
- (4)         set  $v(y + a_i) := \max\{v(y + a_i), v(y) + c_i\}$ .
- (5)     Set  $y := \min\{t : v(t) > v(y)\}$ .
- (6) Compute  $v^* := \max_y \{v(y) : b - \min_{i \in I} a_i < y \leq b\}$ .

**Fig. 2.4** Longest Path Method

already known path from vertex 0 to vertex  $y$ . After termination of the algorithm,  $v(y)$  represents the optimal value, and we have  $v(y) = f(y)$  for all  $y \in V$ .

During the solution process, we iteratively consider all vertices  $y$  in increasing order for which a maximal path is already known, i.e., for which  $v(y) = f(y)$  holds, and we look for extending the path. Adding an arc  $(y, y + a_k)$  with value (length)  $c_k$  defines a path from vertex 0 via vertex  $y$  to vertex  $y + a_k$  with length  $v(y) + c_k$ . If this length is larger than  $v(y + a_k)$ , the current best length, then we update  $v(y + a_k)$  and save the new best path to vertex  $y + a_k$ . Since only arcs  $(j, k)$  with  $j < k$  belong to  $G$ , after the consideration of all arcs  $(y, y + a_k)$  with  $y + a_k \leq b$  and  $k \in I$ , we can identify a minimal vertex  $y'$  with  $y' > y$  for which  $v(y') = f(y')$  holds. This is caused by the fact that all remaining paths still to be constructed, end at vertices  $\bar{y}$  with  $\bar{y} > y'$  (since  $a_k > 0$  for all  $k$ ). Therefore, we also have  $v(y) = f(y)$  for all  $y \leq y'$ . The *Longest Path Method*, stated in Fig. 2.4, is an adaptation of the Algorithm of Ford and Moore to the particular case where  $G$  has only arcs  $(j, k)$  with  $j < k$ .

Again, the numerical expense to compute  $v^* = F(m, b)$  can be estimated by  $O(mb)$  computational operations. The validity of updating  $y$  according to  $y := \min\{t : v(t) > v(y)\}$  is shown in Exercise 2.3.

*Example 2.2* We consider again the knapsack problem of Example 2.1. The computation is presented again in form of a table. In Table 2.2, only the  $y$ -values 0, 4, 7, 9, 11 are used as a start vertex for extending paths. Symbol  $y$  marks the respective iteration step. For the sake of a better overview, only the new found paths are depicted. It is easy to see, only a vector of length  $b$  is needed within an implementation for the computations. A solution corresponding to  $v(b)$  (or  $v(y)$  with  $y < b$ ) can be identified similar to the Algorithm of Gilmore and Gomory. ■

**Table 2.2** Longest Path Method: optimal values  $f(y) = v(y)$  of Example 2.2

| $y$    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8 | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|---|---|---|---|----|---|----|----|----|----|----|----|----|
| $v(y)$ | y |   |   |   | 5 | 6 |   | 10 |   | 12 |    |    |    |    |    |    |
| $v(y)$ |   |   |   |   | y |   |   |    |   |    |    | 15 |    | 17 |    |    |
| $v(y)$ |   |   |   |   |   | y |   |    |   |    |    |    | 16 |    | 18 |    |
| $v(y)$ |   |   |   |   |   |   | y |    |   |    |    |    |    | 20 |    |    |
| $v(y)$ |   |   |   |   |   |   |   | y  |   |    |    |    |    |    |    |    |
| $v(y)$ |   |   |   |   |   |   |   |    | y |    |    |    |    |    |    | 20 |
| $v(y)$ | 0 |   |   |   | 5 | 6 |   | 10 |   | 12 |    | 15 | 16 | 17 | 20 |    |

## 2.4 Branch-and-Bound Algorithms

The basic principle of the *branch-and-bound* method (see e.g. [2, 9]) is as follows: an initial problem, which is ‘hard to solve’, is replaced by a sequence of ‘easier’ subproblems obtained by *branching*. By means of appropriate estimations or computation of bounds for the optimal value of the subproblems, or with dominance considerations, the subproblems are evaluated possibly without their exact solution. This process of branching and bounding is continued until an optimal solution of the initial problem is identified.

In the following we assume, without loss of generality, that the input data are sorted according to

$$\frac{c_i}{a_i} \geq \frac{c_{i+1}}{a_{i+1}}, \quad i = 1, \dots, m-1. \quad (2.3)$$

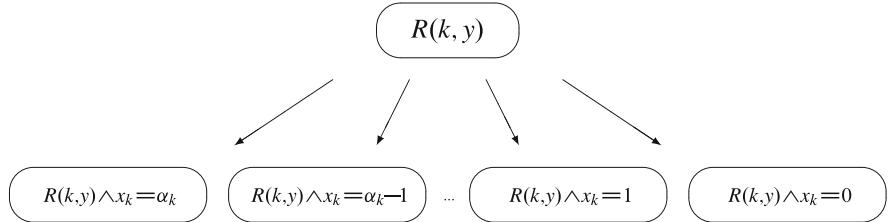
This helps us to formulate a branch-and-bound algorithm for the knapsack problem in a simpler way. Moreover, we define

$$r(k, y) := \max \left\{ \sum_{i=k}^m c_i x_i : \sum_{i=k}^m a_i x_i \leq y, x_i \in \mathbb{Z}_+, i = k, \dots, m \right\}$$

as the optimal value of the *residual problem*  $R(k, y)$ . The residual (or sub-) problem results by fixing the variables  $x_1, \dots, x_{k-1}$ . Thus, if  $x_i = \bar{x}_i$  for  $i = 1, \dots, k-1$ , then we have  $y = b - \sum_{i=1}^{k-1} a_i \bar{x}_i$ .

For *branching* within the branch-and-bound algorithm, and therefore, for defining subproblems, we use the sequence of variables according to the sorting in (2.3). We define a subproblem as follows.

Let  $R(k, y)$  denote the current (sub)problem to be branched (according to a first branching strategy). We partition  $R(k, y)$  into  $\alpha_k + 1$  subproblems where  $\alpha_k := \lfloor y/a_k \rfloor$  is a natural upper bound for  $x_k$ . The subproblems of  $R(k, y)$  are depicted



**Fig. 2.5** Partition into  $\alpha_k + 1$  subproblems

in Fig. 2.5 where  $R(k, y) \wedge x_k = j$  with  $j \in \{0, \dots, \alpha_k\}$  symbolizes a resulting subproblem

$$R(k+1, y - j \cdot a_k) := R(k, y) \wedge x_k = j.$$

We obtain a first upper bound  $B_0(k, y)$  for  $R(k, y)$  with  $k \in \{1, \dots, m-1\}$  and  $y \in \{1, \dots, b\}$  by solving the corresponding *continuous relaxation*

$$B_0(k, y) := \left\lfloor \max \left\{ \sum_{i=k}^m c_i x_i : \sum_{i=k}^m a_i x_i \leq y, x_i \in \mathbb{R}_+, i = k, \dots, m \right\} \right\rfloor$$

which results by dropping the integrality condition, and by rounding down because of the assumed integer input data. It is not hard to show (Exercise 2.6) that

$$r(k, y) \leq B_0(k, y) = \left\lfloor \frac{c_k}{a_k} \cdot y \right\rfloor$$

holds. A somewhat tighter bound results if the integrality of  $x_k$  is additionally exploited. Since we have  $x_k \leq \alpha_k := \lfloor y/a_k \rfloor$  in every feasible solution of  $R(k, y)$ , we obtain for  $k < m$  the bound

$$B_1(k, y) := c_k \alpha_k + \left\lfloor \frac{c_{k+1}}{a_{k+1}} \cdot (y - a_k \alpha_k) \right\rfloor.$$

Clearly, we have  $B_1(k, y) \leq B_0(k, y)$ . For  $k$  with  $1 \leq k \leq m-2$  we possibly obtain an even tighter value if we apply an upper bound proposed by Martello and Toth [8] for the 0/1 knapsack problem. For  $\alpha_{k+1} := \lfloor (y - a_k \alpha_k)/a_{k+1} \rfloor$  we consider the two cases

$$x_{k+1} \leq \alpha_{k+1} \quad \vee \quad x_{k+1} \geq \alpha_{k+1} + 1.$$

Let  $\bar{c} := c_k \alpha_k + c_{k+1} \alpha_{k+1}$  and  $\bar{a} := a_k \alpha_k + a_{k+1} \alpha_{k+1}$ , and let

$$B_2(k, y) := \bar{c} + \left\lfloor (y - \bar{a}) \frac{c_{k+2}}{a_{k+2}} \right\rfloor,$$

$$B_3(k, y) := \begin{cases} \bar{c} + c_{k+1} - \left\lceil (\bar{a} + a_{k+1} - y) \frac{c_k}{a_k} \right\rceil, & \text{if } a_{k+1}(\alpha_{k+1} + 1) \leq y, \\ -\infty, & \text{otherwise.} \end{cases}$$

The symbol  $\lceil q \rceil$  denotes the smallest integer not smaller than  $q \in \mathbb{R}$ , i.e.,  $q$  is rounded up to the nearest integer.

**Proposition 2.1** *The chain of inequalities*

$$r(k, y) \leq \max \{ B_2(k, y), B_3(k, y) \} \leq B_1(k, y) \leq B_0(k, y).$$

holds for the optimal value  $r(k, y)$  of the residual problem  $R(k, y)$ .

*Proof* Because of  $x_k \leq \alpha_k$  in every feasible solution, we have

$$\begin{aligned} r(k, y) &\leq c_k \alpha_k + \lfloor \frac{c_{k+1}}{a_{k+1}} (y - a_k \alpha_k) \rfloor = B_1(k, y) \\ &\leq c_k \alpha_k + \lfloor \frac{c_k}{a_k} (y - a_k \alpha_k) \rfloor = B_0(k, y). \end{aligned}$$

Moreover,

$$B_1(k, y) = c_k \alpha_k + \lfloor \frac{c_{k+1}}{a_{k+1}} (y + a_{k+1} \alpha_{k+1} - \bar{a}) \rfloor \geq \bar{c} + \lfloor \frac{c_{k+2}}{a_{k+2}} (y - \bar{a}) \rfloor = B_2(k, y),$$

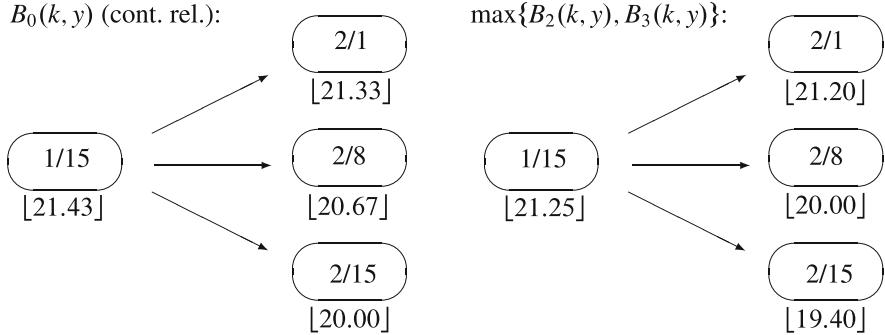
$$\begin{aligned} B_1(k, y) &= c_k \alpha_k + \lfloor \frac{c_{k+1}}{a_{k+1}} (y - a_k \alpha_k) + \frac{c_{k+1}}{a_{k+1}} (\bar{a} + a_{k+1} - y) - \frac{c_{k+1}}{a_{k+1}} (\bar{a} + a_{k+1} - y) \rfloor \\ &= \bar{c} + c_{k+1} + \lfloor -\frac{c_{k+1}}{a_{k+1}} (\bar{a} + a_{k+1} - y) \rfloor = \bar{c} + c_{k+1} - \lceil \frac{c_{k+1}}{a_{k+1}} (\bar{a} + a_{k+1} - y) \rceil \\ &\geq \bar{c} + c_{k+1} - \lceil \frac{c_k}{a_k} (\bar{a} + a_{k+1} - y) \rceil = B_3(k, y) \quad (\text{if } B_3(k, y) > -\infty). \end{aligned}$$

Obviously,  $B_2$  and  $B_3$  are the optimal values of the continuous relaxation of subproblem  $R(k, y) \wedge (x_{k+1} \leq \alpha_{k+1})$  and  $R(k, y) \wedge (x_{k+1} \geq \alpha_{k+1} + 1)$ , respectively. Since every feasible solution of  $R(k, y)$  belongs to one of the both cases, we have  $r(k, y) \leq \max \{ B_2(k, y), B_3(k, y) \}$ . ■

*Example 2.3* We consider once more the knapsack problem of Example 2.1, but now resorted and renumbered according to the inequalities in (2.3):

$$r(1, 15) := \max \{ 10x_1 + 12x_2 + 5x_3 + 6x_4 : 7x_1 + 9x_2 + 4x_3 + 5x_4 \leq 15, x \in \mathbb{Z}_+^4 \}.$$

The application of the upper bound  $B_0$  and, alternatively, of the tighter, combined bound  $\max \{ B_2(k, y), B_3(k, y) \}$ , is depicted in Fig. 2.6. As this (small) example shows, the usage of better bounds can reduce the number of subproblems which have to be considered within the branch-and-bound algorithm. If, for instance, the



**Fig. 2.6** Branching tree: the entries within the ovals correspond to  $k/y$ . The upper bounds are given with respect to the initial problem

### Branch-and-Bound Algorithm

Input:  $c, a, b$

Output:  $v^*, x^*$

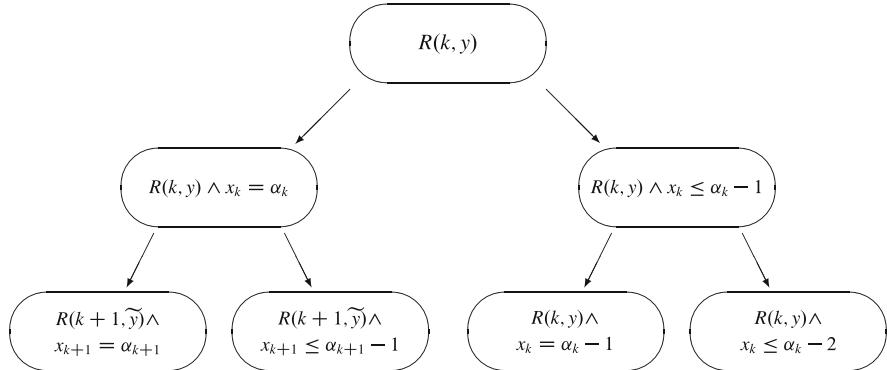
- (1) Set  $x_i := 0$  and  $x_i^* := 0$  for  $i = 1, \dots, m$ ,  $y := b$ .  $v := 0$ ,  $v^* := 0$ ,  $k := 1$ .
- (2) *Bound:* If  $v + B(k, y) \leq v^*$ , then go to (7).
- (3) *Branch:* Set  $x_k := \lfloor y/a_k \rfloor$ ,  $v := v + c_k x_k$ ,  $y := y - a_k x_k$ ,  $k := k + 1$ .
- (4) If  $k = m$ , set  $x_m := \lfloor y/a_m \rfloor$ ,  $v := v + c_m x_m$ ,  $y := y - a_m x_m$ .
- (5) If  $k < m$  and  $y \geq \min\{a_i : i \geq k\}$ , then go to (2).
- (6) If  $v > v^*$ , then set  $v^* := v$ ,  $x_i^* := x_i$  for all  $i = 1, \dots, m$ .
- (7) *Back track:* Set  $v := v - c_m x_m$ ,  $y := y + a_m x_m$ ,  $x_m := 0$ .
- (8) While  $k > 0$  and  $x_k = 0$ , set  $k := k - 1$ . If  $k = 0$ , then stop.
- (9) Set  $x_k := x_k - 1$ ,  $v := v - c_k$ ,  $y := y + a_k$ ,  $k := k + 1$ , and go to (2).

**Fig. 2.7** General Branch-and-Bound algorithm

optimal value 20 is already known (associated with the solution  $x^* = (2, 0, 0, 0)^\top$ ) and all solutions have to be found, then in the first case three and in the second case two subproblems remain to be considered. ■

The application of tighter bounds within a branch-and-bound algorithm can lead, on the one hand, to a reduction of the number of subproblems. On the other hand, the computation of a tighter bound requires, in general, a higher computational effort. For that reason, in any case, one has to check whether the higher computational effort (to compute a tighter single bound) does not lead to an increase of the total run time even if the number of subproblems is reduced. A compromise between the total number of subproblems to be considered and the computational effort per subproblem has to be found.

In the branch-and-bound algorithm, stated in Fig. 2.7, we apply a (general) bounding function  $B(k, y)$  for the residual problem  $R(k, y)$ .



**Fig. 2.8** Alternative concept: partition into two subproblems ( $\tilde{y} = y - \alpha_k a_k$ ,  $\alpha_{k+1} = \lfloor \tilde{y}/a_{k+1} \rfloor$ )

A second opportunity to define subproblems is offered by the *alternative concept*; only two subproblems are defined for  $R(k, y)$ :

$$(R(k, y) \wedge x_k = \alpha_k) \quad \text{and} \quad (R(k, y) \wedge x_k \leq \alpha_k - 1).$$

Note that variable  $x_k$  is that one with the largest relative yield  $c_k/a_k$ . The associated branching schema is depicted in Fig. 2.8. Appropriate upper bounds can be obtained again by means of the corresponding continuous relaxation.

## 2.5 Periodicity and Dominance of Solutions

Within this section we investigate, in particular, statements on the solution of knapsack problems in dependence on the knapsack volume  $b$  while the other input parameters remain unchanged. Without loss of generality, we again assume  $c_i/a_i \geq c_{i+1}/a_{i+1}$  for  $i = 1, \dots, m-1$  and define  $f(b) := \max\{c^\top x : a^\top x \leq b, x \in \mathbb{Z}_+^m\}$ .

**Definition 2.1** If  $a_j \geq k a_i$  and  $c_j \leq k c_i$  for some  $k \in \mathbb{Z}_+$ , then the variable  $x_j$  is *dominated* by  $x_i$  ( $i \neq j$ ) since for any feasible solution  $x$  of (2.1) with  $x_j > 0$  there exists a feasible solution  $\bar{x} = x + x_j(k e^i - e^j)$  with  $\bar{x}_j = 0$  and  $c^\top \bar{x} \leq c^\top x$ .

In a similar way, the idea of *dominance* can be applied to vectors. The vector  $x \in \mathbb{Z}_+^m$  is *dominated* by the vector  $\bar{x}$  (with respect to the knapsack problem) if  $c^\top x \leq c^\top \bar{x}$  and  $a^\top x \geq a^\top \bar{x}$  hold.

Using dominance considerations we can possibly assign the value 0 to some of the variables, and in this way, reduce the size of the problem. In many cases, the concept of dominance can be applied successfully within branch-and-bound algorithms and dynamic programming based approaches, see e.g. Sect. 2.7.

**Theorem 2.2 (Gilmore and Gomory)** *If  $b$  is sufficiently large, then there exists a solution  $x$  of the knapsack problem (2.1) with  $x_1 > 0$ , i.e., there exists a constant  $b^* \geq 0$  such that*

$$f(b) = c_1 + f(b - a_1) \quad \forall b \geq b^*.$$

*Proof* Let  $b^*$  be defined by  $b^* := (a_1 - 1) \sum_{i=2}^m a_i + \min\{a_2, \dots, a_m\}$ .

For some  $b$  with  $b \geq b^*$  let  $x = (x_1, \dots, x_m)^\top$  be a solution of the knapsack problem KP( $b$ ). In case of  $x_1 = 0$ , because of the optimality of  $x$  and the definition of  $b^*$ , there must exist at least one  $k \geq 2$  with  $x_k \geq a_1$ . By means of the transformation  $x'_1 := a_k$ ,  $x'_k := x_k - a_1$ ,  $x'_j := x_j, j \notin \{1, k\}$ , we obtain a feasible solution  $\tilde{x}$  with value  $c^\top \tilde{x} \geq c^\top x$ . Hence,  $\tilde{x}$  is also a solution of the knapsack problem KP( $b$ ) with  $x'_1 > 0$ . ■

**Theorem 2.3** *If  $x \in \mathbb{Z}_+^m$  is a solution of the knapsack problem KP( $b$ ), then  $y \in \mathbb{Z}_+^m$  with  $0 \leq y \leq x$  (i.e.  $y_i \leq x_i$  for all  $i \in I$ ) is a solution of the knapsack problem KP( $\bar{b}$ ) with  $\bar{b} := b - a^\top(x - y)$ .*

*Proof* If we assume that  $y'$  is a better solution of KP( $\bar{b}$ ) than  $y$ , then we get a contradiction to the optimality of  $x$  for KP( $b$ ). ■

The calculation of the minimal  $b^*$  in Theorem 2.2 is difficult, in general. A sufficient condition for the occurrence of periodic solutions is proposed in the following theorem.

**Theorem 2.4** *Let  $A := \max\{a_1, \dots, a_m\}$ . If*

$$f(b) = c_1 + f(b - a_1) \tag{2.4}$$

*for all  $b$  with  $b^* \leq b < b^* + A$ , then equality (2.4) holds for all  $b \geq b^*$ .*

*Proof* Assume that  $f(\bar{b}) = c^\top x$  holds for  $\bar{b} \geq b^* + A$ . Because of the definition of  $A$  there exists  $\tilde{x} \in \mathbb{Z}_+^m$  with  $0 \leq \tilde{x} \leq x$  and  $b^* \leq \bar{b} - a^\top \tilde{x} < b^* + A$ . Thus, we have  $f(\bar{b}) = f(\bar{b} - a^\top \tilde{x}) + c^\top \tilde{x}$ . Since (2.4) is fulfilled for  $b := \bar{b} - a^\top \tilde{x}$ , a solution  $x$  of KP( $\bar{b}$ ) with  $x_1 > 0$  exists. ■

Subsequently we develop a modification of the Longest Path Method which exploits a potential periodicity of intermediate solutions. For that reason, a lexicographic sorting of feasible solutions is realized.

**Definition 2.2** Let  $x, y \in \mathbb{R}^m$ . Vector  $x$  is called *lexicographically larger* than vector  $y$  if  $x_i = y_i$  for  $i = 1, \dots, p-1$ , and  $x_p > y_p$  hold for some  $p \in \{1, \dots, m\}$ .

According to the modeling of a knapsack problem as a longest path problem, a lot of equivalent solutions can occur. For example, a feasible solution  $x_1 = x_2 = 1$  is represented by the two paths  $0 \rightarrow a_1 \rightarrow a_1 + a_2$  and  $0 \rightarrow a_2 \rightarrow a_1 + a_2$ . Within an efficient algorithm, only one of the equivalent paths should be considered. In order

to identify a representative of all optimal paths from vertex 0 to vertex  $y$ , we define

$$\mu(y) := \begin{cases} m + 1, & \text{if } y < \min\{a_i : i \in I\}, \\ \min\{i \in I : f(y) = c_i + f(y - a_i)\}, & \text{otherwise.} \end{cases}$$

Index  $\mu(y)$  represents that variable which has the largest ratio of yield and cost coefficient and which gets a positive value in a solution of  $KP(y)$ .

**Proposition 2.5** *For knapsack problem  $KP(b)$  we identify a solution  $x$  as follows:*

- Set  $x_i := 0$  for  $i = 1, \dots, m$  and  $y := b$ .
- While  $\mu(y) \leq m$ : set  $x_{\mu(y)} := x_{\mu(y)} + 1$ ,  $y := y - a_{\mu(y)}$ .

The obtained vector  $x$  is the lexicographically largest solution of  $KP(b)$ .

*Proof* We assume that  $\tilde{x} \in \mathbb{Z}_+^m$  solves  $KP(b)$  and is lexicographically larger than  $x$ . Then there exists  $p \in I$  with  $\tilde{x}_i = x_i$  for  $i = 1, \dots, p-1$  and  $\tilde{x}_p > x_p$ . Because of

$$f(b) = \sum_{i=1}^p c_i x_i + \sum_{i=p+1}^m c_i x_i = \sum_{i=1}^p c_i x_i + c_p (\tilde{x}_p - x_p) + \sum_{i=p+1}^m c_i \tilde{x}_i$$

we have that  $(0, \dots, 0, x_{p+1}, \dots, x_m)^\top$  and  $(0, \dots, 0, \tilde{x}_p - x_p, \tilde{x}_{p+1}, \dots, \tilde{x}_m)^\top$  are solutions of the knapsack problem  $KP(b')$  with  $b' = b - \sum_{i=1}^p a_i x_i$ . Due to the definition of  $\mu$ , it follows that  $\mu(b) \leq p$  since  $f(b') = f(b' - a_p) + c_p$  because of  $\tilde{x}$ . This is a contradiction to the construction of  $x$ . ■

The basic principle of the *Lexicographic Longest Path Method*, defined in Fig. 2.9, is as follows: Starting at a vertex  $y$ , only those arcs  $(y, y + a_i)$  with  $i \leq \mu(y)$

### Lexicographic Longest Path Method

Input:  $c, a, b$

Output:  $x^*, v(y)$  for  $y = 0, 1, \dots, b$

- (1) Set  $v(y) := 0$  for  $y = 0, \dots, b$ ,  $\mu(0) := m$ , and  $y := 0$ .
- (2) While  $y + \min_{i=1, \dots, m} a_i \leq b$ :
- (3)     For all  $i \in \{1, \dots, \mu(y)\}$  with  $y + a_i \leq b$ :
- (4)         if  $v(y + a_i) = v(y) + c_i$  and  $i < \mu(y + a_i)$ , set  $\mu(y + a_i) := i$ ;
- (5)         if  $v(y + a_i) < v(y) + c_i$ , set
- (6)              $v(y + a_i) := v(y) + c_i$ ,  $\mu(y + a_i) := i$ .
- (7)     Set  $y := \min\{t : v(t) > v(y)\}$ .
- (8) Compute  $v^* := \max_y \{v(y) : b - \min_{i=1, \dots, m} a_i < y \leq b\}$  and  
 $y^* := \min\{y : v(y) = v^*\}$ .
- (9) Set  $x_i^* := 0$  for  $i = 1, \dots, m$  and  $y := y^*$ .
- (10) While  $y > 0$ : set  $i := \mu(y)$ ,  $x_i^* := x_i^* + 1$ ,  $y := y - a_i$ .

**Fig. 2.9** Lexicographic Longest Path Method

are considered whereas any path from vertex 0 via  $y$  to vertex  $y' := y + a_i$  which would result by the arc  $(y, y + a_i)$  with an  $i > \mu(y)$  is not examined. The exchange of the two arcs with vertex difference  $a_i$  and  $a_{\mu(y)}$  leads to a path of the same length.

**Theorem 2.6** *The Lexicographic Longest Path Method computes the lexicographically largest solution of the knapsack problem KP(b).*

*Proof* The proof is done by induction with respect to  $y$  ( $y = 1, \dots, b$ ).

For that reason, we complete the functions  $v$  and  $\mu$  as follows: for  $y \in \{1, \dots, b\}$  for which  $v(y) < v(y - 1)$  after the termination of the algorithm holds, we set  $v(y) := v(y - 1)$  and  $\mu(y) := \mu(y - 1)$  so that  $v$  becomes monotone increasing (in fact, non-decreasing).

Let  $A := \min\{a_1, \dots, a_m\}$ . Then, obviously,  $v(y) = 0$  for  $y \leq A - 1$ .

Furthermore, let  $y \geq A$  such that  $f(y') = v(y')$  for all  $y' < y$ . If  $f(y) = f(y - 1)$ , then vertex  $y$  is not examined in the algorithm with respect to path extensions. Therefore, let  $f(y) > f(y - 1)$ . Moreover, let  $\bar{x} \in \mathbb{Z}_+^m$  denote the lexicographically largest solution of KP(y), i.e.  $c^\top \bar{x} = f(y)$  and  $a^\top \bar{x} = y$ , and let  $j \in I$  be the smallest index with  $x_j > 0$ . Because of Theorem 2.3 we have that  $\tilde{x} := \bar{x} - e^j$  is a solution of KP( $y - a_j$ ) ( $e^j$  denotes the  $j$ th unit vector). Since  $f(y - a_j) = v(y - a_j)$  and since  $\mu(y - a_j) \geq j$ , it follows, due to the algorithm, that  $v(y) \geq v(y - a_j) + c_j = f(y)$ , and therefore,  $f(y) = v(y)$  holds.

The fact that the lexicographically largest solution is provided by the algorithm, is established by Proposition 2.5. ■

The strategy of the Lexicographic Longest Path Method becomes explicitly observable in Exercise 2.13.

## 2.6 Knapsack Problems with Upper Bounds

In many applications the availability of the pieces is restricted. This leads to the problem

$$f(b, u) := \max \left\{ \sum_{i \in I} c_i x_i : \sum_{i \in I} a_i x_i \leq b, x_i \in \mathbb{Z}_+, x_i \leq u_i, i \in I \right\},$$

where  $u = (u_1, \dots, u_m)^\top \in \mathbb{Z}_+^m$  is predefined. As usual, we assume  $c_i > 0$ ,  $0 < a_i \leq b$  for all  $i \in I$  as well as  $c_1/a_1 \geq \dots \geq c_m/a_m$ . Furthermore, without loss of generality, we assume  $0 < u_i \leq b/a_i$  for all  $i \in I$  and  $\sum_{i \in I} a_i u_i > b$ . Appropriate algorithms are preserved by modifying the algorithms for the standard knapsack problem.

In a branch-and-bound algorithm we additionally restrict the number of subproblems which result when variable  $x_i$  is used for branching, by  $u_i$ . In order to ensure  $x_k \leq u_k$  for all  $k \in I$  within a modification of the Algorithm of Gilmore

and Gomory (Sect. 2.2), we take only values of stage  $k - 1$ . For  $k \in I$  and  $y \in \{0, 1, \dots, b\}$  let

$$F(k, y) := \max \left\{ \sum_{i=1}^k c_i x_i : \sum_{i=1}^k a_i x_i \leq y, x_i \in \mathbb{Z}_+, x_i \leq u_i, i \in I \right\}.$$

Then we have for  $k = 1$ ,

$$F(1, y) := c_1 \cdot \min\{u_1, \lfloor y/a_1 \rfloor\}, \quad y = 0, 1, \dots, b,$$

and for  $k = 2, \dots, m$ ,

$$F(k, y) := \max_{0 \leq j \leq \min\{u_k, \lfloor y/a_k \rfloor\}} \{F(k-1, y - ja_k) + jc_k\}, \quad y = 0, 1, \dots, b.$$

During the identification of a corresponding solution we also have to take care that the conditions  $x_i \leq u_i$ ,  $i \in I$ , are met.

In order to obtain a ‘forward-recurrence’ for the knapsack problem with upper bounds, we need to modify the Longest Path Method. For that, we define a directed  $(m+2)$ -partite graph  $G = (V, E)$ . The vertex set  $V$  consists of  $m+2$  disjoint subsets  $V_k$  where  $V_0 := \{0\}$ ,  $V_{m+1} := \{b\}$  and  $V_k \subseteq \{0, 1, \dots, b\}$  for  $k \in I$ . The vertex sets  $V_k$  are defined recursively for  $k = 1, \dots, m$  as follows:

$$V_k := \{y : y = y' + j \cdot a_k, j = 0, \dots, \min\{u_k, \lfloor (b-y')/a_k \rfloor\}, y' \in V_{k-1}\}.$$

Similarly, the arc set  $E$  consists of  $m+1$  subsets, i.e.,  $E = \cup_{k=1}^{m+1} E_k$  with  $E_k \subseteq V_{k-1} \times V_k$  for  $k = 1, \dots, m+1$ . According to the setting of  $V_k$  we define for  $k = 1, \dots, m$ :

$$E_k := \{(y', y) : y' \in V_{k-1}, y = y' + j \cdot a_k, j = 0, \dots, \min\{u_k, \lfloor (b-y')/a_k \rfloor\}\}.$$

Furthermore, let  $E_{m+1} := \{(y, b) : y \in V_m\}$ . We assign the value (length)  $j \cdot c_k$  to the arc  $(y, y + j \cdot a_k) \in E_k$ . Thus, this arc represents the fixing of the  $k$ th variable according to  $x_k := j$ . All arcs in  $E_{m+1}$  get the value 0. Consequently, each path in  $G$  from  $0 \in V_0$  to vertex  $b \in V_{m+1}$  models a feasible solution of the knapsack problem with upper bounds, and vice versa. In order to obtain  $f(b, u)$ , a longest path in  $G$  has to be computed. The proposed approach is summarized in Fig. 2.10. After the termination of the  $k$ th step in (2), we have  $\max\{h(k, y') : y' \leq y\} = F(k, y)$  for all  $y \in \{0, \dots, b\}$ . A more efficient variant of the proposed approach can be found in [7].

**Modified Longest Path Method**Input:  $c, a, b, u$ Output:  $f(b, u)$ 

- (1) For  $y = 0, \dots, b$  set  $h(1, y) := c_1 \cdot \min\{u_1, \lfloor y/a_1 \rfloor\}$ .
- (2) For  $k = 2, \dots, m$ :
- (3)     for  $y = 0, \dots, b$  set  $h(k, y) := 0$ ;
- (4)     for  $y = 0$  and all  $y \in V_{k-1}$  with  $y > 0$  and
- (5)          $h(k-1, y) > \max\{h(k-1, y') : y' \in V_{k-1}, y' < y\}$ :
- (6)         for  $j = 0, \dots, \min\{u_k, \lfloor (b-y)/a_k \rfloor\}$  set
- (7)          $h(k, y + j \cdot a_k) := \max\{h(k, y + j \cdot a_k), h(k-1, y) + j \cdot c_k\}$ .
- (8) Compute  $f(b, u) = \max\{h(m, y) : 0 < y \leq b\}$ .

**Fig. 2.10** Modified Longest Path Method

## 2.7 Sets of Potential Allocation Points

In this section, we consider again the (standard) knapsack problem as in Sect. 2.1:

$$\text{KP}(c, a, b) : \quad \sum_{i \in I} c_i x_i \rightarrow \max \quad \text{s. t.} \quad \sum_{i \in I} a_i x_i \leq b, \quad x_i \in \mathbb{Z}_+ \text{ for } i \in I \quad (2.5)$$

where  $a = (a_1, \dots, a_m)^\top \in \mathbb{Z}_{>}^m$ ,  $c = (c_1, \dots, c_m)^\top \in \mathbb{Z}_{>}^m$ , and  $b \in \mathbb{Z}_{>}$  are given.

If we consider  $a^\top x \leq y$  with a parameter  $y \in \mathbb{R}$ , then we obtain the following optimal value function  $f : \mathbb{R} \rightarrow \mathbb{Z}_+ \cup \{-\infty\}$  related to  $\text{KP}(c, a, b)$ :

$$f(y) := \max \left\{ \sum_{i \in I} c_i x_i : \sum_{i \in I} a_i x_i \leq y, x_i \in \mathbb{Z}_+ \text{ for } i \in I \right\} \quad \text{for all } y \in \mathbb{R} \quad (2.6)$$

where  $f(y) := -\infty$  for each  $y < 0$ . It is well known that  $f$  is piecewise constant and increasing. Its jump discontinuities are nonnegative integers. The set of all these jump discontinuities of  $f$  depends on  $c$  and  $a$  and is a subset of

$$S(a) := \left\{ r = \sum_{i \in I} a_i x_i : x_i \in \mathbb{Z}_+, i \in I \right\}. \quad (2.7)$$

The fact that the optimal value function of  $\text{KP}(c, a, b)$  changes (increases) only at discrete points can be used in branch-and-bound or dynamic programming based approaches to reduce the computational complexity of solving the knapsack problem.

If the knapsack problem (2.5) is used to model a cutting or packing problem, we call  $S(a)$  *set of potential allocation points*. Replacing  $a$  by  $\ell = (\ell_1, \dots, \ell_m)^\top$  and  $b$  by  $L$  we see that  $\text{KP}(c, \ell, L)$  models a one-dimensional cutting problem as

considered in Sect. 1.6. In this case,  $x_i$  represents the number how often piece  $i$  is cut. The set  $S(\ell)$  contains infinitely many elements whereas the points (coordinates) for cutting the raw material are bounded by  $L$ . Therefore, we introduce the finite set

$$S(\ell, L) := \{r \in S(\ell) : r \leq L\}. \quad (2.8)$$

Of course,  $S(\ell, L)$  contains all those jump discontinuities of the optimal value function arising from  $\text{KP}(c, \ell, L)$  which are not larger than  $L$ . Depending on  $c$ , additional points, not being a jump discontinuity, may belong to  $S(\ell, L)$  as well. Thus, the question arises whether one can describe the set of jump discontinuities exactly. This is possible in the important case when  $c = \ell$ . Then the knapsack problem  $\text{KP}(\ell, \ell, L)$  has the optimal value function  $f$  given by

$$f(y) = \max \left\{ \sum_{i \in I} \ell_i x_i : \sum_{i \in I} \ell_i x_i \leq y, x_i \in \mathbb{Z}_+, i \in I \right\} \quad \text{for all } y \in \mathbb{R}$$

and  $S(\ell, L)$  is exactly the set of those jump discontinuities of  $f$  which are not larger than  $L$ .

Let  $x^*$  denote a solution of  $\text{KP}(c, \ell, L)$  with  $\ell^\top x^* < L$ , i.e., there is some waste of raw material. Then, to cut the items according to  $x^*$ , infinitely many possibilities exist to choose a pattern; more precisely, the coordinates of the items of the solution. If the items are placed as left as possible on the raw material, then the number of such patterns is finite, all the waste lies right to the items, and all coordinates of the cut positions belong to  $S(\ell)$ . Such patterns are often called *normalized* or *left-justified* as already mentioned in Sect. 1.3. Note that Herz [6] used the terms *discretization point* for  $r \in S(\ell)$  and *canonical* for left-justified patterns.

For a set  $T \subset \mathbb{Z}_+$  and  $y \in \mathbb{R}_+$  let  $p_T(y)$  and  $s_T(y)$  denote the *predecessor of  $y$  with respect to  $T$*  and the *successor of  $y$  with respect to  $T$* , respectively, i.e.,

$$p_T(y) := \max\{r \in T : r \leq y\} \quad \text{and} \quad s_T(y) := \min\{r \in T : r \geq y\}$$

for all  $y \in [\min\{r : r \in T\}, \max\{r : r \in T\}]$ . In terms of the one-dimensional cutting problem,  $p_{S(\ell)}(y)$  denotes the largest allocation point less than or equal to  $y$ . With other words,  $p_{S(\ell)}(y)$  is the maximum usable length when the raw material has length  $y$ . The term  $s_{S(\ell)}(y)$  denotes the minimum length of a combination of item lengths needed to cover the length  $y$ . Obviously, we have

$$y - \ell_{\min} < p_{S(\ell)}(y) \leq y \quad \text{for all } y \geq \ell_{\min}$$

where  $\ell_{\min} := \min\{\ell_i : i \in I\}$ , and

$$p_{S(\ell)}(y) + p_{S(\ell)}(L - y) \leq p_{S(\ell)}(L) \quad \text{for all } y \in [0, L].$$

**Algorithm BDP**Input:  $c, a, b, T$ Output:  $g$ 

- (1) Set  $g(0) := 0, y := 0.$
- (2) While  $y < p_T(b)$  do
- (3)      $y := s_T(y + 1),$
- (4)      $g(y) := \max_{i \in I} \{c_i + g(p_T(y - a_i)); y \geq a_i\}.$

**Fig. 2.11** The Backward Dynamic Programming (BDP) algorithm**Algorithm FDP**Input:  $c, a, b, T$ Output:  $g$ 

- (1) Set  $g(0) := 0, y := 0.$
- (2) While  $y \leq p_T(b - \min\{a_i : i \in I\})$  do
- (3)     For all  $i \in I$  with  $y + a_i \leq p_T(b)$  do
- (4)          $g(s_T(y + a_i)) := \max\{g(s_T(y + a_i)), c_i + g(y)\},$
- (5)      $\bar{y} := y,$
- (6)     Repeat  $y := s_T(y + 1)$  until  $g(\bar{y}) < g(y).$

**Fig. 2.12** The Forward Dynamic Programming (FDP) algorithm

Based on these definitions, the knapsack problem  $KP(c, a, b)$  can be solved by means of the following backward dynamic programming (BDP) algorithm (Fig. 2.11) which is a modification of the Algorithm of Gilmore and Gomory, Sect. 2.2. If set  $T$ , used in this algorithm, contains at least all jump discontinuities of the optimal value function of  $KP(c, a, b)$ , then Algorithm BDP provides a function  $g : T \rightarrow \mathbb{Z}_+$  by which a solution of  $KP(c, a, b)$  can easily be determined. For example,  $T := S(a, b)$  would do the job. Later on, it will turn out that Algorithm BDP can even successfully be used for solving knapsack problems if  $T$  only contains a certain subset of jump discontinuities.

**Theorem 2.7** *Let the set  $T$  contain at least all jump discontinuities of the optimal value function  $f$  of  $KP(c, a, b)$ . If Algorithm BDP is used for determining  $g : T \rightarrow \mathbb{Z}_+$ , then we have*

$$f(y) = g(p_T(y)) \quad \text{for all } y \in [0, b].$$

This well-known result can also be obtained for the following forward dynamic programming (FDP) algorithm (Fig. 2.12) which is an adaptation of the Longest Path Method, Sect. 2.3.

Note that  $T = S(a, b)$  implies  $s_T(y + a_i) = y + a_i \in T$  for all  $i \in I$  and all  $y \in T$  with  $y + a_i \leq b$ . Thus,  $s_T(y + a_i)$  can be replaced by  $y + a_i$  in step (4) of Algorithm FDP. Moreover, because of the repeat-loop in step (6), some updates in steps (3) and (4) can probably be saved compared to step (4) of Algorithm BDP.

The worst-case complexity of both algorithms, BDP and FDP, is  $O(b + m|T|)$  since  $y$  is increased at most  $b$  times, and at most  $m$  comparisons are done in the max-terms for each element of  $T$ . Thus, both are pseudo-polynomial algorithms.

In order to determine a *reduced set of potential allocation points* that is sufficient to obtain an optimal solution of  $KP(c, a, b)$  by Algorithm BDP or FDP, we will apply some dominance considerations (cf. [11, 12]). For that purpose, let  $b > \max_{i \in I} a_i$  be satisfied. In view of the *separability* of the optimal value function  $f$ , we have

$$f(b) = \max_{0 < y \leq b/2} \{f(y) + f(b - y)\}.$$

Since  $f$  is piecewise constant with jump discontinuities in  $S(a)$  it further follows that

$$\begin{aligned} f(b) &= \max_{0 < y \leq b/2} \{f(p_{S(a)}(y)) + f(p_{S(a)}(b - y))\} \\ &= \max_{r \in S(a), 0 < r \leq b/2} \{f(r) + f(p_{S(a)}(b - r))\}. \end{aligned} \quad (2.9)$$

By  $p_{S(a)}(r) \leq p_{S(a)}(b - p_{S(a)}(b - r))$ , we obtain

$$f(b) = \max_{r \in S(a), 0 < r \leq b/2} \{f(p_{S(a)}(b - p_{S(a)}(b - r))) + f(p_{S(a)}(b - r))\}.$$

This formula motivates the definition of the *reduced set of potential allocation points* by

$$S^{red}(a, b) := \{p_{S(a)}(b - r) : r \in S(a, b)\}.$$

Consequently, we have

$$f(b) = \max_{r \in T, 0 < r \leq b/2} \{f(r) + f(p_T(b - r))\} \quad \text{with} \quad T := S^{red}(a, b). \quad (2.10)$$

**Theorem 2.8** *Let  $f$  denote the optimal value function of  $KP(c, a, b)$ . If Algorithm BDP (or Algorithm FDP) with  $T = S^{red}(a, b)$  is used to determine  $g : T \rightarrow \mathbb{Z}_+$ , then*

$$f(r) = g(r) \quad \text{for all } r \in T \quad \text{and} \quad f(y) \geq g(p_T(y)) \quad \text{for all } y \in [0, b]$$

hold.

**Table 2.3** Potential allocation points of Example 2.4

| $y$             | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8 | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|-----------------|---|---|---|---|---|---|---|----|---|----|----|----|----|----|----|----|
| $S(a, b)$       | * |   |   |   | * |   |   | *  | * | *  |    | *  | *  | *  | *  | *  |
| $S^{red}(a, b)$ | * |   |   |   | * |   |   | *  | * |    |    | *  |    |    |    | *  |
| $f(y)$          | 0 |   |   |   | 5 |   |   | 10 |   | 12 |    | 15 |    | 17 | 20 |    |

Due to  $S^{red}(a, b) \subseteq S(a, b)$ , recursion (2.10) might be less expensive than that one in (2.9). Moreover, dependent on the instance, significant savings are possible if Algorithms BDP or FDP are applied for the solution of  $KP(c, a, b)$  with  $T = S^{red}(a, b)$ . Note that using  $S^{red}(a, b)$  instead of  $S(a, b)$  is an application of the Nicholson principle [10]. Since  $p_{S(a)}(y) \geq p_{S^{red}(a,b)}(y)$  and  $s_{S(a)}(y) \leq s_{S^{red}(a,b)}(y)$  for all  $y \in [0, b]$ , the application of Algorithm FDP with  $T = S^{red}(a, b)$  does not longer provide left-justified patterns, in general.

*Example 2.4* Let us consider the instance of the knapsack problem  $KP(c, a, b)$  with  $c := (12, 10, 5)^\top$ ,  $a := (9, 7, 4)^\top$ , and  $b := 15$ . In Table 2.3 the elements of sets  $S(a, b)$  and  $S^{red}(a, b)$  are marked by  $\star$ . Additionally, the optimal value function  $f$  is tabulated at their jump discontinuities. ■

In Example 2.4, we have  $|S^{red}(a, b)| < |S(a, b)| < b$ . Moreover, we can see that the set of jump discontinuities is not a subset of  $S^{red}(a, b)$ , in general. Obviously, the cardinalities of  $S(a, b)$  and  $S^{red}(a, b)$  strongly depend on the input data. Nevertheless, there is a high potential to save memory and computation time by using  $S^{red}(a, b)$  instead of  $S(a, b)$ . In particular, the cardinality does not change if the unit of measure is changed, for instance from cm to mm.

Investigations on how to compute  $S(a, b)$  efficiently can be found in [3]. The computational effort for determining  $S(a, b)$  and  $S^{red}(a, b)$  is bounded above by  $O(mb)$ . More precisely, it is bounded by  $O(b + m|S(a, b)|)$  due to the application of Algorithm FDP for  $KP(a, a, b)$ .

## 2.8 Exercises

**Exercise 2.1** Consider an instance of the knapsack problem with rational input data which do not fulfill the conditions  $c_i > 0$ ,  $0 < a_i \leq b$ ,  $i \in I$ ,  $b > 0$ ,  $a_i \neq a_j$  for  $i \neq j$ .

- (a) Show that it can be transformed to an instance of the standard knapsack problem, and that statements of its solvability can be drawn.
- (b) Show by means of an example that the assumption that all input data have to be rational, is important with respect to the solvability.

**Exercise 2.2** Formulate a strategy (an algorithm) to compute all solutions and apply it to Example 2.1.

**Exercise 2.3** In difference to the Algorithm of Gilmore and Gomory where variable  $y$  takes all integer values from 0 to  $b$ , in the Longest Path Method the updating rule  $y := \min\{t : f(t) > f(y)\}$  is applied. Show the correctness of this practice.

**Exercise 2.4** Solve the knapsack problem

$$\max\{5x_1 + 9x_2 + 11x_3 + 8x_4 : 4x_1 + 7x_2 + 9x_3 + 6x_4 \leq 20, x_i \in \mathbb{Z}_+, i = 1, \dots, 4\}$$

- (a) with the Algorithm of Gilmore and Gomory,
- (b) with the Longest Path Method,
- (c) with a branch-and-bound algorithm,
- (d) with the BDP algorithm,
- (e) with the FDP algorithm.

**Exercise 2.5** How does the computational effort change if the input parameters  $a_i$ ,  $i \in I$ , and  $b$  are multiplied with an integer factor  $k$  larger than 1? Compare the Algorithm of Gilmore and Gomory with the Longest Path Method.

**Exercise 2.6** Show that, if condition (2.3) is fulfilled, then  $B_0(k, y) = \lfloor c_k \cdot y/a_k \rfloor$  holds for all  $k \leq m$  and  $y \geq 0$ .

**Exercise 2.7** Which upper bound results by the continuous relaxation if the additional constraint  $x_k \leq \bar{x}_k$  has to be regarded for problem  $R(k, y)$ ? Here  $\bar{x}_k \in \mathbb{Z}_+$  denotes a bound for  $x_k$  with  $\bar{x}_k \leq y/a_k$ .

**Exercise 2.8** Let  $x \in \mathbb{Z}_+^m$  denote the lexicographically largest solution of knapsack problem KP( $b$ ). Show that  $\mu(b) \leq \mu(b - a^\top \tilde{x})$  holds for each  $\tilde{x} \in \mathbb{Z}_+^m$  with  $0 \leq \tilde{x} \leq x$ .

**Exercise 2.9** For some  $k \in I$  let  $b \geq a_k u_k$  and  $f(b) > f(b - a_k u_k) + c_k u_k$  be satisfied. Show that then  $x_k < u_k$  holds in every solution  $x$  of the knapsack problem KP( $b$ ).

**Exercise 2.10** Show that, if a sorting according to  $c_i/a_i \geq c_{i+1}/a_{i+1}$  for  $i = 1, \dots, m-1$  is present, then

$$\sum_{i=1}^k c_i u_i + \frac{c_{k+1}}{a_{k+1}}(b - \sum_{i=1}^k a_i u_i)$$

is the optimal value of the continuous relaxation associated to the knapsack problem KP( $b$ ) with upper bounds  $u_i$ ,  $i \in I$ , where  $k$  is determined by  $\sum_{i=1}^k a_i u_i \leq b < \sum_{i=1}^{k+1} a_i u_i$ .

**Exercise 2.11** Show that  $p_{S(\ell)}(L - x) + p_{S(\ell)}(L - p_{S(\ell)}(L - x)) \leq L$  holds for all  $x \in [0, L]$ .

**Exercise 2.12** Show that  $p_{S(\ell)}(x) \leq p_{S(\ell)}(L - p_{S(\ell)}(L - x))$  holds for all  $x \in [0, L]$ .

**Exercise 2.13** Solve the knapsack problem

$$\max\{10x_1 + 12x_2 + 5x_3 : 7x_1 + 9x_2 + 4x_3 \leq 30, x_i \in \mathbb{Z}_+, i = 1, 2, 3\}$$

with the Modified Longest Path Method.

## 2.9 Solutions

**To Exercise 2.1** (a) We consider knapsack problems of form

$$f(b) := \max\{c^\top x : a^\top x \leq b, x \in \mathbb{Z}_+^m\}$$

and we do not assume that the conditions  $c_i > 0, 0 < a_i \leq b, a_i \neq a_j$  for  $i \neq j, i, j \in I = \{1, \dots, m\}$ , are fulfilled and that all input data are integers. Since, by assumption, all input parameters are rational, we can derive the integrality of all input data by multiplying with the least common denominator. Therefore, we suppose in the following that  $c_i, a_i \in \mathbb{Z}$  for all  $i \in I$  as well as  $b \in \mathbb{Z}$  hold.

First, let  $a_i = 0$  for some  $i$  and  $c_i \neq 0$ . If  $c_i > 0$  and if feasible solutions exist, then the objective function is unbounded above so that KP( $b$ ) has no solution. In case of  $c_i < 0$ , then  $x_i = 0$  holds in each solution. Furthermore, if  $c_i < 0$  and  $a_i > 0$ , then again  $x_i = 0$  holds in each solution. In case of  $c_i > 0$  and  $a_i < 0$  for some  $i \in I$  then again, the objective function is unbounded above so that KP( $b$ ) has no solution.

Let  $I^+ := \{i \in I : c_i \geq 0, a_i > 0\}$  and  $I^- := \{i \in I : c_i < 0, a_i < 0\}$ . We assume  $I^+ \neq \emptyset$  and  $I^- \neq \emptyset$ . The problem (D),

$$(D) \quad w = \min b \cdot u \quad \text{s.t.} \quad a_i u \geq c_i, i \in I, u \in \mathbb{R}_+,$$

is the dual problem of the continuous relaxation of KP( $b$ ) which we denote by (P). Problem (D) is solvable if and only if (P) is solvable.

Because of  $I^+ \neq \emptyset$  we obtain, on the one hand, the condition  $u \geq \max\{c_i/a_i : i \in I^+\}$ , and on the other hand, because of  $I^- \neq \emptyset$ , we have  $u \leq \min\{c_i/a_i : i \in I^-\}$ . Hence, problem KP( $b$ ) is solvable if and only if

$$\max_{i \in I^+} \frac{c_i}{a_i} \leq \min_{i \in I^-} \frac{c_i}{a_i}.$$

In the case where KP( $b$ ) is solvable, we can find upper bounds for all  $x_i$  for  $i \in I^-$  without loosing any solution, e.g.,  $u := \min\{a_i : i \in I^-\}$ . Using the substitution  $\tilde{x}_i := u - x_i, i \in I^-$ , we obtain an equivalent knapsack problem with positive input data. (For example, consider  $z = \max -3x_1 + 4x_2$  s.t.  $-3x_1 + 5x_2 \leq 2, x_i \in \mathbb{Z}_+$ .) Note that the case  $c_i a_i < 0$  can be examined in the same way.

### Computation of all solutions

Input:  $c, a, b, F(m, y)$  for  $y = 0, 1, \dots, b$

Output:  $x^k, k = 1, 2, \dots$

- (1)  $y := b, x_i := 0, i = 1, \dots, m, k := 0, i := 1.$
- (2) While  $i \leq m$  and  $F(m, y) > 0$  set:
  - (3) if  $F(m, y - a_i) + c_i = F(m, y)$ , then  $x_i := x_i + 1, y := y - a_i,$
  - (4) otherwise  $i := i + 1.$
- (5) If  $i \leq m$  and  $F(m, y) = 0$  set
- (6)  $k := k + 1, x_i^k := x_i, i = 1, \dots, m.$  ( $k$ -th solution)
- (7) Set  $y := y + a_m x_m, x_m := 0.$
- (8) If  $x_i = 0, i = 1, \dots, m$ , then stop.
- (9) Set  $i := \max\{j: x_j > 0\}, x_i := x_i - 1, y := y + a_i, i := i + 1.$
- (10) Go to (2).

**Fig. 2.13** Algorithm to compute all solutions

(b) The instance

$$z = \max \pi x_1 - x_2 \quad \text{s. t.} \quad \pi x_1 - x_2 \leq 1, x_1, x_2 \in \mathbb{Z}_+,$$

which contains irrational input data, has no solution. On the one hand, the only integer point with  $z = 1$  is  $(0, -1)$ , but it is infeasible. On the other hand, it is possible to find feasible solutions arbitrarily close to the upper bound 1.

**To Exercise 2.2** In order to find *all* solutions we have to construct a search tree. We obtain all solutions in lexicographically decreasing order. To keep it simple, let  $F(m, y) := -\infty$  if  $y < 0$ .

An algorithm to obtain all solutions is given in Fig. 2.13. We obtain the two solutions  $x^1 = (2, 1, 0, 0)^\top$  and  $x^2 = (0, 2, 0, 0)^\top$  for Example 2.1.

**To Exercise 2.3** Suppose that  $f(y) = f(y + 1)$  holds for  $y \in V, y < b$ .

Due to the definition of graph  $G$ , there exists a path from vertex  $y + 1$  to vertex  $b$  with the same length like a path from  $y$  to  $b - 1$ . Hence, there cannot exist any path from  $y + 1$  to  $b$  which has a larger value in comparison to paths from  $y$  to  $b - 1$ . Therefore, the examination of vertex  $y + 1$  with respect to path extension is not necessary.

**To Exercise 2.4** The computation according to the Algorithm of Gilmore and Gomory as well as of the Longest Path Method is depicted in Tables 2.4 and 2.5. In Table 2.5, symbol  $y$  marks the current iteration. Obviously, only one vector of length  $b$  is needed for the calculations.

**Table 2.4** Algorithm of Gilmore and Gomory: optimal values  $F(k, y)$  to Exercise 2.4

| y    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|------|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| v(y) | 0 |   |   | 5 |   |   |   | 10 |    |    |    | 15 |    |    |    | 20 |    |    |    |    | 25 |
| v(y) |   |   |   |   |   |   |   | 9  |    |    |    | 14 |    |    | 18 | 19 |    |    | 23 | 24 |    |
| v(y) |   |   |   |   |   |   |   |    | 11 |    |    |    | 16 |    |    |    |    | 21 |    |    |    |
| v(y) |   |   |   |   |   |   |   |    |    | 13 |    |    | 16 | 17 |    |    | 21 | 22 | 24 | 25 | 26 |
| v(y) |   |   |   |   |   |   |   |    |    |    | 13 | 14 | 16 | 17 | 18 | 19 | 21 | 22 | 24 | 25 | 26 |

**Table 2.5** Longest Path Method applied to Exercise 2.4

| y    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8 | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |  |
|------|---|---|---|---|---|---|---|----|---|----|----|----|----|----|----|----|----|----|----|----|----|--|
| v(y) | y |   |   |   | 5 |   | 8 | 9  |   | 11 |    |    |    |    |    |    |    |    |    |    |    |  |
| v(y) |   |   | y |   |   |   |   | 10 |   | 13 | 14 |    | 16 |    |    |    |    |    |    |    |    |  |
| v(y) |   |   |   | y |   |   |   |    |   |    |    | 16 | 17 |    | 19 |    |    |    |    |    |    |  |
| v(y) |   |   |   |   | y |   |   |    |   |    |    |    |    | 18 |    | 20 |    |    |    |    |    |  |
| v(y) |   |   |   |   |   | y |   |    |   |    |    |    |    |    |    | 21 |    |    |    |    |    |  |
| v(y) |   |   |   |   |   |   | y |    |   |    |    |    |    |    |    |    | 22 |    |    |    |    |  |
| v(y) |   |   |   |   |   |   |   | y  |   |    |    |    |    |    |    | 21 | 22 |    | 24 |    |    |  |
| v(y) |   |   |   |   |   |   |   |    | y |    |    |    |    |    |    |    | 23 |    | 25 |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   | y  |    |    |    |    |    |    | 24 | 25 |    |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   |    | y  |    |    |    |    |    |    |    | 26 |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   |    |    | y  |    |    |    |    |    |    |    |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   |    |    |    | y  |    |    |    |    |    |    |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   |    |    |    |    | y  |    |    |    |    |    |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   |    |    |    |    |    | y  |    |    |    |    |    |    |  |
| v(y) |   |   |   |   |   |   |   |    |   |    |    |    |    |    |    |    |    |    |    |    |    |  |

Branch-and-Bound: Resorting and renumbering yield the problem

$$z = \max\{8a_1 + 9a_2 + 5a_3 + 11a_4 : 6a_1 + 7a_2 + 4a_3 + 9a_4 \leq 20, a_i \in \mathbb{Z}_+ \forall i\}.$$

We obtain  $z = 24$  for the first subproblem which is defined by  $a_1 = 3$ ,  $z = 25$  for the subproblem with  $a_1 = 2$  and  $a_2 = 1$ , and  $z = 26$  for the subproblem with  $a_1 = 2$ ,  $a_2 = 0$ , and  $a_3 = 2$ .

Because of  $8 + \frac{9}{7} \cdot 14 = 26$ , the optimal value is equal to 26.

**To Exercise 2.5** The computational effort of the Algorithm of Gilmore and Gomory increases (in principle) by the factor  $k$  since now  $y = 1, \dots, kb$  values have to be calculated.

The computational effort of the Longest Path Method does not increase in principle since the number of jump discontinuities  $y$  of  $f$  remains unchanged. In fact, the jump discontinuities  $y$  are transferred to  $ky$ .

**To Exercise 2.6** Let  $\sum_{i=k}^m a_i x_i \leq y$  and  $x_i \in \mathbb{Z}_+$  for  $i = k, \dots, m$ . Then

$$\sum_{i=k}^m c_i x_i = \sum_{i=k}^m \frac{c_i}{a_i} a_i x_i \leq \frac{c_k}{a_k} \sum_{i=k}^m a_i x_i \leq \frac{c_k}{a_k} y.$$

Since  $\sum_{i=k}^m c_i x_i$  is an integer, the statement follows.

**To Exercise 2.7** We obtain  $B_1(k, y)$ .

**To Exercise 2.8** The vector  $x - \tilde{x}$  is a solution of  $\text{KP}(b - a^\top \tilde{x})$ . Because of  $0 \leq x_i - \tilde{x}_i \leq x_i$  for all  $i$ , we have  $\mu(b) = \min\{i \in I : f(b) = c_i + f(b - a_i)\} \leq \min\{i \in I : f(b - a^\top \tilde{x}) = c_i + f(b - a^\top \tilde{x} - a_i)\} = \mu(b - a^\top \tilde{x})$ .

**To Exercise 2.9** Let  $x$  be a solution of  $\text{KP}(b, u)$  with  $x_k \geq u_k$ . Then  $\tilde{x} := x - u_k e^k$  is a solution of  $\text{KP}(b - a_k u_k, u - u_k e^k)$  since otherwise a contradiction to the optimality of  $x$  results. Contrariwise, we have  $f(b) = f(b - a_k u_k) + c_k u_k$  which contradicts to the assumption. Consequently,  $x_k < u_k$  has to hold.

**To Exercise 2.10** Let  $\bar{x} = (\alpha_1, \dots, \alpha_m)^\top$  be defined by  $\alpha_i = u_i$ ,  $i = 1, \dots, k$ ,  $\alpha_{k+1} = (b - \sum_{i=1}^k a_i u_i)/a_{k+1}$  and  $\alpha_i = 0$  for  $i > k+1$ . Then  $\bar{x}$  is a feasible solution of the continuous relaxation of  $\text{KP}(b, u)$ . It remains to show,  $\bar{x}$  is a solution. Let  $x$  denote a solution of the continuous relaxation of  $\text{KP}(b, u)$  with  $x_i < \alpha_i$  for some  $i \leq k$ . Therefore,  $c^\top x \geq c^\top \bar{x}$ . Because of the optimality of  $x$  we have  $a^\top x = b$  (since  $c_i > 0$  for all  $i$ ) and for this reason there exists  $j \geq k+1$  with  $x_j > \alpha_j$ .

Moreover, let  $\tilde{x}$  be defined by  $\tilde{x}_j := x_j - \varepsilon$ ,  $\tilde{x}_i := x_i + \frac{a_i}{a_j} \varepsilon$ ,  $\tilde{x}_r := x_r$  for  $r \neq i, j$  with  $\varepsilon := \min\{x_j, (u_i - x_i) \frac{a_i}{a_j}\} > 0$ . Obviously,  $\tilde{x}$  is feasible since  $\tilde{x}_i \leq x_i + \frac{a_i}{a_j} (u_i - x_i) \frac{a_i}{a_j} = u_i$ ,  $a^\top \tilde{x} = a^\top x + a_i \frac{a_i}{a_j} \varepsilon - a_j \varepsilon = a^\top x \leq b$ ,  $\tilde{x}_j \geq x_j - \varepsilon = 0$  and  $0 \leq \tilde{x} \leq u$ . Furthermore, we have  $c^\top \tilde{x} \geq c^\top \bar{x} = c^\top x + c_i \frac{a_i}{a_j} \varepsilon - c_j \varepsilon = c^\top x + a_j \left(\frac{c_i}{a_i} - \frac{c_j}{a_j}\right) \varepsilon \geq c^\top x$ , thus  $c^\top x = c^\top \tilde{x}$ . Therefore,  $\tilde{x}$  is a solution of the continuous relaxation of  $\text{KP}(b, u)$ . By this construction we obtain  $\tilde{x} = \bar{x}$ .

**To Exercise 2.11** Because of  $p_{S(\ell)}(x) \leq x$  we have

$$p_{S(\ell)}(L-x) + p_{S(\ell)}(L - p_{S(\ell)}(L-x)) \leq p_{S(\ell)}(L-x) + L - p_{S(\ell)}(L-x) = L \text{ for } x \in [0, L].$$

**To Exercise 2.12** Since  $p_{S(\ell)}(\cdot)$  is monotone increasing we have

$$p_{S(\ell)}(x) = p_{S(\ell)}(L - (L-x)) \leq p_{S(\ell)}(L - p_{S(\ell)}(L-x)) \text{ for } x \in [0, L].$$

**To Exercise 2.13** The computation according to the Lexicographic Longest Path Method is depicted in Table 2.6.

**Table 2.6** Lexicographic Longest Path Method applied to Exercise 2.13

## References

1. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms and Applications* (Prentice Hall, Upper Saddle River, 1993)
2. K.-H. Borgwardt, *Optimierung, Operations Research, Spieltheorie* (Birkhäuser, Basel, 2001)
3. G.F. Cintra, F.K. Miyazawa, Y. Wakabayashi, E.C. Xavier, Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *Eur. J. Oper. Res.* **191**, 61–85 (2008)
4. M.R. Garey, D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
5. P.C. Gilmore, R.E. Gomory, The theory and computation of knapsack functions. *Oper. Res.* **14**, 1045–1075 (1966)
6. J.C. Herz, Recursive computational procedure for two-dimensional stock cutting. *IBM J. Res. Dev.* **16**, 462–469 (1972)
7. H. Kellerer, U. Pferschy, D. Pisinger, *Knapsack Problems* (Springer, Berlin/Heidelberg, 2004)
8. S. Martello, P. Toth, *Knapsack Problems* (Wiley, Chichester, 1990)
9. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988)
10. T.A.J. Nicholson, Finding the shortest route between two points in a network. *Comput. J.* **9**, 275–280 (1966)
11. G. Scheithauer, *Zuschnitt- und Packungsoptimierung* (Vieweg + Teubner, Wiesbaden, 2008)
12. J. Terno, R. Lindemann, G. Scheithauer, *Zuschnittprobleme und ihre praktische Lösung* (Verlag Harri Deutsch, Thun und Frankfurt/Main, 1987)

# Chapter 3

## One-Dimensional Bin Packing

The *one-dimensional bin packing problem* (1BPP) is one of the most famous problems in combinatorial optimization. The 1BPP is closely related to the *one-dimensional cutting stock problem* (1CSP) in terms of modeling, however, both problems possess their particularities, especially with respect to solution methods. Therefore, we will consider the 1CSP in another chapter to address the problem specific issues more properly.

The BPP is known to be NP-hard [16]. For that reason, we will consider, besides exact solution approaches, lower and upper bounds for the optimal value as well as reduction methods and some extensions.

### 3.1 Problem Statement and Modeling

The 1BPP can be defined as follows. Let  $n$  *items* (pieces), each having an integer *weight*  $w_i, i \in I := \{1, \dots, n\}$ , and an unlimited number of identical *bins* of integer *capacity*  $C$  be given. Then the objective is to pack all items into the minimal number of bins so that the total weight packed in any bin does not exceed the capacity. Obviously, we can assume, without loss of generality, that  $0 < w_i \leq C$  holds for all  $i \in I$ .

There exists a number of equivalent formulations. For instance, given a set  $S$  of  $n$  positive integers, find a partition of  $S$  of minimal cardinality such that the total sum of numbers in each subset of the partition is not larger than a given value. Frequently, a normalized definition of the 1BPP is considered, too. There, the capacity is set to 1 and, hence, the weights are rational numbers in  $(0, 1]$ .

For convenience, we denote an instance of the one-dimensional BPP by  $E = (n, w, C)$  with  $w = (w_1, \dots, w_n)$ . Moreover, a corresponding *normalized* instance  $E' = (n, w', 1)$  results by setting  $w'_i := w_i/C$  for all  $i \in I$ .

Let  $\bar{z} = \bar{z}(E)$  be any upper bound of the optimal value of 1BPP instance  $E$ , for example, obtained by a heuristic. That means, not more than  $\bar{z}$  bins are needed to pack all items without exceeding the bin capacity. Let  $J := \{1, \dots, \bar{z}\}$  be an appropriate index set. To model the 1BPP as an ILP, we introduce two types of binary variables:

$$y_j = \begin{cases} 1, & \text{if bin } j \text{ is used in the solution,} \\ 0, & \text{otherwise,} \end{cases} \quad j \in J,$$

$$x_{ij} = \begin{cases} 1, & \text{if item } i \text{ is packed into bin } j, \\ 0, & \text{otherwise,} \end{cases} \quad i \in I, j \in J.$$

Then, an ILP model of the 1BPP is as follows:

### ILP model of the 1BPP

$$\zeta^{BPP} = \zeta^{BPP}(E) := \min \sum_{j \in J} y_j \quad \text{s.t.} \quad (3.1a)$$

$$\sum_{i \in I} w_i x_{ij} \leq C y_j, \quad j \in J, \quad (3.1b)$$

$$\sum_{j \in J} x_{ij} = 1, \quad i \in I, \quad (3.1c)$$

$$x_{ij}, y_j \in \mathbb{B}, \quad i \in I, j \in J. \quad (3.1d)$$

According to (3.1a), we aim to minimize the number of used bins. Constraints (3.1b) impose that the capacity of every used bin is not exceeded, and constraints (3.1c) ensure that all items are packed. Obviously, since the number of binary variables depends on the number  $n$  of items and the upper bound  $\bar{z}$ , solving the ILP model can become very costly.

Note that model (3.1) possesses a high degree of *symmetry* since at least  $\zeta^{BPP}!$  equivalent solutions exist (resulting by permuting the pattern indices). We will address this issue below in Sect. 4.8.2.

Another, pattern-oriented ILP model is due to Gilmore and Gomory proposed in [17] for the one-dimensional cutting stock problem. We will go into more details within the respective chapter.

According to constraint (3.1b), a pattern  $x \in \mathbb{B}^n$  is feasible if  $\sum_{i \in I} w_i x_i \leq C$  holds. Let  $J$  be an index set of the set of all feasible patterns  $x^j \in \mathbb{B}^n, j \in J$ . We assign a *decision variable*  $y_j \in \mathbb{B}$  to pattern  $x^j$  to indicate whether pattern  $x^j$  is used

or not. Then, the *Gilmore/Gomory model* is given by:

### Gilmore/Gomory model of the BPP

$$z^{BPP} = \min \sum_{j \in J} y_j \quad \text{s.t.} \quad (3.2a)$$

$$\sum_{j \in J} x_{ij} y_j = 1, \quad i \in I, \quad (3.2b)$$

$$y_j \in \mathbb{B}, \quad j \in J. \quad (3.2c)$$

Objective function (3.2a) counts the total number of bins needed to pack all items, and restrictions (3.2b) ensure that each item is packed. Interestingly, model (3.2) is also a correct model for two- and three-dimensional bin packing problems provided that index set  $J$  represents the respective set of all feasible patterns.

Clearly, model (3.2) is a correct model of the BPP, but it possesses some weakness because of the, in general, exponential number of possible patterns (see Exercise 4.3). Nevertheless, in case of cutting stock problems, the respective Gilmore/Gomory model has high importance, as we will see in Chap. 4.

## 3.2 Lower Bounds

Since the BPP belongs to the class of NP-hard optimization problems, frequently only heuristic solutions are available for instances of medium or large size. To evaluate their quality (performance), lower bounds are meaningful. Moreover, for applying branch-and-bound approaches to solve 1BPP instances (of small or medium size) lower bounds are essential. Due to both aspects, a large number of lower bounds has been proposed in literature. We address some of them.

### 3.2.1 Natural Bounds

We consider an instance  $E = (n, w, C)$  of the 1BPP, and let  $I := \{1, \dots, n\}$ . Clearly, the total weight of all items determines a minimum total capacity of bins, that means, at least

$$lb_c = lb_c(E) := \left\lceil \frac{1}{C} \sum_{i \in I} w_i \right\rceil$$

bins are necessary to pack all items. The lower bound  $lb_c$  is (mostly) named the *continuous bound* of instance  $E$ .

**Proposition 3.1** *For any instance  $E = (n, w, C)$  of the 1BPP with integer input data, we have*

$$lb_c(E) \leq z^{BPP}(E) \leq 2 \cdot lb_c(E)$$

where  $z^{BPP}(E)$  denotes the optimal value of instance  $E$ .

*Proof* Let us consider model (3.1) and let  $y_j^*, x_{ij}^*$ ,  $i \in I, j \in J^* := \{j \in J : y_j^* = 1\}$ , represent an optimal solution of  $E$ , that means,  $z^{BPP} = \sum_{j \in J^*} y_j^* = |J^*|$ . Then, the total weight in all, but possibly one, used bins is larger than  $C/2$  since otherwise, a contradiction to the optimality of  $y^*$  would arise. Therefore, we have

$$\sum_{i \in I} w_i = \sum_{j \in J^*} \sum_{i \in I} w_i x_{ij}^* > \frac{C}{2} \left( \sum_{j \in J^*} y_j^* - 1 \right) = \frac{C}{2} (z^{BPP} - 1).$$

Hence,

$$z^{BPP} < \frac{2}{C} \sum_{i \in I} w_i + 1.$$

Since  $z^{BPP} \in \mathbb{Z}$ , we obtain

$$z^{BPP} \leq \left\lceil \frac{2}{C} \sum_{i \in I} w_i \right\rceil \leq 2lb_c.$$

In case of only smaller item weights, we can obtain a tighter interval of  $z^{BPP}$ . ■

**Lemma 3.2** *Let  $E = (n, w, C)$  be an instance of the 1BPP with  $\sum_{i \in I} w_i > C$ . Furthermore, let  $q \geq 2$ ,  $q \in \mathbb{N}$ . If  $w_i \leq C/q$  for all  $i \in I$ , then there exists a subset  $I^* \subset I$  with*

$$\frac{q}{q+1}C < \sum_{i \in I^*} w_i \leq C.$$

*Proof* Without loss of generality, we assume  $w_i \geq w_{i+1}$  for all  $i \in I$ . Let  $\Delta_0 := C$ . We define

$$x_i := \min\{1, \lfloor \Delta_{i-1}/w_i \rfloor\}, \quad \Delta_i := \Delta_{i-1} - w_i x_i$$

for all  $i = 1, \dots, n$ . Then, let  $I^* := \{i \in I : x_i = 1\}$ . That means, we assign items to  $I^*$  as long as possible according to a greedy strategy. Obviously, we have  $\sum_{i \in I} x_i \geq q$ . Let  $k := \min\{i \in I : x_i = 0\}$ . Then,  $k > q$  holds. Because of

assumption, not all items could be packed. If  $w_k > C/(q + 1)$ , then the  $q$  packed items prove the statement. Otherwise, since the weight  $w_k$  leads to infeasibility, a total weight of more than  $C - C/(q + 1)$  has already been packed. ■

**Corollary 3.3** *Let  $q \geq 2$ ,  $q \in \mathbb{N}$  be given. If  $w_i \leq C/q$  for all  $i \in I$ , then we have*

$$lb_c(E) \leq z^{BPP}(E) \leq \frac{q+1}{q} lb_c(E).$$

The factor  $(q + 1)/q$  is called the absolute worst-case performance ratio of lower bound  $lb_c$ .

**Definition 3.1** Let  $\mathcal{E}$  be the set of all instances  $E$  of a minimization problem, and let  $lb(E)$  be a lower bound of the optimal value  $OPT(E)$  for the instance  $E \in \mathcal{E}$ . The *absolute worst-case performance ratio*  $\rho(lb)$  and the *asymptotic worst-case performance ratio*  $\rho_\infty(lb)$  of lower bound  $lb$  are defined as

$$\rho(lb) := \sup_{E \in \mathcal{E}} \frac{OPT(E)}{lb(E)}, \quad \rho_\infty(lb) := \lim_{k \rightarrow \infty} \sup_{E \in \mathcal{E}: OPT(E) \geq k} \frac{OPT(E)}{lb(E)}.$$

Another natural lower bound results from the maximal number of items which can be packed into a single bin. Let

$$\kappa_1 := \max\{\lfloor C/w_i \rfloor : i \in I\}$$

denote an upper bound of the maximum number of items in a pattern. Then, obviously,

$$lb_\kappa := \lceil n/\kappa_1 \rceil$$

defines a lower bound of  $z^{BPP}$  since not more than  $\kappa_1$  items can be packed into a bin.

Possibly, a tighter bound results if  $\kappa_1$  is replaced by

$$\kappa_2 := \max\left\{\sum_{i \in I} a_i : \sum_{i \in I} w_i a_i \leq C, a_i \in \mathbb{B}, i \in I\right\}.$$

Although  $\kappa_2$  is the optimal value of a knapsack problem, it can be obtained in linear time after sorting the items according to increasing weights. (Sorting of  $n$  numbers takes  $O(n \log n)$  time.)

**Proposition 3.4** *For any instance  $E = (n, w, C)$  of the 1BPP with integer input data, we have*

$$lb_\kappa(E) \leq z^{BPP}(E) \leq \kappa \cdot lb_\kappa(E)$$

where  $z^{BPP}(E)$  denotes the optimal value and  $\kappa$  an upper bound of the maximum number of items in a feasible pattern of instance  $E$ .

Exercise 3.9 asks for a proof of the proposition.

### 3.2.2 Combinatorial Bounds

Let  $p \in [1, \lceil C/2 \rceil] \cap \mathbb{Z}$ . We define index sets  $I(a, b) \subseteq I$  according to

$$I(a, b) := \{i \in I : a < w_i \leq b\}.$$

Then, a tighter bound in comparison to  $lb_c$  is proposed by Martello and Toth [25]. It is oriented on the occurrence of items with larger weight.

$$lb_{MT1}(E) := \max_{p \in \{1, \dots, \lceil C/2 \rceil\}} \left\{ |I(C-p, C)| + \left\lceil \frac{1}{C} \sum_{i \in I(p-1, C-p)} w_i \right\rceil \right\}.$$

The asymptotic worst-case performance ratio of  $lb_{MT1}(E)$  is 1.5, [11].

Similar to a lower bound, proposed by Martello and Toth [26] for the two-dimensional BPP, we can derive another lower bound for the 1BPP by a further splitting of  $I$ . For short, let  $I_1(p) := I(C-p, C)$ ,  $I_2(p) := I(C/2, C-p)$ , and  $I_3(p) := I(p-1, C/2)$ , then

$$lb_{MT2}(E, p) := |I_1(p)| + |I_2(p)| + \max \left\{ 0, \left\lceil \frac{1}{C} \left( \sum_{i \in I_3(p)} w_i - (C \cdot |I_2(p)| - \sum_{i \in I_2(p)} w_i) \right) \right\rceil \right\}.$$

Taking the maximal value, we obtain

$$lb_{MT2}(E) := \max \{lb_{MT2}(E, p) : p \in [1, \lceil C/2 \rceil] \cap \mathbb{Z}\}.$$

As already mentioned in [26], in order to compute  $lb_{MT1}(E)$  or  $lb_{MT2}(E)$ , only values  $p \in \{w_i : i \in I, w_i \leq C/2\}$  have to be considered since for the other the index sets do not change.

In analogy to the concept of (reduced) sets of potential allocation points, see Sect. 2.7, the computational effort can be reduced due to a proposition given in [19].

**Proposition 3.5** *Let  $\tilde{w}_1$  and  $\tilde{w}_2$  denote two neighbored item weights with  $C/2 < \tilde{w}_1 < \tilde{w}_2$  and let  $\tilde{w}_0 := \min_{i \in I} \{w_i : C - \tilde{w}_2 < w_i \leq C - \tilde{w}_1\}$  exist. Then we have*

$$lb_{MT2}(E, \tilde{w}_0) = \max_p \{lb_{MT2}(E, p) : C - \tilde{w}_2 < p \leq C - \tilde{w}_1\}.$$

*Proof* For all  $p \in \mathbb{Z}$  with  $C - \tilde{w}_2 < p \leq C - \tilde{w}_1$ , we obtain  $I_1(p) = I_1(\tilde{w}_0)$ ,  $I_2(p) = I_2(\tilde{w}_0)$ , and  $I_3(p) \subseteq I_3(\tilde{w}_0)$ . ■

Now, we assume that  $w_1 \leq w_2 \leq \dots \leq w_n$  is fulfilled. Then we define the set  $V$  in analogy to the definition of the reduced set of potential allocation points according to

$$V := \left\{ w_j : w_1 < w_j \leq \frac{C}{2}, \exists w_i > \frac{C}{2} \text{ with } w_j + w_i > C, w_{j-1} + w_i \leq C \right\} \cup \{w_1\}.$$

**Corollary 3.6** *Because of Proposition 3.5, we have:  $lb_{MT2}(E) = \max\{lb_{MT2}(E, p) : p \in V\}$ .*

To illustrate this issue we adapt an example taken out of [19].

*Example 3.1* Consider an instance of the 1BPP with input data

$$n = 12, C = 32, w_i \in \{2, 3, 6, 7, 8, 10, 11, 15, 19, 21, 25, 28\}.$$

For better understanding, we use item sets  $W_k(p) := \{w_i : i \in I_k(p)\}$  for all  $k \in \{1, 2, 3\}$  and  $p \in V$  instead of index sets  $I_k(p)$ .

Firstly, we obtain  $V = \{2, 6, 8, 15\}$ . For  $p = 2$  we have  $W_1(2) = \emptyset$ ,  $W_2(2) = \{19, \dots, 28\}$ ,  $W_3(2) = \{2, \dots, 15\}$ . Because of  $W_1(3) = W_1(2)$ ,  $W_2(3) = W_2(2)$  and  $W_1(3) \subset W_1(2)$ ,  $lb_{MT2}(E, 2) \geq lb_{MT2}(E, 3)$ . For  $p = 6$  we obtain  $W_1(6) = \{28\}$ ,  $W_2(6) = \{19, \dots, 25\}$  and  $W_3(6) = \{6, \dots, 15\}$ . For  $p = 7$  we have  $W_1(7) = W_1(6)$ ,  $W_2(7) = W_2(6)$ ,  $W_3(7) \subset W_3(6)$ , and therefore,  $lb_{MT2}(E, 7) \leq lb_{MT2}(E, 6)$ . Moreover,  $W_1(8) = \{25, 28\}$ ,  $W_2(8) = \{19, 21\}$ ,  $W_3(8) = \{8, \dots, 15\}$  and  $W_1(15) = \{19, \dots, 28\}$ ,  $W_2(15) = \emptyset$ ,  $W_3(8) = \{15\}$ .

Finally, we obtain the lower bound  $lb_{MT2}(E) = 5 = lb_{MT2}(E, 6)$ . ■

The bound  $lb_{MT2}(E)$  can be computed in  $O(n)$  time if the item weights are presorted. The worst-case performance ratio of  $lb_{MT2}(E)$  is 1.5, [26].

The bound  $lb_{MT2}(E, p)$  is based on the number of *larger* items; *small* items are disregarded, and the total weights of all items in  $I_3(p)$  is continuously distributed.

Another lower bound can be obtained in analogy to  $lb_\kappa$ . Therefore we consider the maximum number of items that can be packed into a bin if an item of  $I_2(p)$  is already contained therein.

$$lb_s(E, p) := |I_1(p)| + |I_2(p)| + \max \left\{ 0, \left\lceil \frac{|I_3(p)| - \sum_{i \in I_2(p)} \lfloor (C - w_i)/p \rfloor}{\lfloor C/p \rfloor} \right\rceil \right\},$$

$$lb_s(E) := \max\{lb_s(E, p) : p \in \{1, \dots, \lceil C/2 \rceil\}\}.$$

The worst-case performance ratio of  $lb_s$  is not known.

### 3.2.3 Dual Feasible Functions

A concept to possibly strengthen lower bounds is based on the following definition.

**Definition 3.2** Any function  $u : [0, 1] \rightarrow [0, 1]$  is called *dual feasible function* (DFF), if for every finite set  $S$  of positive numbers the implication

$$\sum_{x \in S} x \leq 1 \quad \Rightarrow \quad \sum_{x \in S} u(x) \leq 1 \quad (3.3)$$

holds.

Considering the dual problem

$$\max \sum_{i \in I} u_i \quad \text{s.t.} \quad \sum_{i \in I} x_{ij} u_i \leq 1, \quad j \in J,$$

of the LP relaxation of the pattern-oriented BPP model (3.2),

$$\min \sum_{j \in J} y_j \quad \text{s.t.} \quad \sum_{j \in J} x_{ij} y_j = 1, \quad y_j \geq 0, \quad j \in J,$$

we see, every feasible pattern  $x^j, j \in J$ , has to fulfill condition

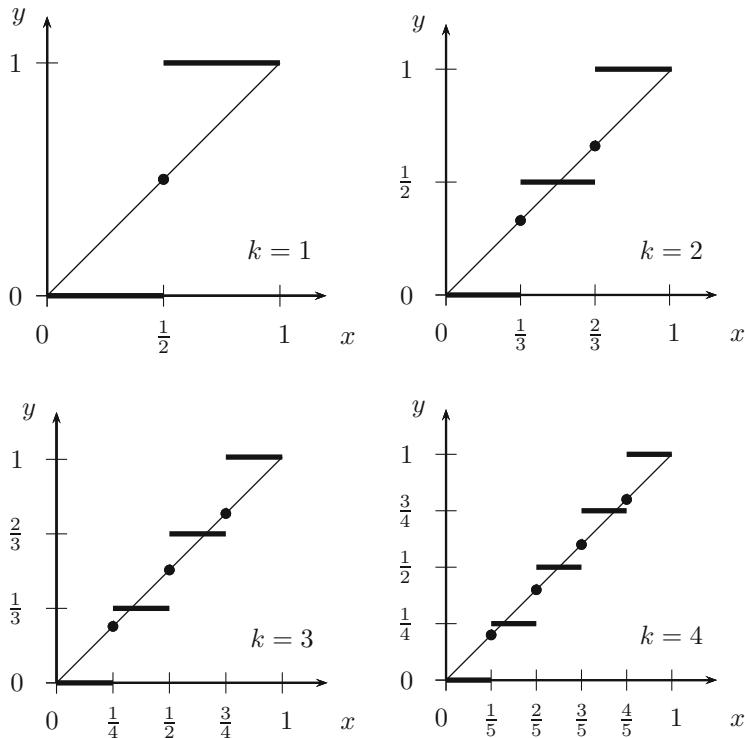
$$u^\top x^j \leq 1.$$

Indeed, scaling the input data of a 1BPP instance  $E = (n, w, C)$  according to  $C' := 1$  and  $w'_i := w_i/C$  for all  $i \in I$  allows the application of DFFs. Let  $E' = (n, w', C')$  denote the resulting normalized instance. Note that we allow item weights of  $E'$  to equal zero to maintain the number of piece types. Obviously, due to (3.3), every feasible pattern of  $E$  is feasible for  $E'$ , as well. Interestingly, it can happen that

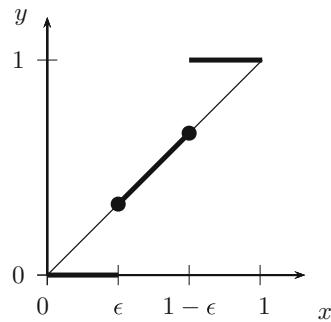
$$lb_c(E) < lb_c(u(E')) \quad \text{where} \quad u(E') = (n, u(\ell'), u(L'))$$

i.e., DFFs can be used to improve the continuous bound.

Notice that if  $x \in \mathbb{B}^n$  is feasible, then each  $x' \in \mathbb{B}^n$  with  $x' \leq x$  (component-wise) is feasible, too. Therefore, we can assume the DFFs to be nonnegative functions. In Figs. 3.1, 3.2, and 3.3 some of the following DFFs are illustrated.



**Fig. 3.1** Dual feasible functions  $y = u^{(k)}(x)$  for  $k = 1, \dots, 4$



**Fig. 3.2** Dual feasible function  $y = U^{(\epsilon)}(x)$

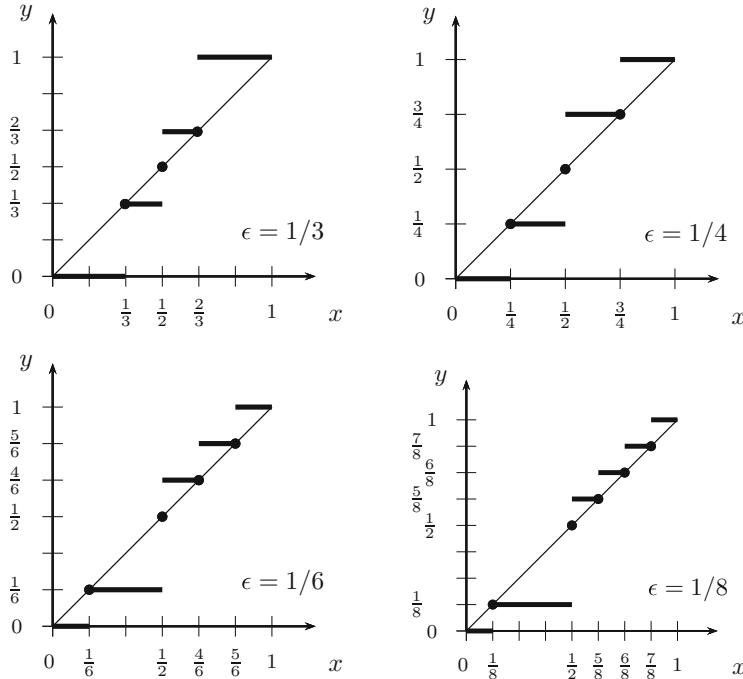


Fig. 3.3 Dual feasible functions  $y = \phi^{(\varepsilon)}(x)$

Exemplarily, we list DFFs proposed in [13]. Let  $k \in \mathbb{N}$  and  $\varepsilon \in [0, \frac{1}{2}]$ .

$$u^{(k)}(x) := \begin{cases} x, & \text{if } x(k+1) \in \mathbb{N}, \\ \lfloor x(k+1) \rfloor / k, & \text{otherwise,} \end{cases}$$

$$U^{(\varepsilon)}(x) := \begin{cases} 1, & \text{if } 1 - \varepsilon < x \leq 1, \\ x, & \text{if } \varepsilon \leq x \leq 1 - \varepsilon, \\ 0, & \text{if } 0 \leq x < \varepsilon. \end{cases}$$

$$\phi^{(\varepsilon)}(x) := \begin{cases} 1 - \frac{\lfloor (1-x)\varepsilon^{-1} \rfloor}{\lfloor \varepsilon^{-1} \rfloor}, & \text{if } 1/2 < x \leq 1, \\ \frac{1}{\lfloor \varepsilon^{-1} \rfloor}, & \text{if } \varepsilon \leq x \leq 1/2, \\ 0, & \text{otherwise.} \end{cases}$$

A survey on dual feasible functions is given in [8] together with dominance considerations and an introduction to *maximal dual feasible functions*. Moreover, a comprehensive analysis of DFFs can be found in [2].

Among many other results available in related literature, we present exemplarily the following proposition which states a relationship between DFFs and lower bounds:

**Proposition 3.7** *Let  $E'$  denote the scaled instance related to 1BPP instance  $E$ . Then we have*

$$lb_{MT2}(E) = \max\{lb_c(U^{(\varepsilon)}(E')) : \varepsilon \in [0, \frac{1}{2}]\}.$$

Using Corollary 3.6, we obtain by means of Proposition 3.7:

$$lb_{MT2}(E) = \max\{lb_c(U^{(\varepsilon)}(E')) : \varepsilon \in V'\} \quad \text{with} \quad V' := \left\{\frac{v}{C} : v \in V\right\} \cup \left\{\frac{1}{2}\right\}.$$

In [13, 19] an improved lower bound  $lb_*^{(q)}(E)$  for  $q \geq 2$  is proposed:

$$lb_*^{(q)}(E) := \max \left\{ lb_{MT2}(E), \max_{2 \leq h \leq q} lb_{MT}^{(h)}(E) \right\} \quad \text{with}$$

$$lb_{MT}^{(h)}(E) := \max \left\{ lb_c(u^{(h)}(U^{(\varepsilon)}(E'))) : \varepsilon \in [0, \frac{1}{2}] \right\}.$$

Moreover, it is shown that bound  $lb_*^{(q)}(E)$  can be computed in  $O(n)$  time for any instance with  $n$  items if their weights are presorted. The asymptotic worst-case performance ratio of  $lb_*^{(q)}(E)$  is  $\rho_\infty = 4/3$ , [11].

A similar concept which does not require scaling of input data is the following:

**Definition 3.3** A function  $f : [0, C] \rightarrow [0, C']$  with  $C, C' \in \mathbb{N}$  is called *discrete dual feasible function* (DDFF) if for any finite set  $S$  of positive integers the following implication holds:

$$\sum_{x \in S} x \leq C \quad \Rightarrow \quad \sum_{x \in S} f(x) \leq f(C) = C'.$$

As example, we give here the DDFF  $u_{CCM}^k$  proposed in [6]. Let  $k \in [1, C/2] \cap \mathbb{N}$ .

$$u_{CCM}^k(x) := \begin{cases} 2(\lfloor C/k \rfloor - \lfloor (C-x)/k \rfloor), & \text{if } x > C/2, \\ \lfloor C/k \rfloor, & \text{if } x = C/2, \\ 2\lfloor x/k \rfloor, & \text{if } x < C/2. \end{cases}$$

The interrelation to dual feasible functions is obvious. Their application is considered, for instance, in [4] and [6] to get stronger lower bounds for two-dimensional bin packing problems.

Moreover, also data-dependent dual feasible functions are known in literature, [6, 23].

**Definition 3.4** Let  $I = \{1, \dots, n\}$ ,  $w \in \mathbb{N}^n$ , and  $C \in \mathbb{N}$  with  $w_i \leq C$  for all  $i \in I$ . A *data-dependent dual feasible function* (DDDF)  $u$  associated with  $C$  and  $w$  is a discrete mapping from  $[0, C]$  to  $[0, C']$  such that

$$\forall \tilde{I} \subseteq I : \sum_{i \in \tilde{I}} w_i \leq C \Rightarrow \sum_{i \in \tilde{I}} u(w_i) \leq C'.$$

Further DFFs can be found, for instance, in [6]. As Fekete and Schepers showed in [13], DFFs can be combined to obtain new ones. This technique is used, for instance, in [5].

### 3.2.4 LP-Based Lower Bound

Considering the 1BPP, in [7] it is shown that the optimal value  $z_{LP}$  of the LP relaxation of model (3.2) provides a strong lower bound for  $z^{BPP}$ .

**Theorem 3.8** *For every instance of the 1BPP, we have*

$$z^{BPP} \leq \frac{4}{3} \lceil z_{LP} \rceil.$$

Moreover, the factor 4/3 is best-possible.

Note that in [7] a 1BPP instance with  $n = 15$  items is given for which  $lb_c = z_{LP} = 3$ ,  $z^{BPP} = 4$ , and hence  $z^{BPP} = \frac{4}{3} \lceil z_{LP} \rceil$  hold.

## 3.3 Reduction Methods and Equivalence of Instances

In many cases reduction methods and equivalence considerations can help to reduce the computational effort for solving problems. Therefore, we introduce a definition of *equivalent instances* of the 1BPP. After that, we address some possibilities of reduction.

**Definition 3.5** The 1BPP instance  $E' = (n', w', C')$  with  $w' \in \mathbb{R}_{>}^{n'}$ ,  $C' \in \mathbb{R}_{>}$  is *equivalent* to the 1BPP instance  $E = (n, w, C)$  if

$$n = n' \wedge \forall a \in \mathbb{B}^n : [a^\top w' \leq C' \Leftrightarrow a^\top w \leq C]. \quad (3.4)$$

Obviously, scaling an instance leads to an equivalent one. However, the definition allows to change specific input data, for instance, to decrease or increase one or

several item weights. With respect to lower bounds, increasing of item weights and/or decreasing of the bin capacity can possibly improve a lower bound. Such a modification of input data, leading to an equivalent instance, is sometimes called *reduction method*. Some of the following ideas are used in lower bounds considered above.

Assigned to 1BPP instance  $E = (n, w, C)$ , let  $w_{\min} := \min_{i \in I} w_i$  and  $w_{\max} := \max_{i \in I} w_i$ . If

$$w_{\min} + w_{\max} > C,$$

then  $E$  contains items which cannot be packed together with another one. In this case, we can increase some weights  $w_i < C$  according to

$$w_i \rightarrow w'_i := C \quad \text{for all } i \in I \text{ with } w_i > C - w_{\min}.$$

If this happens, at least the continuous lower bound  $lb_c$  increases.

A reduction of the bin capacity is possible if

$$C_{\min} := \max\{w^T a : w^T a \leq C, a \in \mathbb{B}^n\} < C$$

holds since for all patterns some unused capacity arises. Moreover, also if  $C = C_{\min}$  holds, possibly specific item weights can be increased. If

$$\mu_i := \max\{w^T a : w^T a \leq C - w_i, a_i = 0, a \in \mathbb{B}^n\} < C - w_i$$

is satisfied, then the weight of item  $i$  can be increased to  $w'_i := C - \mu_i$ .

Another reduction method consists in removing specific items to reduce the size of the instance (number of items). This leads to a non-equivalent one with respect to Definition 3.5. However, the relationship between the optimal values is obvious. For example, if  $w_i + w_{\min} > C$ , then item  $i$  could be removed.

Moreover, if we have two items  $i$  and  $j$  ( $i \neq j$ ) with  $w_i + w_j = C_{\min}$ , then both can be removed since there exists a solution of the 1BPP instance  $E$  which contains the pattern formed by items  $i$  and  $j$ .

This statement can be generalized as follows. We assume that  $w_i + w_j \leq C$  for two items  $i$  and  $j$  ( $i \neq j$ ) and  $C - w_i < w_p + w_q$  for all  $p \neq q \in I \setminus \{i\}$  hold. If there does not exist any item  $k$  with  $w_k > w_j$  and  $w_i + w_k \leq C$ , then there exist a solution having items  $i$  and  $j$  together in a bin.

Note that removing (separating) waste-less patterns does not lead to an optimal solution, in general, as the instance  $E = (6, (4, 4, 4, 7, 7, 7), 12)$  proves. Furthermore, concerning the one-dimensional cutting stock problem, a characterization of equivalent instances by means of maximal feasible and minimal infeasible patterns is given in Sect. 4.7.

### 3.4 Heuristics

Since the 1BPP belongs to the class of NP-hard problems [16], the majority of the related work deals with heuristic approaches. We list here some of the most used methods.

In all of the following heuristics, the items are sequentially assigned to the bins. If a further bin is required to pack the next item, then it has to be *opened*, and it remains open until it is *closed*. Items are only allowed to be packed into open bins.

#### Next Fit (NF)

Place the items in the given order. Place the next item into the current open bin if it fits. Otherwise, close that bin, open a new bin, and pack the current item in the new bin.

In order to illustrate the heuristics, we consider the following instance of the 1BPP with  $n = 10$  items and capacity  $C = 10$ :

$$E = (10, (4, 8, 5, 1, 7, 6, 1, 4, 2, 2), 10).$$

Let  $C(k)$  denote the total weight assigned to bin  $k$ . The NF heuristic fills 6 bins, for short  $\text{NF}(E) = 6$ , as follows: bin 1 with item 1,  $C(1) = 4$ ; bin 2 with item 2,  $C(2) = 8$ ; bin 3 with items 3 and 4,  $C(3) = 5 + 1$ ; bin 4 with item 5,  $C(4) = 7$ ; bin 5 with items 6 and 7,  $C(5) = 6 + 1$ ; and bin 6 with items 8, 9, 10,  $C(6) = 4 + 2 + 2$ .

The NF algorithm is clearly wasteful with respect to the bin capacity, in general, depending on the item weights. Another heuristic which uses more than one open bin, and therefore produce, in general, better results, is

#### First Fit (FF)

Place the items in the given order. Place the next item into the lowest numbered bin in which it fits. If it does not fit into any open bin, open a new bin and pack the current item in the new bin.

The FF heuristic uses in total 5 bins, i.e.  $\text{FF}(E) = 5$ , as follows: bin 1 with items 1, 3, 4,  $C(1) = 4 + 5 + 1$ ; bin 2 with items 2 and 7,  $C(2) = 8 + 1$ ; bin 3 with items 5 and 9,  $C(3) = 7 + 2$ ; bin 4 with items 6 and 8,  $C(4) = 6 + 4$ ; and bin 5 with item 10,  $C(5) = 2$ .

Instead of simply choosing the first bin in which the next item fits one can pack the item into a bin best-possible in a certain sense.

### Best Fit (BF)

Place the items in their given order. Place the next item into that bin which will have the smallest unused capacity after the item is placed in the bin. If it does not fit in any bin, open a new bin and pack the current item in the new bin.

For the considered instance  $E$ , the BF heuristic performs exactly as the FF heuristic, leading to  $\text{BF}(E) = 5$ .

A variation on Best Fit is Worst Fit.

### Worst Fit (WF)

Place the items in their given order. Place the next item into that open bin with largest unused capacity. If it does not fit in any open bin, open a new bin and pack the current item in the new bin.

The WF heuristic uses in total 5 bins, i.e.  $\text{WF}(E) = 5$ , as follows: bin 1 with items 1 and 3,  $C(1) = 4 + 5$ ; bin 2 with items 2 and 4,  $C(2) = 8 + 1$ ; bin 3 with item 5,  $C(3) = 7$ ; bin 4 with items 6 and 7,  $C(4) = 6 + 1$ ; and bin 5 with items 8, 9, 10,  $C(5) = 4 + 2 + 2$ .

Note that the total unused capacity is distributed more uniformly on the open bins in comparison to the FF and BF heuristics.

If it is possible, depending on the practical background, sorting the items according to decreasing (in fact, non-increasing) weights can lead to a better performance of heuristics. In applications, *offline* and *online* scenarios are distinguished. In the offline case, a sorting prior to processing a heuristic can be performed, but not in the online case. We will address this issue in more detail below, in particular for the two-dimensional strip packing problem.

### Next Fit Decreasing (NFD)

Sort the items in decreasing order of weight. Place the next item into the current open bin if it fits. Otherwise, close the current open bin, open a new bin, and pack the current item in the new bin.

Sorting the items according to decreasing weights, we obtain the instance  $\tilde{E}$ :

$$\tilde{E} = (10, (8, 7, 6, 5, 4, 4, 2, 2, 1, 1), 10).$$

The NFD heuristic uses in total 5 bins, i.e.  $\text{NFD}(\tilde{E}) = 5$ , as follows: bin 1 with item 1,  $C(1) = 8$ ; bin 2 with item 2,  $C(2) = 7$ ; bin 3 with item 3,  $C(3) = 6$ ; bin 4 with items 4 and 5,  $C(4) = 5 + 4$ ; and bin 5 with items 6–10,  $C(5) = 4 + 2 + 2 + 1 + 1$ .

### First Fit Decreasing (FFD)

Sort the items in decreasing order. Place the next item into the first open bin in which it fits. If it does not fit into any bin, open a new bin and pack the current item in the new bin.

The FFD heuristic uses in total 4 bins, i.e.  $\text{FFD}(\widetilde{E}) = 4$ , as follows: bin 1 with items 1 and 7,  $C(1) = 8 + 2$ ; bin 2 with items 2, 8, 9,  $C(2) = 7 + 2 + 1$ ; bin 3 with items 3 and 5,  $C(3) = 6 + 4$ ; and bin 4 with items 4, 6, 10,  $C(4) = 5 + 4 + 1$ . Because of  $lb_c(\widetilde{E}) = \lceil \sum_{i \in I} w_i / C \rceil = 4$ , the FFD heuristic provides an optimal solution.

### Best Fit Decreasing (BFD)

Sort the items in decreasing order. Place the next item into that open bin with smallest unused capacity in which it fits. If it does not fit into any open bin, open a new bin and pack the current item in the new bin.

### Worst Fit Decreasing (WFD)

Sort the items in decreasing order. Place the next item into that open bin with largest unused capacity in which it fits. If it does not fit into any bin, open a new bin and pack the current item in the new bin.

Concerning the example, both, the BFD and the WFD heuristic, provide the same solution as the FFD heuristic.

In addition, in the *Minimum Bin Slack* (MBS) heuristic the current bin is filled best-possible with the not yet packed items in each iteration step, [18]. That means, a series of knapsack problems has to be solved, in general. Modifications of the MBS heuristic are applied in [14].

Based on the LP relaxation of model (3.2), a heuristic, named LPB (LP based) heuristic, is proposed in [7] together with performance results (see next subsection). Furthermore, a couple of *folklore* heuristics is collected in [22].

## 3.5 Performance Results for 1BPP Heuristics

To characterize the performance (solution quality) of a heuristic algorithm, we consider its *worst-case* (or *absolute*) and its *asymptotic performance ratio*, similar to the characterization of lower bounds. Additionally, we give some outlook to *average-case* performance results.

### 3.5.1 Worst-Case Performance

Let  $\mathcal{E}$  denote the set of all instances of a minimization problem. We define for  $E \in \mathcal{E}$ :

- $\text{OPT}(E)$  ... optimal value (for BPP: number of bins of an optimal packing),
- $A(E)$  ... objective value of algorithm A  
(for BPP: number of bins of the packing obtained by algorithm A).

**Definition 3.6** A constant  $\rho(A) \in \mathbb{R}_+$  satisfying

$$A(E) \leq \rho(A) \cdot \text{OPT}(E) \quad \text{for all } E \in \mathcal{E}$$

is called *absolute (worst-case) performance bound* or *ratio*. If

$$A(E) \leq \rho_\infty(A) \cdot \text{OPT}(E) + \gamma \quad \text{for all } E \in \mathcal{E},$$

then the factor  $\rho_\infty(A) \in \mathbb{R}_+$  is called the *asymptotic performance bound* or *ratio* of algorithm A where  $\gamma \in \mathbb{R}$  is a constant, too.

**Theorem 3.9** For any instance  $E = (n, w, C)$  of the 1BPP, we have

$$\text{NF}(E) \leq 2 \cdot \text{OPT}(E) + 1$$

implying  $\rho_\infty(\text{NF}) = 2$  and  $\rho(\text{NF}) = 3$ . The ratio 2 is best-possible.

*Proof* Let  $I_j$  denote an index set of the items placed into bin  $j$  by the NF algorithm,  $j = 1, \dots, \text{NF}(E)$ . Then, for  $j = 1, \dots, \lfloor \text{NF}(E)/2 \rfloor$ , we have

$$\sum_{i \in I_{2j-1}} w_i + \sum_{i \in I_{2j}} w_i > C.$$

Adding these inequalities, we get

$$\lfloor \text{NF}(E)/2 \rfloor \cdot C < \sum_{i \in I} w_i.$$

Thus, we obtain

$$\frac{\text{NF}(E) - 1}{2} \leq \left\lfloor \frac{\text{NF}(E)}{2} \right\rfloor < \frac{1}{C} \sum_{i \in I} w_i \leq \text{lb}_c(E) \leq \text{OPT}(E)$$

which proves the claim.

In order to show that 2 is best-possible, we consider a sequence of 1BPP instances  $E_n = (2n, w, n)$ ,  $n = 1, 2, \dots$ , with  $w_{2i-1} = 1$  and  $w_{2i} = n$ ,  $i = 1, \dots, n$ . Obviously,  $\text{NF}(E) = 2n$ , but  $\text{OPT}(E) = n + 1$ . ■

We list some more of the known results. In 1974, Johnson et al. [21] have shown that

$$\text{FF}(E) \leq \frac{17}{10} \text{OPT}(E) + 2 \quad \text{and} \quad \text{BF}(E) \leq \frac{17}{10} \text{OPT}(E) + 2$$

hold as well as

$$\text{FFD}(E) \leq \frac{11}{9} \text{OPT}(E) + 4 \quad \text{and} \quad \text{BFD}(E) \leq \frac{11}{9} \text{OPT}(E) + 4$$

implying

$$\rho_\infty(\text{FF}) = \rho_\infty(\text{BF}) = \frac{17}{10} \quad \text{and} \quad \rho_\infty(\text{FFD}) = \rho_\infty(\text{BFD}) = \frac{11}{9}.$$

Additionally, therein it is also shown that the asymptotic ratios are best-possible. The tightness of 11/9 is addressed in Exercise 3.9. Furthermore, if  $C/5 \leq w_i \leq C$  for all items  $i \in I$  of  $E$ , then  $\text{BFD}(E) = \text{FFD}(E)$  holds, but, if  $C/6 \leq w_i \leq C$  for all items  $i \in I$  of  $E$ , then  $\text{BFD}(E) < \text{FFD}(E)$  is possible as Exercise 3.7 shows. However, in the general case  $\text{BFD}(E) > \text{FFD}(E)$  can happen as well, see Exercise 3.8.

Moreover, similar results are obtained when all weights of an instance are not larger than a fraction of the bin capacity. For that purpose, let  $w_i \leq \omega \cdot C \leq C/2$  hold for all  $i \in I$  with some  $\omega \in \mathbb{R}$ . Then the following result is proposed in [21].

**Theorem 3.10** *For any positive  $\omega \leq 1/2$ , let  $m = \lfloor 1/\omega \rfloor$ . Then we have:*

- (a) *For each  $k \geq 1$  there exists a 1BPP instance  $E = (n, w, C)$  with  $\text{OPT}(E) = k$  such that*

$$\text{FF}(E) \geq \frac{m+1}{m} \text{OPT}(E) - \frac{1}{m}$$

*holds.*

- (b) *For any 1BPP instance  $E = (n, w, C)$  with  $w_i \leq \omega \cdot C$  for all  $i \in I$ , the relation*

$$\text{FF}(E) \leq \frac{m+1}{m} \text{OPT}(E) + 2$$

*is satisfied. Both, (a) and (b), hold as well if FF is replaced by BF.*

In 2010, in [27] the following results were presented:

$$\text{FF}(E) \leq \frac{17}{10} \text{OPT}(E) + \frac{7}{10} \quad \text{and} \quad \rho(\text{FF}) \leq \frac{12}{7}.$$

Concerning the FFD heuristic, an assertion easy to prove is

**Theorem 3.11** *For any instance  $E$  of the 1BPP, we have*

$$\text{FFD}(E) \leq 1.5 \cdot \text{OPT}(E)$$

implying  $\rho(\text{FFD}) \leq 1.5$ .

*Proof* For short, let  $k := \text{FFD}(E)$ . We consider the number  $j := \lceil \frac{2}{3}k \rceil$ .

If bin  $j$  contains an item  $i$  with  $w_i > C/2$ , then each bin  $j' < j$  is filled more than  $C/2$  because of sorting the items in decreasing order. Hence,

$$\text{OPT}(E) \geq j = \lceil \frac{2}{3}k \rceil \geq \frac{2}{3}k.$$

Otherwise, bin  $j$  and any bin  $j' > j$  does not contain any item with weight larger than  $C/2$ . Hence, the bins  $j, j+1, \dots, k$  contain at least  $2(k-j) + 1$  items, none of them fits into the bins  $1, \dots, j-1$ . Thus, we have

$$\begin{aligned} \frac{1}{C} \sum_{i \in I} w_i &> \min\{j-1, 2(k-j)+1\} \\ &= \min\{\lceil \frac{2}{3}k \rceil + 1, 2(k - \lceil \frac{2}{3}k \rceil) - 1\} \\ &= \lceil \frac{2}{3}k \rceil - 1. \end{aligned}$$

Since  $\text{OPT}(E) \geq lb_c(E)$  and  $\text{OPT}(E) \in \mathbb{Z}_+$ , this implies

$$\text{OPT}(E) \geq \lceil \frac{2}{3}k \rceil \geq \frac{2}{3} \cdot \text{FFD}(E)$$

and, hence, the assertion. ■

In 1974, Johnson et al. proposed that  $\text{FFD}(E) \leq (11/9)\text{OPT}(E) + 4$  holds. Meanwhile, it is shown in [12] that

$$\text{FFD}(E) \leq \frac{11}{9}\text{OPT}(E) + \frac{2}{3} \quad \text{and} \quad \rho_\infty(\text{FFD}) = \frac{11}{9} < \frac{3}{2} \leq \rho(\text{FFD}). \quad (3.5)$$

Moreover, instances are presented therein proving the tightness of  $3/2$ . Such an instance is  $E = (n, w, C)$  with  $n = 20$ ,  $C = 60$ , and

$$w_i = \begin{cases} 31, & i = 1, \dots, 4, \\ 17, & i = 5, \dots, 8, \\ 16, & i = 9, \dots, 12, \\ 13, & i = 13, \dots, 20, \end{cases}$$

(see Exercise 3.5). The relation  $1.5 \leq \rho(\text{FFD})$  is addressed in Exercise 3.6.

Concerning the LP-based heuristic LPB, proposed in [7], the following performance result is known:

**Theorem 3.12** *For every instance  $E$  of the 1BPP, we have*

$$LPB(E) \leq \frac{4}{3} \lceil z_{LP}^{BPP} \rceil \leq \frac{4}{3} z^{BPP}.$$

That means, the absolute worst-case performance ratio, and hence the asymptotic one, of the LP based heuristic is bounded by  $4/3$ . As we will see in the CSP chapter, there is not known any 1BPP instance having a gap between  $z^{BPP}$  and  $z_{LP}^{BPP}$  larger than 2. We will address this issue in more detail in Sect. 4.11.

A result which does not hold for the 1CSP is

**Proposition 3.13** *For every instance  $E = (n, w, C)$  of the 1BPP with  $z^{BPP} \leq 3$ , we have*

$$z^{BPP} = \lceil z_{LP}^{BPP} \rceil.$$

*Proof* For short, let  $z_{LP} := z_{LP}^{BPP}$ ,  $W := \sum_{i \in I} w_i$ , and  $\bar{C} := W/z_{LP}$ . Thus, since  $\bar{C}$  is the average value of capacity used in the bins of the LP solution, we have  $\bar{C} \leq C_{max}$  where

$$C_{max} := \max \left\{ \sum_{i \in I} w_i x_i : \sum_{i \in I} w_i x_i \leq C, x_i \in \mathbb{B}, i \in I \right\}.$$

Furthermore, due to the definition of  $\bar{C}$ , we obtain

$$z_{LP} = \frac{W}{\bar{C}} \geq \frac{W}{C_{max}} \geq \frac{W}{C}.$$

If  $z^{BPP} = 1$  holds, then  $z_{LP} = 1$  is necessarily satisfied.

If  $z^{BPP} = 2$  holds, then the relations  $W > C$  and, hence,  $z_{LP} > 1$  are fulfilled.

In case of  $z^{BPP} = 3$  let  $I^* \subset I$  be an appropriate index set with  $\sum_{i \in I^*} w_i = C_{max}$ . Then,  $\sum_{i \in I \setminus I^*} w_i > C$  follows since otherwise a contradiction would arise. Hence, we have  $W > C_{max} + C$  and obtain

$$z_{LP} \geq \frac{W}{C_{max}} > \frac{C_{max} + C}{C_{max}} \geq 2$$

implying  $\lceil z_{LP} \rceil = 3$ . ■

A remarkable improvement in bounding the gap between the heuristic and optimal value is presented in [20] where a logarithmic additive integrality gap is proposed for an LP-based polynomial time approximation algorithm.

### 3.5.2 Average-Case Performance

Besides statements concerning the worst-case or asymptotic performance, frequently assertions with respect to the *average case* performance of heuristics are of interest.

Their detailed formulation and verification is not the aim of this book since considerable knowledge of stochastic theory is needed for such purpose. Instead, we simply present some results for information and to give some insights in such investigations.

To give an example of an average-case performance result, we consider the one-dimensional bin packing problem. Average-case results are based on a certain distribution of the input data. We assume that the item weights  $w_i$  are uniformly distributed random numbers in  $(0, 1]$  for all  $i \in I$ . The bin capacity is  $W = 1$ . Moreover, we consider the following simple heuristic FOLD.

#### FOLD heuristic

- (1) Choose parameter  $\alpha \in (\frac{1}{2}, 1)$ .
- (2) Pack all  $w_i$  with  $w_i > \alpha$  in a single bin.
- (3) Sort the remaining values  $w_i$  according to  $w_1 \leq w_2 \leq \dots \leq w_{n'}$ .
- (4) If  $w_i + w_{n-i+1} \leq 1$ , then pack both pieces in a bin, otherwise in two,  $i = 1, \dots, [n'/2]$ . If  $n'$  is odd, then pack the last piece in an extra bin.

Let  $E(FOLD(n))$  denote the expectation (value) of the number of bins required by the FOLD heuristic in dependence of the number  $n$  of pieces. The following theorem is presented in [15].

**Theorem 3.14 (Frederickson [15])** *For  $\alpha = 1 - n^{-1/3}$  and  $n \in \mathbb{N}$ , we have*

$$\frac{E(FOLD(n))}{n/2} = 1 + O(n^{-1/3}).$$

Other examples for such kind of statements are given, for instance, in [10]. Therein the two-dimensional problem of packing rectangles into a strip of minimal height is addressed and simple heuristics are considered with respect to their average performance. Continuative investigations can be found in the book [9].

## 3.6 Exact Approaches and Extensions

For instances of small or medium size, the ILP model of the 1BPP, presented in Sect. 3.1, can be used to obtain optimal solutions with guarantee.

A branch-and-bound algorithm to solve the 1BPP is developed by Martello and Toth in [25]. A faster approach is proposed in [24].

Depending on the magnitude of input data (of  $C$ ), the ILP formulation of the arcflow model for the 1CSP, Sect. 4.8.4, can be used to solve a 1BPP to optimality.

The scenario of several different bin capacities (lengths) is also of interest within bin packing problems. Such problems are frequently called *variable size* or *multiple bin packing problem*. Exact and heuristic approaches are proposed, e.g., in [1] and [3] where also additional practical restrictions are considered. We will address problems with different (stock) lengths in Sect. 4.10 in more detail.

### 3.7 Exercises

**Exercise 3.1** Consider the instance  $E = (20, (10, \dots, 19, 1, \dots, 10), 20)$  and apply all heuristics described in Sect. 3.4. What is the optimal value?

**Exercise 3.2** Consider the 1BPP instance  $E_n = (n, w, C)$  with  $n = 18k$ ,  $k \in \mathbb{N}$ ,  $C = 120$ , and

$$w_i = \begin{cases} 18, & i = 1, \dots, 6k, \\ 41, & i = 6k + 1, \dots, 12k, \\ 61, & i = 12k + 1, \dots, 18k. \end{cases}$$

Verify that  $\rho(\text{FF}) \geq 5/3$  and  $\rho(\text{BF}) \geq 5/3$  are implied by  $E_n$  for any  $k \in \mathbb{N}$ .

**Exercise 3.3** Let  $E = (n, w, C)$  be a 1BPP instance. Prove, if  $w_i \leq w_{i+1}$  for  $i = 1, \dots, n - 1$ , then  $\text{NF}(E) = \text{FF}(E) = \text{BF}(E) = \text{WF}(E)$  holds.

**Exercise 3.4** Consider the 1BPP instance  $E_n = (n, w, C)$  with  $n = 37k$ ,  $k \in \mathbb{N}$ ,  $C = 101$ , and

$$w_i = \begin{cases} 51, & i = 1, \dots, 10k, \\ 34, & i = 10k + 1, \dots, 20k, \\ 16, & i = 20k + 1, \dots, 23k, \\ 10, & i = 23k + 1, \dots, 30k, \\ 6, & i = 30k + 1, \dots, 37k. \end{cases}$$

Apply some heuristics, in particular the FF heuristic for the given and the reverse order of weights. Which lower bounds for  $\rho(\text{FF})$  and  $\rho(\text{BF})$  are obtained?

**Exercise 3.5** Show the tightness of the statement in (3.5) by means of the normalized instances  $E_k = (n, w, C)$  with  $n = 20 + 30k$ ,  $k = 0, 1, \dots$ ,  $C = 1$ ,  $\varepsilon > 0$

sufficiently small, and

$$w_i = \begin{cases} 1/2 + \varepsilon, & i = 1, \dots, 4 + 6k, \\ 1/4 + 2\varepsilon, & i = 5 + 6k, \dots, 8 + 12k, \\ 1/4 + \varepsilon, & i = 9 + 12k, \dots, 12 + 18k, \\ 1/4 - 2\varepsilon, & i = 13 + 18k, \dots, 20 + 30k. \end{cases}$$

**Exercise 3.6** Show that  $\rho(\text{FFD}) \geq 3/2$  by means of the normalized instances  $E = (n, w, C)$  with  $n = 6$ ,  $C = 1$ ,  $\varepsilon > 0$  sufficiently small, and

$$w_i = \begin{cases} 1/3 + 2\varepsilon, & i = 1, 2, \\ 1/3 + \varepsilon, & i = 3, 4, \\ 1/3 - 3\varepsilon, & i = 5, 6. \end{cases}$$

**Exercise 3.7 ([21])** Show that  $\text{BFD}(E) < \text{FFD}(E)$  is possible by means of the normalized instances  $E_k = (n, w, C)$  with  $n = 30k$ ,  $k = 1, 2, \dots$ ,  $C = 1$ ,  $\varepsilon > 0$  sufficiently small, and

$$w_i = \begin{cases} 3/5 + 2\varepsilon, & i = 1, \dots, 5k, \\ 2/5, & i = 5k + 1, \dots, 15k, \\ 1/5, & i = 15k + 1, \dots, 20k, \\ 1/5 - \varepsilon, & i = 20k + 1, \dots, 30k. \end{cases}$$

**Exercise 3.8 ([21])** Show that  $\text{BFD}(E) > \text{FFD}(E)$  can happen by means of the normalized instances  $E_k = (n, w, C)$  with  $n = 33k$ ,  $k = 1, 2, \dots$ ,  $C = 1$ ,  $\varepsilon > 0$  sufficiently small, and

$$w_i = \begin{cases} 2/3, & i = 1, \dots, 3k, \\ 1/3 + \varepsilon, & i = 3k + 1, \dots, 15k, \\ 1/6, & i = 15k + 1, \dots, 21k, \\ 1/6 - \varepsilon, & i = 21k + 1, \dots, 33k. \end{cases}$$

**Exercise 3.9** Give a proof of Proposition 3.4 on page 51.

## 3.8 Solutions

**To Exercise 3.1** We obtain  $\text{NF}(E) = 14$ ,  $\text{NFD}(E) = 13$ ,  $\text{FF}(E) = \text{WF}(E) = 12$ ,  $\text{BF}(E) = \text{FFD}(E) = \text{BFD}(E) = \text{WFD}(E) = 10$ ,  $\text{MBS}(E) = 11$  (without sorting), and  $\text{MBSD}(E) = 11$  (with sorting according to decreasing weights).

Because of  $lb_c(E) = 10$ , we have  $z^{BPP} = 10$ .

**To Exercise 3.2** Obviously,  $z^{BPP} = 6k$  for any  $k \in \mathbb{N}$  since for each  $i = 1, \dots, 6k$ , we have  $w_i + w_{6k+i} + w_{12k+i} = C$ . However, the FF heuristic packs  $k$  bins each with six items of weight 18, 3k bins each with two items of weight 41, and 6k bins each with one item of weight 61. Hence,  $\text{FF}(E_n) = 10k$ , and therefore,  $\rho(\text{FF}) \geq \text{FF}(E_n)/\text{OPT}(E_n) = 5/3$ .

**To Exercise 3.3** Obvious.

**To Exercise 3.4** For the given order of weights, we obtain  $\text{FF}(E_n) = \text{OPT}(E_n) = 10k$ . However, for increasing weights we obtain  $\text{FF}(E_n) = \text{BF}(E_n) = 17k$  implying  $\rho(\text{FF}) \geq \text{FF}(E_n)/\text{OPT}(E_n) = 17/10$ .

**To Exercise 3.5** Let  $t_k$ ,  $k = 1, \dots, 4$  denote the four item types. We represent any pattern  $a \in \mathbb{B}^n$  as a tuple  $\bar{a} \in \mathbb{Z}^4$  where  $\bar{a}_k$  counts the number items of type  $k$  that are contained in  $a$ . (This is the compact representation of patterns in the cutting stock problem, see Sect. 1.6 or Chap. 4.) An optimal packing is given by  $4 + 6k$  times  $(1, 0, 1, 1)$ , and  $2 + 3k$  times  $(0, 2, 0, 2)$ . Hence, in total  $9k + 6$ . According to the FFD heuristic, we obtain  $4 + 6k$  times  $(1, 1, 0, 0)$ ,  $1 + 2k$  times  $(0, 0, 3, 0)$ , once  $(0, 0, 3, 1)$ ,  $1 + 3k$  times  $(0, 0, 0, 4)$ , and once  $(0, 0, 0, 1)$ . Hence, in total  $11k + 8$ .

**To Exercise 3.6** It is obvious,  $\text{OPT}(E) = 2$  and  $\text{FFD}(E) = 3$ .

**To Exercise 3.7** According to the BFD heuristic, we obtain the following (compact) patterns:  $5k$  times  $(1, 0, 0, 2)$  and  $5k$  times  $(0, 2, 1, 0)$ . Hence,  $\text{BFD}(E_k) = 10k$ . The FFD heuristic produces the (compact) patterns  $5k$  times  $(1, 0, 1, 0)$ ,  $5k$  times  $(0, 2, 0, 1)$ , and  $k$  times  $(0, 0, 0, 5)$ . Hence,  $\text{FFD}(E_k) = 11k$ .

**To Exercise 3.8** According to the FFD heuristic, we obtain the following (compact) patterns:  $3k$  times  $(1, 0, 2, 0)$  and  $6k$  times  $(0, 2, 0, 2)$ . Hence,  $\text{FFD}(E_k) = 9k$ . The BFD heuristic produces the (compact) patterns  $3k$  times  $(1, 0, 0, 2)$ ,  $6k$  times  $(0, 2, 1, 0)$ , and  $k$  times  $(0, 0, 0, 5)$ . Hence,  $\text{BFD}(E_k) = 10k$ .

**To Exercise 3.9** Let  $y_j^*, x_{ij}^*$ ,  $i \in I, j \in J^* := \{j \in J : y_j^* = 1\}$ , represent an optimal packing of  $E$ , that means,  $z^{BPP} = \sum_{j \in J^*} y_j^* = |J^*|$ . Then, because of (3.2b), we have

$$n = \sum_{i \in I} \sum_{j \in J^*} x_{ij} \leq \kappa \sum_{j \in J^*} 1 = \kappa \cdot z^{BPP} \quad \text{and} \quad z^{BPP} \leq n = \kappa \frac{n}{\kappa} \leq \kappa \cdot lb_\kappa.$$

The factor  $\kappa$  is best-possible due to the following instance: for some  $r \in \mathbb{N}$  let  $n = (r+1)\kappa$ ,  $w_i = W/\kappa$  for  $i = 1, \dots, \kappa$ , and  $w_i = W/2 + \varepsilon$  for  $i = \kappa+1, \dots, n$  with a sufficiently small  $\varepsilon > 0$ . Then,  $lb_\kappa = r+1$  and  $z^{BPP} = r \cdot \kappa + 1$  hold implying  $z^{BPP}/lb_\kappa \rightarrow \kappa$  for  $r \rightarrow \infty$ .

## References

1. C. Alves, J.M.V. de Carvalho, Accelerating column generation for variable sized bin-packing problems. *Eur. J. Oper. Res.* **183**(3), 1333–1352 (2007)
2. C. Alves, F. Clautiaux, J.V. de Carvalho, J. Rietz, *Dual-Feasible Functions for Integer Programming and Combinatorial Optimization* (Springer, Berlin, 2016)
3. J. Bang-Jensen, R. Larsen, Efficient algorithms for real-life instances of the variable size bin packing problem. *Comput. Oper. Res.* **39**(11), 2848–2857 (2012)
4. M.A. Boschetti, A. Mingozzi, The two-dimensional finite bin packing problem. part I: new lower bounds for the oriented case. *4OR* **1**, 27–42 (2003)
5. M.A. Boschetti, L. Montaletti, An exact algorithm for the two-dimensional strip-packing problem. *Oper. Res.* **58**, 1774–1791 (2010)
6. J. Carlier, F. Clautiaux, A. Moukrim, New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Comput. Oper. Res.* **34**, 2223–2250 (2007)
7. L.M.A. Chan, D. Simchi-Levi, J. Bramel, Worst-case analysis, linear programming and the bin-packing problem. *Math. Program.* **83**, 213–227 (1998)
8. F. Clautiaux, C. Alves, J.V. de Carvalho, A survey of dual-feasible and superadditive functions. *Ann. OR* **179**(1), 317–342 (2010)
9. E.G. Coffman Jr., G.S. Luecker, *Probabilistic Analysis of Packing and Partitioning Algorithms* (Wiley, New York, 1991)
10. E.G. Coffman, P.W. Shor, Average-case analysis of cutting and packing in two dimensions. *Eur. J. Oper. Res.* **44**(2), 134–144 (1990)
11. T.G. Crainic, G. Perboli, M. Pezzuto, R. Tadei, Computing the asymptotic worst-case of bin packing lower bounds. *Eur. J. Oper. Res.* **183**, 1295–1303 (2007)
12. G. Dosa, The tight bound of first fit decreasing bin-packing algorithm is  $\text{FFD(I)} \leq 11/9 \text{OPT(I)} + 6/9$ , in *ESCAPE 2007* (Springer, Heidelberg, 2007), pp. 1–11
13. S.P. Fekete, J. Schepers, A general framework for bounds for higher-dimensional orthogonal packing problems. *Math. Methods Oper. Res.* **60**(2), 311–329 (2004)
14. K. Fleszar, K.S. Hindi, New heuristics for one-dimensional bin-packing. *Comput. Oper. Res.* **29**, 821–839 (2002)
15. G.N. Frederickson, Probabilistic analysis for simple one- and two-dimensional bin packing algorithms. *Inf. Process. Lett.* **11**, 156–161 (1980)
16. M.R. Garey, D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
17. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting-stock problem (Part I). *Oper. Res.* **9**, 849–859 (1961)
18. J.N.D. Gupta, J.C. Ho, A new heuristic algorithm for the one-dimensional bin-packing problem. *Prod. Plan. Control* **10**(6), 598–603 (1999)
19. M. Haouari, A. Gharbi, Fast lifting procedures for the bin packing problem. *Discrete Optim.* **2**, 201–218 (2005)
20. R. Hoberg, T. Rothvoss, A logarithmic additive integrality gap for bin packing. Technical report, Univ. of Washington, Seattle (2015)
21. D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.* **3**(4), 299–325 (1974)
22. J. Jylänki, A thousand ways to pack the bin – practical approach to two-dimensional rectangle bin packing. Technical report, online (2010)
23. A. Khanafer, F. Clautiaux, E. Talbi, New lower bounds for bin packing problems with conflicts. *Eur. J. Oper. Res.* **206**(2), 281–288 (2010)
24. R.E. Korf, Optimal rectangle packing: new results, in *ICAPS-04 Proceedings* (2004), pp. 142–149

25. S. Martello, P. Toth, *Knapsack Problems* (Wiley, Chichester, 1990)
26. S. Martello, P. Toth, Lower bounds and reduction procedures for the bin packing problem. *Discrete Appl. Math.* **28**(1), 59–70 (1990)
27. B. Xia, Z. Tan, Tighter bounds of the first fit algorithm for the bin-packing problem. *Discrete Appl. Math.* **158**, 1668–1675 (2010)

# Chapter 4

## One-Dimensional Cutting Stock

In many applications as, for instance, in round timber cutting or in the furniture industry where lots of rectangular pieces have to be cut, a single piece of raw material is not sufficient to satisfy all order demands. In case of large demands, in general, a considerable number of raw material pieces of identical or various different sizes, is needed. Here, one of the optimization targets consists in the minimization of material usage with respect to some predefined objective function. This problem is known within the relevant literature as the *Cutting Stock Problem* (CSP). Because of its importance, we already mentioned an ILP model in Sect. 1.6 for the case of identical raw material.

In difference to *one-dimensional Bin Packing Problems* (1BPP) where each item is considered to be a unique one, in a *one-dimensional Cutting Stock Problem* (1CSP, or simply CSP), the number  $m$  of different piece types is rather small, but their order demands (or availability in case of packing problems) are mostly large.

Another differentiator of bin packing and cutting stock problems could be the magnitude of the respective optimal value  $z^*$ . If  $z^*$  is small in comparison to the total number of items, then the instance is of BPP type, otherwise the problem type depends on the number of different patterns in a solution. Nevertheless, we can model the BPP and the CSP in the same way, for instance as the pattern-oriented Gilmore/Gomory model (Sect. 4.2) or by means of other models presented in Sect. 4.8.

Solution methods proposed for the CSP which are based on the LP relaxation are, in general, not useful for BPP since mostly the LP solution consists of as many patterns as item types whereas  $z^*$  is expected to be small in comparison to the number of items. Therefore, heuristics for CSP which are based on appropriate rounding an LP solution are less attractive.

In the beginning of this chapter, we consider the 1CSP with a single type of raw material. We present a solution strategy which is also applicable to higher-dimensional problems as, for instance, in the furniture industry when the production of rectangular pieces has to be optimized. Subsequently, we address generalizations

and present alternative models. Finally, we investigate the relation between the standard ILP model and its LP relaxation and observe a small gap for any 1CSP instance.

## 4.1 Problem Statement and Notations

A typical statement of the one-dimensional cutting stock problem is as follows:  $b_i \in \mathbb{N}$  pieces of type (length)  $\ell_i \in \mathbb{R}_>$  have to be cut out of identical raw material of length  $L \in \mathbb{R}_>$  for all  $i \in I := \{1, \dots, m\}$ . Which minimal number of raw material pieces is needed to satisfy all order demands? And which patterns have to be applied?

In a two-dimensional scenario, rectangular pieces of size  $\ell_i \times w_i$  and order demand  $b_i$ ,  $i \in I$ , have to be obtained from a minimal number of rectangular plates (for instance, glass panes) of size  $L \times W$ . In a similar way, three-dimensional cutting stock and bin packing problems can be formulated. Without loss of generality, we assume in the following (if not stated otherwise) that all input data are integers.

As defined in Chap. 1, a feasible cutting pattern is an arrangement of desired objects which satisfies all cutting conditions. In the case of cutting stock problems with homogeneous stock material, we can simply represent a pattern by a nonnegative integer vector

$$a = (a_1, \dots, a_m)^\top \in \mathbb{Z}_+^m$$

where component  $a_i$  indicates how many copies of piece type  $i \in I$  are produced if one item of raw material is cut according to pattern  $a$ . As mentioned above, since the real positions of the pieces are not provided by vector  $a$ , several different arrangements belong to  $a$ , in general. Therefore, in a formal sense, we need to differentiate between a pattern (an arrangement) and its representation by a *pattern vector*  $a$ , but, in order to keep it simple,  $a$  is called a pattern, too.

In the one-dimensional cutting stock problem, vector  $a \in \mathbb{Z}_+^m$  represents a feasible pattern if and only if

$$\ell^\top a = \sum_{i \in I} \ell_i a_i \leq L \tag{4.1}$$

holds where  $\ell = (\ell_1, \dots, \ell_m)^\top$ . Otherwise,  $a$  is called *infeasible*.

*Remark 4.1* In reality, often a positive (saw) kerf has to be regarded. The handling of such a situation is considered in Exercise 1.1.

We call a pattern vector  $a \in \mathbb{Z}_+^m$  *elementary* or *homogeneous* if it is an integer multiple of a unit vector, i.e., if only pieces of the same type are obtained. Pattern  $a$  is said to be *maximal* if  $\ell^\top a + \min\{\ell_i : i \in I\} > L$ , that means, no useful leftover remains.

In case of small order demands, as for instance in the bin packing problem, in particular, if  $b_i < L/\ell_i$  for some  $i \in I$ , the supplementary restrictions

$$a_i \leq b_i \quad \text{for all } i \in I \text{ with } b_i < L/\ell_i \quad (4.2)$$

can be meaningful to define feasibility of a pattern. We say, pattern  $a$  is *proper* if it is feasible and satisfies condition (4.2).

To distinguish various pattern vectors, we use upper index  $j \in \mathbb{N}$ . Let index set  $J$  represent a set of feasible patterns  $a^j = (a_{1j}, \dots, a_{mj})^\top \in \mathbb{Z}_+^m$ ,  $j \in J$ , in the respective models; frequently the set of all feasible patterns.

Within this chapter,  $E = (m, \ell, L, b)$  denotes an instance of the 1CSP where  $b \in \mathbb{N}^m$  indicates the order demands. Note that in contrast to the 1CSP, in case of the 1BPP, we always assume  $b_i = 1$  for all piece types. Hence, an instance  $E = (m, \ell, L, b)$  of the 1CSP can be interpreted as an instance  $E' = (n, w, W)$  of the 1BPP where  $n = \sum_{i \in I} b_i$ ,  $W = L$ , and, e.g.,  $w_j = \ell_i$  for  $j = \sum_{k=1}^{i-1} b_k + 1, \dots, \sum_{k=1}^i b_k$ ,  $i = 1, \dots, m$ .

## 4.2 The Gilmore/Gomory Model

In case of the cutting stock problem, we do not regard (4.2), if not stated otherwise, since we assume larger order demands in a cutting stock problem, as for instance,  $b_i > L/\ell_i$  for all  $i \in I$ . In the latter case, conditions (4.2) are redundant.

In order to formulate a mathematical model as proposed by Gilmore and Gomory [16], let  $J$  be an appropriate index set of the set of all feasible patterns  $a^j \in \mathbb{Z}_+^m$ ,  $j \in J$ . We define for every feasible pattern  $a^j$  a *nonnegative integer variable*  $x_j$  which indicates the frequency pattern  $a^j$  has to be cut. In this way, we obtain the following ILP model of the cutting stock problem which is named the *Gilmore/Gomory model*. Frequently, it is called the *standard* or *pattern-oriented* model, too.

### Gilmore/Gomory model of the cutting stock problem

$$z^* = z^{CSP} = \min \sum_{j \in J} x_j \quad \text{s.t.} \quad (4.3a)$$

$$\sum_{j \in J} a_{ij} x_j \geq b_i, \quad i \in I, \quad (4.3b)$$

$$x_j \in \mathbb{Z}_+, \quad j \in J. \quad (4.3c)$$

The objective function (4.3a) counts the total number of raw material pieces, needed to fulfill all order demands,. Restrictions (4.3b) ensure that sufficiently many copies of type  $i$  are obtained for all  $i \in I$ . Naturally, all variables have to be integers since they model frequencies of using patterns, see condition (4.3c). Note that overproduction is allowed which implies that only maximal patterns should be considered within the model.

It is obvious that model (4.3) is also a model of the bin packing problem. But, regarding conditions (4.2) which, in general, reduces the total number of patterns, leads to a more adjusted model for the bin packing problem. We will concern this aspect in subsequent sections.

*Remark 4.2* The minimization of the total number of raw material pieces according to (4.3a) is equivalent to the minimization of total waste if overproduction is counted as waste, too. This aspect is addressed in Exercise 4.1.

Let  $n$  denote the total number of patterns involved in the considered model, i.e.  $n := |J|$ . Using matrix  $A := (a^j)_{j \in J}$  and vectors  $e = (1, \dots, 1)^\top \in \mathbb{R}^n$ ,  $x = (x_1, \dots, x_n)^\top$ , and  $b = (b_1, \dots, b_m)^\top$ , model (4.3) can be written in the compact form

$$z^{CSP} = \min e^\top x \quad \text{s.t.} \quad Ax \geq b, \quad x \in \mathbb{Z}_+^n. \quad (4.4)$$

If index set  $J$  represents *all* feasible patterns, then model (4.4) is equivalent to

$$z^{CSP} = \min e^\top x \quad \text{s.t.} \quad Ax = b, \quad x \in \mathbb{Z}_+^n. \quad (4.5)$$

This equivalence (see Exercise 4.2) of both ILP models consists in the identity of the optimal values. Obviously, every solution of (4.5) is also a solution of (4.4), but not vice versa.

As proposed in this section, the cutting stock problem can be formulated as an ILP problem with, in general, an exponential number of variables. It belongs to the class of NP-hard problems, [15], i.e., with high probability there is no exact solution method with polynomial time complexity. For that reason, branch-and-bound algorithms, or cutting plane methods, or a combination of both (leading to *branch-and-cut*) have to be applied. A serious disadvantage of these approaches is constituted in the hardly estimable computational effort needed to solve an instance. Therefore, when tackling instances of practical relevance, one often has to be satisfied with an approximate solution. For that reason, a common solution approach for the cutting stock problem consists in solving the *continuous relaxation* of model (4.3) and applying heuristics which construct integer solutions based on the, in general, noninteger solution of the relaxation. The continuous relaxation of (4.3), frequently called *LP relaxation*, results by omitting the integer condition:

$$z_{LP}^{CSP} = \min e^\top x \quad \text{s.t.} \quad Ax \geq b, \quad x \in \mathbb{R}_+^n. \quad (4.6)$$

Since the optimal value of (4.6) provides a lower bound for any feasible solution of the integer problem, it constitutes a useful measure for the quality of feasible integer solutions obtained by heuristics.

*Remark 4.3* We will investigate in more detail the relationship between the optimal values of the 1CSP and its LP relaxation in Sect. 4.11. There is not known any instance with difference of optimal values larger than 1.2.

In case of round timber cutting, and for many other applications of the cutting stock problem, the number  $n$  of all patterns is, in general, as large as not longer manageable. The construction alone would take a too long time, or storing all patterns would require a huge amount of memory. If, for instance, all lengths  $\ell_i$ ,  $i \in I$ , and the material length  $L$  are characterized by the relations  $L/10 \leq \ell_i \leq L/5$  for all  $i \in I$ , then we can expect that  $n > 10^6$  holds for  $m = 40$ , and  $n > 10^8$  for  $m = 60$ . A recursion to compute the number of feasible and maximal patterns is considered in Exercise 4.3.

The exponential increase of  $n$  might induce to consider only a subset of all patterns within the model. But, for this scenario we do not have any meaningful estimation how good the resulting solution is. Example 4.1 illustrates this problematic aspect as we will see in the following.

*Example 4.1* Let rods of lengths  $L = 70$  be given. Pieces of lengths  $\ell_1 = 20$ ,  $\ell_2 = 22$ ,  $\ell_3 = 25$ , and  $\ell_4 = 26$  have to be produced with quantities  $b_1 = 30$ ,  $b_2 = 30$ ,  $b_3 = 30$ , and  $b_4 = 120$ , respectively. We limit ourselves to the patterns  $a^1, a^2, \dots, a^{11}$  whose material utilization is at least 80%. These patterns define index set  $J = \{1, \dots, 11\}$  and matrix  $A$ :

| $L = 70$      | cutting patterns |       |       |       |       |       |       |       |       |          |          |
|---------------|------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|
|               | $a^1$            | $a^2$ | $a^3$ | $a^4$ | $a^5$ | $a^6$ | $a^7$ | $a^8$ | $a^9$ | $a^{10}$ | $a^{11}$ |
| $\ell_1 = 20$ | 1                | 0     | 0     | 1     | 1     | 2     | 0     | 2     | 1     | 2        | 3        |
| $\ell_2 = 22$ | 0                | 2     | 2     | 1     | 1     | 0     | 3     | 0     | 2     | 1        | 0        |
| $\ell_3 = 25$ | 2                | 0     | 1     | 0     | 1     | 0     | 0     | 1     | 0     | 0        | 0        |
| $\ell_4 = 26$ | 0                | 1     | 0     | 1     | 0     | 1     | 0     | 0     | 0     | 0        | 0        |

$\overbrace{\hspace{10em}}^{\mathbf{= A}}$

We continue this example after describing a solution method in the next section. ■

As already mentioned above, in order to get an, in general, approximate solution of the cutting stock problem of form (4.4), we exploit information provided by a solution of the corresponding continuous relaxation (4.6). Then, a feasible, and hence, integer solution of the cutting stock problem can be obtained by simply rounding up the continuous solution. Simple, but more appropriate heuristics are proposed in Sect. 4.4.

Prior to that, we describe how to solve the LP relaxation (4.6). The approach is based on the (*revised*) simplex method which is applicable to solve linear optimization problems with equality constraints. Because of the, in general, huge number of patterns, we apply *column generation*.

In order to transform problem (4.6) to an equivalent problem with equality constraints, we introduce nonnegative *slack variables*  $s_i$ ,  $i \in I$ . Let  $s = (s_1, \dots, s_m)^\top$ . We obtain a problem of form

$$z_{LP} = \min e^\top x \quad \text{s.t.} \quad Ax - I_m s = b, \quad x \in \mathbb{R}_+^n, \quad s \in \mathbb{R}_+^m, \quad (4.7)$$

where  $I_m$  denotes the identity matrix of dimension  $m$ .

## 4.3 Solving the Continuous Relaxation

First of all, we give a short description of the revised simplex method. Then a possibility to control the termination of the solution process is proposed—the Farley bound. Finally, we explain the principle of column generation by means of the 1CSP.

### 4.3.1 The Revised Simplex Method

The *revised simplex method* is a variant of the (primal) simplex method for solving *linear optimization problems* firstly proposed by Dantzig and Wolfe [10]. Let us consider the linear minimization problem with equality constraints

$$z = \min c^\top x \quad \text{s.t.} \quad Ax = b, x \geq 0 \quad (4.8)$$

where  $c \in \mathbb{R}^n$ ,  $b \in \mathbb{R}^m$ , and  $A = (a^j)_{j \in J} \in \mathbb{R}^{m \times n}$  are input parameters. Without loss of generality, we assume  $b \geq 0$ . Obviously,  $x \in \mathbb{R}^n$  is the vector of unknowns. In order to simplify the description, we assume that matrix  $A$  has rank  $m$ , i.e., the rows of  $A$  are linearly independent. In the cutting stock problems, considered here, this assumption is always satisfied. It is easy to see, problem (4.7) has the form (4.8) with a coefficient matrix  $(A| -I_m)$  and variables  $(x, s)$ .

Due to the full rank of  $A$  there exist  $m$  linearly independent column vectors of  $A$  which we combine to the square matrix  $A_B$ .  $A_B$  is called *basis matrix* since its columns define a basis of  $\mathbb{R}^m$ . Thus, matrix  $A_B$  is *regular (non-singular)* and its *inverse* (matrix)  $A_B^{-1}$  exists. We group the columns of  $A$  which are not in  $A_B$  together in the matrix  $A_N \in \mathbb{R}^{m \times (n-m)}$ .  $A_N$  is called *non-basis matrix*. Corresponding to the columns we define vectors  $x_B, c_B \in \mathbb{R}^m$  and  $x_N, c_N \in \mathbb{R}^{n-m}$  so that we can rewrite problem (4.8) as follows:

$$\begin{aligned} z &= \min c^\top x = \min c_B^\top x_B + c_N^\top x_N \quad \text{s.t.} \\ Ax &= A_B x_B + A_N x_N = b, \quad x_B \geq 0, x_N \geq 0. \end{aligned} \quad (4.9)$$

Vector  $x_B$  contains all variables related to  $A_B$  which, therefore, are named *basis variables*. The variables in  $x_N$  are called *non-basis variables*. Since  $A_B$  has an inverse, we obtain

$$x_B = -A_B^{-1} A_N x_N + A_B^{-1} b \quad (4.10)$$

from (4.9). In this form, we see that the non-basis variables can be chosen, in fact, arbitrarily (we only have to regard the nonnegativity conditions) while the basis variables depend on  $x_N$ . Replacing  $x_B$  in the objective function by (4.10) we obtain

$$z = (c_N^\top - c_B^\top A_B^{-1} A_N) x_N + c_B^\top A_B^{-1} b.$$

If  $A_B^{-1}b \geq 0$  holds in (4.10), then we easily get a feasible solution  $x$  of problem (4.8) by setting  $x_N := 0$ :

$$x \leftrightarrow \begin{pmatrix} x_B \\ x_N \end{pmatrix} := \begin{pmatrix} A_B^{-1}b \\ 0 \end{pmatrix}.$$

**Theorem 4.1 (Optimality Criterion)** *Let  $A_B$  be a basis matrix of problem (4.8) and  $x_B$  a corresponding vector of basis variables. If  $A_B^{-1}b \geq 0$  and if  $c_N^\top - c_B^\top A_B^{-1}A_N \geq 0$ , then*

$$\bar{x} \leftrightarrow \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix} = \begin{pmatrix} A_B^{-1}b \\ 0 \end{pmatrix} \text{ is a solution of (4.8).}$$

If matrix  $A_B$  meets the assumptions of the theorem, then it is called *optimal basis matrix*.

*Proof*  $\bar{x} \leftrightarrow \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix}$  is a feasible solution since  $A\bar{x} = A_B \cdot A_B^{-1}b + A_N \cdot 0 = b$  and  $\bar{x} \geq 0$ . For each feasible solution  $x \leftrightarrow \begin{pmatrix} x_B \\ x_N \end{pmatrix}$  with  $x_B = -A_B^{-1}A_Nx_N + A_B^{-1}b$  and  $x_N \geq 0$ , we have:

$$z(x) = c_B^\top x_B + c_N^\top x_N = c_B^\top A_B^{-1}b + (c_N^\top - c_B^\top A_B^{-1}A_N)x_N \geq c_B^\top A_B^{-1}b = z(\bar{x})$$

because of  $c_N^\top - c_B^\top A_B^{-1}A_N \geq 0$ . Hence,  $\bar{x}$  is a feasible solution with minimal objective value. ■

In order to describe the revised simplex method, let  $a^j$  denote the  $j$ th column of  $A$ ,  $J = \{1, \dots, n\}$  an index set of all columns as well as  $B$  and  $N$  the respective index sets of  $A_B$  and  $A_N$ .

Since the revised simplex method is a variant of the *primal (standard) simplex method*, the relation  $A_B^{-1}b \geq 0$  is always satisfied. To test a chosen basis matrix  $A_B$  for optimality, we need to compute the minimal *reduced cost coefficient*

$$\bar{c} := \min\{c_j - d^\top a^j : j \in N\} \quad \text{with} \quad d^\top := c_B^\top A_B^{-1} \quad (4.11)$$

where  $d$  denotes the vector of *simplex multipliers*. If  $\bar{c} \geq 0$ , then  $A_B$  defines an optimal solution, otherwise a new column is determined which has to be inserted into the basis matrix according to the simplex rules.

To describe a general step of the *Revised Simplex Method*, summarized in Fig. 4.1, we assume that an initial feasible solution is available which is determined by  $A_B$ , that means, index sets  $B$  and  $N$  are known and  $A_B^{-1}b \geq 0$  holds in (4.9).

It is to notice that a termination due to step (4) cannot occur for cutting or packing problems provided a meaningful model is applied.

Various approaches can be used to implement the method. To obtain the inverse of the changed basis matrix the *formula of Sherman and Morrison* can be applied

**General Step of the Revised Simplex Method**

Input:  $B, N, A_B, A_N, c_B, c_n, b$

Output:  $\bar{B}, \bar{N}$

- (1) Compute  $d^\top = c_B^\top A_B^{-1}$  and  $\bar{c} := c_\tau - d^\top a^\tau := \min\{c_j - d^\top a^j : j \in N\}$ .
- (2) If  $\bar{c} \geq 0$ , then stop: a solution is found.
- (3) Compute the transformed pivot-column  $\bar{a} := -A_B^{-1} a^\tau$ .
- (4) If  $\bar{a}_i \geq 0$  for all  $i \in I$ , then stop:  $z$  is unbounded.
- (5) Compute the transformed right-hand side  $\bar{b} := A_B^{-1} b$ .
- (6) Identify a pivot-row  $\sigma$  according to  $\frac{\bar{b}_\sigma}{-\bar{a}_\sigma} := \min \left\{ \frac{\bar{b}_i}{-\bar{a}_i} : \bar{a}_i < 0, i \in I \right\}$ .
- (7) Set  $\bar{B} := (B \setminus \{\tau\}) \cup \{\sigma\}$ ,  $\bar{N} := (N \setminus \{\sigma\}) \cup \{\tau\}$  and update  $A_B, A_N, x_B, x_N, c_B$ , and  $c_N$ , respectively.

**Fig. 4.1** General step of the Revised Simplex Method

(cf., for instance, [24]). However, the computation of an inverse is avoided, in general. In efficient implementations of the simplex method, systems of linear equations are solved instead. We refer the interested reader to the relevant literature. Here, we only propose a clear schema to describe the method which allows us some compact computations. We call it the *revised simplex tableaux*.

| $x_B$ | $A_B^{-1}$                   | $\bar{b} = A_B^{-1} b$ | $\bar{a} = -A_B^{-1} a^\tau$       |
|-------|------------------------------|------------------------|------------------------------------|
|       | $d^\top = c_B^\top A_B^{-1}$ | $z_0 = d^\top b$       | $\bar{c} = c_\tau - d^\top a^\tau$ |

A *basis exchange* with pivot  $\bar{a}_\sigma = [-A_B^{-1} a^\tau]_\sigma$ , i.e., determining  $x_\tau$  by dissolving equation  $\sigma$  and replacing  $x_\tau$  within the other equations, leads to the transformed entries and therefore, the input data for the next step, if still necessary. Such a basis exchange can be formulated as follows: we identify all entries of  $A_B^{-1}$ ,  $\bar{b}$ ,  $d^\top$ , and  $z_0$  with the elements of matrix  $D = (d_{ij})$  with  $D = \begin{pmatrix} A_B & \bar{b} \\ d^\top & z_0 \end{pmatrix}$ . Then, all updated values of  $A_B^{-1}$ ,  $\bar{b}$ ,  $d^\top$ , and  $z_0$  can be obtained according to

**Basis exchange with pivot-row  $\sigma$** 

Input:  $D, \bar{a}, \sigma$

Output:  $\bar{D}$

- (1)  $\bar{D}_{\sigma j} := -D_{\sigma j}/\bar{a}_\sigma, \quad j = 1, \dots, m + 1,$
- (2)  $\bar{D}_{ij} := D_{ij} + \bar{a}_i \bar{D}_{\sigma j}, \quad i, j = 1, \dots, m + 1, i \neq \sigma.$

**Remark 4.4** Besides the efficient computation of the updates of  $\bar{b}$ ,  $d$ ,  $\bar{c}$  and  $\bar{a}$ , further aspects have to be taken into account during the solution process. On the one side, the increase of rounding errors has to be controlled, and on the other side, cycles

have to be excluded. To avoid cycling in the simplex method, the *pivot-rule of Bland* [31] can be applied.

*Example 4.2 (Continuation of Example 4.1)* We consider again the cutting stock problem of Example 4.1 with input data  $m = 4$ ,  $\ell = (20, 22, 25, 26)^\top$ ,  $b = (30, 30, 30, 120)^\top$ , and  $L = 70$ . Due to the 80% requirement of material utilization, we got the patterns  $a^1, \dots, a^{11}$ . These patterns form matrix  $A$  of the optimization problem

$$z = \min c^\top x \quad \text{s.t.} \quad Ax \geq b, \quad x \geq 0, \quad \text{integer.}$$

Neglecting the integer constraint and introducing slack variables  $s_i$ ,  $i \in I$ , we obtain the LP problem

$$z = \min c^\top x \quad \text{s.t.} \quad Ax - I_m s = b, \quad x \geq 0, \quad s \geq 0,$$

where  $c = (1, \dots, 1)^\top \in \mathbb{R}^{11}$  and  $I_m$  is the identity matrix (Table 4.1).

Matrix  $A_B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 \\ 2 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$  defines a first feasible solution: 30 times  $a^1$  and

120 times  $a^2$ . Only waste-less patterns are used. Due to the slack variables  $s_2$  and  $s_3$ , since we have overproduction for piece types 2 and 3,  $x_B = (x_1, s_2, s_3, x_2)^\top$  is the vector of basis variables and  $c_B = (1, 0, 0, 1)^\top$  the respective vector of objective coefficients. Summarizing, we obtain (revised simplex) Tableau  $ST_1$  (depicted with the supplementary column  $-A_B^{-1}a^6$ ):

| $ST_1$   | $A_B^{-1}$ |    |    | $A_B^{-1}b$ | $-A_B^{-1}d^6$   |
|----------|------------|----|----|-------------|------------------|
| $x_1$    | 1          | 0  | 0  | 0           | 30               |
| $s_2$    | 0          | -1 | 0  | 2           | 210              |
| $s_3$    | 2          | 0  | -1 | 0           | 30               |
| $x_2$    | 0          | 0  | 0  | 1           | 120              |
| $d^\top$ | 1          | 0  | 0  | 1           | 150              |
|          |            |    |    |             | $\bar{c}_6 = -2$ |

According to the optimality criterion, we have to verify whether a column  $a^\tau$  in  $A$  or  $-I_m$  exists with  $\bar{c}_\tau := c_\tau - d^\top a^\tau < 0$  which is not contained in  $A_B$ , or equivalently, for which  $d^\top a^\tau > c_\tau$  holds. For the non-basis patterns, we obtain the following

**Table 4.1** Input data of Example 4.2

reduced (cost) coefficients of the objective function:  $\bar{c}_3 = \bar{c}_7 = 1$ ,  $\bar{c}_4 = \bar{c}_8 = \bar{c}_{10} = -1$ ,  $\bar{c}_5 = \bar{c}_9 = 0$ ,  $\bar{c}_6 = \bar{c}_{11} = -2$ . The reduced coefficients of the slack variables  $s_1$  and  $s_4$  are both 1. Therefore, column  $a^6 = (2, 0, 0, 1)^\top$  is one with minimal reduced coefficient  $\bar{c}_6 = -2$ , i.e.  $\tau = 6$ . Next, we compute vector  $\bar{a} = -A_B^{-1}a^6$  (see Tableau  $ST_1$ ) and obtain the third row as pivot-row. Performing the basis exchange, Tableau  $ST_2$  results to:

| $ST_2$   | $A_B^{-1}$ |    |           | $A_B^{-1}b$ |
|----------|------------|----|-----------|-------------|
| $x_1$    | 0          | 0  | $1/2\ 0$  | 15          |
| $s_2$    | -1         | -1 | $1/2\ 2$  | 195         |
| $x_6$    | $1/2$      | 0  | $-1/4\ 0$ | $15/2$      |
| $x_2$    | $-1/2$     | 0  | $1/4\ 1$  | $225/2$     |
| $d^\top$ | 0          | 0  | $1/2\ 1$  | 135         |

with  $A_B = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & -1 & 0 & 2 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$

In this way, we get all updated terms  $A_B^{-1}$ ,  $A_B^{-1}b$ ,  $d$ , and  $z_0$ , which have to be recomputed after the exchange of columns  $-e^3$  and  $a^6$  in  $A_B$ . The resulting objective value  $z = 135$  cannot be decreased since now all reduced coefficients are nonnegative. Applying an appropriate rounding, we obtain the feasible solution

$$15 \text{ times } a^1, \quad 112 \text{ times } a^2, \quad \text{and eight times } a^6.$$

Hence, 135 pieces of raw material are needed. The degree of material utilization is only 54.29%. Altogether, 31 pieces of length  $\ell_1$ , 224 pieces of length  $\ell_2$ , 30 pieces of length  $\ell_3$  and 120 pieces of length  $\ell_4$  are cut. Although only patterns with small waste are contained in the model, a high percentage of total waste arises. This is mainly caused by the a priori restriction on a subset of patterns. We will see in the following that all order demands can be satisfied with only 87 pieces of raw material. ■

As main consequence which we can draw based on the example: the consideration of only a subset of patterns can lead to a drastic increase of the optimal value, and therefore, should be avoided if possible. Otherwise, one needs to estimate the worsening, but this will be a hard job in general. As we will see in Sect. 4.3.3, a restriction on a subset of patterns is mostly not necessary.

### 4.3.2 The Farley Bound

Due to a very large number of variables, as for cutting stock problems, one has to regard the occurrence of rounding errors when computing  $\bar{c}$  according to (4.11). Rounding errors can result, in particular, because of the repeated update of the inverse of the basis matrix. To avoid unnecessary simplex steps, since the optimality of a current solution is not proved correctly, an approach proposed by Farley [14] can be helpful.

**Proposition 4.2** Let  $z^*$  be the optimal value of the linear minimization problem (4.8), and let  $x^0 = (x_B^0, x_N^0)^\top$  denote a feasible solution of (4.8) and  $d \in \mathbb{R}^m$  the corresponding vector of simplex multipliers. If  $c_j > 0$  for all  $j \in J$  and  $\tilde{c} := \max\{d^\top a^j / c_j : j \in J\}$ , then we have

$$z^* \geq d^\top b / \tilde{c}.$$

*Proof* Since  $x^0$  is a basis solution of (4.8), we have the representation of its objective value

$$z(x) = z_0 + \sum_{j \in J} (c_j - d^\top a^j)x_j \quad \text{with } d^\top = c_B^\top A_B^{-1} \text{ and } z_0 = d^\top b = z(x^0)$$

for all  $x$  with  $Ax = b$  where matrix  $A_B$  contains the respective columns  $a^j$  of  $A$  related to the basis variables  $x_B$ . In particular, for an optimal solution  $x^*$  of (4.8) the relation

$$z(x^*) = z^* = z_0 + \sum_{j \in J} c_j \left(1 - \frac{d^\top a^j}{c_j}\right)x_j^* \geq z_0 + \sum_{j \in J} c_j (1 - \tilde{c})x_j^* = z_0 + (1 - \tilde{c})z^*$$

holds. Obviously, we have  $\tilde{c} \geq 1$  since  $d^\top a^j = c_j$  for all basis variables. Consequently, we obtain  $z^* \geq z_0 / \tilde{c}$ . ■

Proposition 4.2 can be motivated as well as follows. The optimization problem

$$\max b^\top u \quad \text{s.t.} \quad A^\top u \leq c, \quad u \in \mathbb{R}^m, \quad (4.12)$$

is the *dual problem* of (4.8), and vector  $u^0 := d / \tilde{c}$  is a feasible solution of (4.12) since  $(u^0)^\top a^j = d^\top a^j / \tilde{c} \leq c_j$  holds for all  $j \in J$ . Caused by the duality theory of linear programming, it is known that the function value  $b^\top u^0$  of any feasible solution of (4.12) defines a lower bound for  $z^*$ .

If we only assume  $c_j \geq 0$  for all  $j \in J$  in Proposition 4.2, then we obtain the same statement with  $\tilde{c} := \max\{d^\top a^j / c_j : j \in J, c_j > 0\}$  provided  $d^\top a^j \leq 0$  holds for all  $j \in J$  with  $c_j = 0$ .

The lower bound of the optimal value given in Proposition 4.2 is especially applicable for solving the continuous relaxation of models (4.4) and (4.5) since  $c_j = 1$  holds for all patterns. Hence, if the relation  $\lceil z_0 \rceil = \lceil z_0 / \tilde{c} \rceil$  is true for a basis solution  $x^0$  with objective value  $z_0$ , then  $\lceil z_0 \rceil = \lceil z^* \rceil$  is implied.

### 4.3.3 Column Generation

We explain the principle of *column generation* by means of the one-dimensional cutting stock problem. But the applicability of this technique is much wider and we will point out this in respective areas.

We consider again the problem to optimally cut pieces of lengths  $\ell_i, i \in I = \{1, \dots, m\}$ , out of raw material of length  $L$ . Every column  $a = (a_1, \dots, a_m)^\top$  of  $A$  which represents a feasible pattern satisfies the following conditions:

$$\sum_{i=1}^m \ell_i a_i \leq L, \quad a_i \geq 0, \quad \text{integer}, \quad i \in I.$$

The computation of a pattern having minimal reduced objective coefficient, i.e., the verification of optimality or the identification of a pivot column, is equivalent to solve the optimization problem:

*Determine nonnegative integers  $a_i, i \in I$ , which satisfy condition  $\sum_{i=1}^m \ell_i a_i \leq L$  and minimize the reduced objective coefficient  $\bar{c} = 1 - \sum_{i \in I} d_i a_i$ .*

(Observe that  $c_j = 1$  holds for all patterns and see Exercise 4.4.) It is obvious that this problem is equivalent to the knapsack problem

$$w = \max \sum_{i=1}^m d_i a_i \quad \text{s.t.} \quad \sum_{i=1}^m \ell_i a_i \leq L, \quad a_i \in \mathbb{Z}_+, \quad i \in I. \quad (4.13)$$

It can be solved efficiently by appropriate algorithms as addressed in Chap. 2. Hence, the technique of column generation consists in the solution of a *column generation problem* in order to compute the minimal reduced objective coefficient according to (4.11). In case of the 1CSP this leads to a knapsack problem of form (4.13).

The applicability of the column generation technique arises when the columns of matrix  $A$  possess a certain structure which enables to formulate a respective mathematical model. In the context of cutting and packing problems, this depends on the opportunity to characterize a pattern by formulas.

*Remark 4.5* Frequently, problem (4.8) is called the *master problem*, and the column generation problem the *slave problem*. We will use these notations as well.

The principle of column generation to solve an LP problem of form

$$z = \min \left\{ \sum_{j \in J} c_j x_j : \sum_{j \in J} a_j^j x_j = b, \quad x_j \geq 0, \quad j \in J \right\}$$

is summarized in Fig. 4.2. For the sake of simplicity, we assume that the problem is solvable, and that  $\Gamma$  denotes an oracle (procedure) which provides a solution of the slave problem.

To illustrate the technique of column generation, we consider once more the one-dimensional cutting stock problem of Example 4.2.

*Example 4.3 (Continuation of Example 4.2)* We have  $L = 70$  and  $\ell = (20, 22, 25, 26)^\top$ . As initial basis matrix  $A_B$ , we choose a diagonal matrix whose columns represent elementary patterns  $a^j = (a_{1j}, \dots, a_{mj})^\top$  ( $m = 4$ ) with

**Column generation for solving problem (4.8)**

Input:  $c, b, \Gamma$

Output:  $x^c, z_c$

- (1) Let  $k := 0$ .
  - (2) Choose a subset  $\{a^j : j \in J_0\}$  of the columns such that
- $$(P_k) : \min\left\{\sum_{j \in J_k} c_j x_j : \sum_{j \in J_k} a^j x_j = b, x_j \geq 0, j \in J_k\right\}$$
- is solvable.
- (3) Solve the restricted master problem  $(P_k)$ . Let  $x^k$  denote a solution,  $z_k$  its value, and  $d^k$  the vector of simplex multipliers.
  - (4) Solve the slave problem
- $$\bar{c} = \min\{c_j - (d^k)^\top a^j : j \in J\}$$
- by means of oracle  $\Gamma$ . Let  $\bar{c} = c_\tau - (d^k)^\top a^\tau$  for some  $\tau \in J$ .
- (5) If  $\bar{c} \geq 0$ , then  $x^c := x^k$  is a solution of (4.8) with value  $z_c = z_k - \bar{c}$  – stop.
  - (6) Set  $k := k + 1, J_k := J_{k-1} \cup \{\tau\}$  and go to (3).

**Fig. 4.2** Column generation for solving problem (4.8)

$a_{jj} = \lfloor L/\ell_j \rfloor$  and  $a_{ij} = 0$  for  $i \neq j, j = 1, \dots, m$ , i.e., every pattern contains a single piece type as often as possible:

$$a^1 = (3, 0, 0, 0)^\top, a^2 = (0, 3, 0, 0)^\top, a^3 = (0, 0, 2, 0)^\top, a^4 = (0, 0, 0, 2)^\top.$$

Due to this choice the basis inverse  $A_B^{-1}$  is immediately available: Herewith, we obtain Tableau  $ST_1$  (initially without column  $-A_B^{-1}a^5$ ):

| $ST_1$   | $A_B^{-1}$      | $A_B^{-1}b$ | $-A_B^{-1}a^5$ |  |
|----------|-----------------|-------------|----------------|--|
| $x_1$    | 1/3             | 10          | -1/3           |  |
| $x_2$    | 1/3             | 10          | 0              |  |
| $x_3$    | 1/2             | 15          | -1             |  |
| $x_4$    | 1/2             | 60          | 0              |  |
| $d^\top$ | 1/3 1/3 1/2 1/2 | 95          | -1/3           |  |

with  $A_B = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$ .

Now we have to check whether the chosen basis matrix provides an optimal solution or, respectively, whether a pattern exists which can lead to an improved feasible solution. For that purpose, we solve the knapsack problem (4.13)

$$w = \max \left\{ \frac{1}{3} a_1 + \frac{1}{3} a_2 + \frac{1}{2} a_3 + \frac{1}{2} a_4 : a \in G \right\}.$$

where

$$G := \{a = (a_1, \dots, a_4)^\top \in \mathbb{Z}_+^4 : 20a_1 + 22a_2 + 25a_3 + 26a_4 \leq 70\}$$

denotes the set of all feasible patterns. Because of  $d_1 \geq d_2$  and  $\ell_1 < \ell_2$ , we can choose  $a_2 = 0$  (because of dominance), and since  $d_3 \geq d_4$  and  $\ell_3 < \ell_4$ , we can set  $a_4 := 0$ . Therefore, the knapsack problem reduces to

$$w = \max \left\{ \frac{1}{6} (2a_1 + 3a_3) : 20a_1 + 25a_3 \leq 70, a_1, a_3 \in \mathbb{Z}_+ \right\}.$$

Obviously,  $a_1 = 1, a_3 = 2$  is a solution with optimal value  $4/3 > 1$ . Since  $4/3 > 1$ , the associated pattern  $a^5 = (1, 0, 2, 0)^\top$  has to be inserted into the basis matrix. After computing  $\bar{a} = -A_B^{-1}a^5$  and  $\bar{c}_5 = -1/3$  (cf. Tableau  $ST_1$ ), we obtain Tableau  $ST_2$  according to the simplex rules:

| $ST_2$   | $A_B^{-1}$ |      | $A_B^{-1}b$ | $-A_B^{-1}a^6$ |
|----------|------------|------|-------------|----------------|
| $x_1$    | 1/3        | -1/6 | 5           | -2/3           |
| $x_2$    |            | 1/3  | 10          | 0              |
| $x_5$    |            | 1/2  | 15          | 0              |
| $x_4$    |            | 1/2  | 60          | -1/2           |
| $d^\top$ | 1/3        | 1/3  | 1/3         | 1/2            |
|          | 90         |      |             | -1/6           |

$$\text{with } A_B = \begin{pmatrix} 3 & 0 & 1 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

The verification of optimality leads to the knapsack problem

$$w = \max \left\{ \frac{1}{3} a_1 + \frac{1}{3} a_2 + \frac{1}{3} a_3 + \frac{1}{2} a_4 : a \in G \right\}.$$

We obtain a further pattern  $a^6 = (2, 0, 0, 1)^\top$  and with it Tableau  $ST_3$ :

| $ST_3$   | $A_B^{-1}$ |      | $A_B^{-1}b$ | $-A_B^{-1}a^7$ |
|----------|------------|------|-------------|----------------|
| $x_6$    | 1/2        | -1/4 | 15/2        | 0              |
| $x_2$    |            | 1/3  | 10          | -2/3           |
| $x_5$    |            | 1/2  | 15          | 0              |
| $x_4$    | -1/4       | 1/8  | 1/2         | 225/4          |
| $d^\top$ | 1/4        | 1/3  | 3/8         | 1/2            |
|          | 355/4      |      |             | -1/6           |

$$\text{with } A_B = \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}.$$

Now, the corresponding knapsack problem is

$$w = \max \left\{ \frac{1}{24} (6a_1 + 8a_2 + 9a_3 + 12a_4) : a \in G \right\}.$$

From it, we obtain pattern  $a^7 = (0, 2, 0, 1)^\top$  and Tableau  $ST_4$ .

| $ST_4$   | $A_B^{-1}$ |      |         | $A_B^{-1}b$ |
|----------|------------|------|---------|-------------|
| $x_6$    | 1/2        | -1/4 |         | 15/2        |
| $x_7$    |            | 1/2  |         | 15          |
| $x_5$    |            | 1/2  |         | 15          |
| $x_4$    | -1/4       | -1/4 | 1/8 1/2 | 195/4       |
| $d^\top$ | 1/4        | 1/4  | 3/8 1/2 | 345/4       |

$$\text{with } A_B = \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 1 & 0 & 2 \end{pmatrix}.$$

The knapsack problem

$$w = \max \left\{ \frac{1}{8} (2a_1 + 2a_2 + 3a_3 + 4a_4) : a \in G \right\},$$

resulting from Tableau  $ST_4$ , has optimal value  $w = 1$ . Hence, there does not exist any pattern which can improve the current solution.

In analogy to the patterns, we need to check whether one of the columns  $-e^i$  ( $i \in I$ ), belonging to the slack variables, has to be inserted into the basis matrix. We obtain the corresponding reduced objective coefficient  $\bar{c}_i$  according to  $\bar{c}_i = 0 - d^\top(-e^i) = d^\top e^i = d_i$ ,  $i \in I$ . Since  $d_i \geq 0$  holds for all  $i \in I$ , Tableau  $ST_4$  is optimal, and we can extract a solution of the continuous relaxation of the 1CSP:

$$x_4 = 48.75, x_5 = 15, x_6 = 7.5, x_7 = 15, \quad \text{and} \quad z_{LP}^* = 86.25.$$

Hence, at least 87 rods have to be cut. Rounding up the above solution, we obtain an integer solution with function value 87. Hence, a solution of the cutting stock problem is found. We have to cut 49 rods according to pattern  $a^4 = (0, 0, 0, 2)^\top$ , 15 times  $a^5 = (1, 0, 2, 0)^\top$ , 8 times  $a^6 = (2, 0, 0, 1)^\top$ , and 15 times  $a^7 = (0, 2, 0, 1)^\top$ . Now, the material utilization is equal to 84.24%. The improvement in comparison to that in Example 4.2 is caused by the usage of pattern  $a^4$  with only 74.29% material utilization. This pattern is considered since the column generation technique incorporates, in principle, all feasible patterns. ■

We point out that a simple rounding up of the LP solution does not lead, in general, to a solution or good approximate solution for the integer problem. Several, more appropriate heuristic approaches are described in the subsequent section.

As already mentioned, the applicability of the column generation technique is not restricted to the one-dimensional cutting stock problem. Column generation (i.e. slave) problems which arise during the solution of two-dimensional cutting stock problems are addressed in Chap. 6. In that cases, the simplex multipliers define again the piece valuations.

In order to avoid a regeneration of columns (patterns) previously removed from the basis in some simplex step, a *pool* (a subset) of feasible patterns is initially considered as the master problem. This reduced problem is solved to optimality with the simplex method. Then, by means of a slave problem, it is checked whether

the current pool provides an solution of the entire problem. If not, one or several new columns are obtained which are added to the pool. The newly obtained pool defines the next master problem which has to be solved, and so on, until the current reduced master problem yields a solution for the original problem.

Summarizing, we have the important advantage of the column generation technique: only a restricted number of patterns (columns) are contained within the column pool, but all feasible patterns are regarded by means of the slave problem. Since the effort to solve the slave problems directly determines the total run-time, the mentioned advantage becomes less important the larger the effort per slave problem is.

## 4.4 Getting Integer Solutions, Heuristics

Since the 1CSP (as a variant of the 1BPP) belongs to the NP-hard optimization problems [15], various heuristic approaches are developed. We address here, in particular, such which are based on an LP solution.

Of course, any BPP heuristic, described in Sect. 3.4, can be used to get a feasible solution of the 1CSP, for instance, the Next Fit, First Fit, or Best Fit heuristic. Since these heuristics are very fast, they can be used in *meta-heuristics* (see Sect. 7.4.2). Related performance results, such as  $\rho_\infty(\text{NF}) = 2$ ,  $\rho(\text{FF}) = \rho(\text{BF}) = 1.75$ , and  $\rho_\infty(\text{FF}) = \rho_\infty(\text{BF}) = 1.7$  are known, see Sect. 3.5, implying that their performance is bounded by the same ratio when applied in an *online* scenario.

Improved fast heuristics are obtained if the pieces are sorted according to non-increasing lengths in the beginning. Both, the First Fit Decreasing and the Best Fit Decreasing heuristic run in  $O(n \log n)$  time, where  $n$  is the total number of items to be cut, with performance ratios  $\rho(\text{FFD}) = \rho(\text{BFD}) = 3/2$  [32] and  $\rho_\infty(\text{FFD}) = \rho_\infty(\text{BFD}) = 11/9$  [2].

A more expensive heuristic for the 1CSP applies a series of knapsack problems. Let  $E = (m, \ell, L, b)$  be an instance of the 1CSP, and let  $\tilde{b}^k$  denote the remaining order demands after cutting  $k$  patterns with appropriate frequencies. Of course,  $\tilde{b}^0 = b$  in the beginning. Then a solution  $a^*$  of the knapsack problem

$$\max\left\{\sum_{i \in I} \ell_i a_i : \sum_{i \in I} \ell_i a_i \leq L, a_i \leq \tilde{b}_i^{k-1}, i \in I\right\}$$

defines pattern  $a^k := a^*$  to be cut next with frequency

$$x_k := \min\{\lfloor \tilde{b}_i^{k-1} / a_i^k \rfloor : a_i^k > 0, i \in I\}.$$

Then,  $\tilde{b}^k := \tilde{b}^{k-1} - a^k x_k$ , and the process is continued until  $\tilde{b}^k = 0$ . For such an approach it is characteristic that in the beginning patterns with high material usage are computed whereas in the sequel this degree decreases together with small

frequencies of application of patterns. Therefore, it is more advantageous to use other profit coefficients in the knapsack problems as, for instance, in the *sequential value correction method* (see Sect. 7.4.2).

Feasible integer solutions of the 1CSP can also be obtained based on solutions of the LP relaxation of, e.g., the standard (pattern-oriented) model. We consider an instance  $E = (m, \ell, L, b)$  of the 1CSP. Let  $a^j, j \in J$  denote all feasible maximal patterns which define the pattern matrix  $A \in \mathbb{Z}_+^{m \times n}$  where  $n = |J|$ . Furthermore, let  $x^c \in \mathbb{R}_+^n$  be a solution of the continuous relaxation (4.6) of the Gilmore/Gomory model, i.e., we have  $Ax^c \geq b$ . Then, at most  $m$  components of  $x^c$  are nonzero. Let  $J^c := \{j \in J : x_j^c > 0\}$ . The function value  $z(x^c) = \sum_{j \in J^c} x_j^c$  provides a lower bound, called *LP bound*, for the optimal value  $z^*$  of the integer problem.

It is easy to see that the relations

$$z(\lfloor x^c \rfloor) \leq \lceil z(x^c) \rceil \leq z^* \leq z(\lceil x^c \rceil) \leq z(\lfloor x^c \rfloor) + m$$

hold due to rounding up  $x^c$ . Let  $\bar{x} := \lceil x^c \rceil$  and  $\underline{x} := \lfloor x^c \rfloor$ . Obviously,  $\bar{x}$  is a feasible solution of (4.3) with some overproduction  $\bar{b} := A\bar{x} - b \geq 0$ , whereas  $\underline{x}$  is, in general, infeasible because of some underproduction  $\underline{b} := b - A\underline{x} \geq 0$ . Based on these observations, various opportunities exist to construct feasible integer solutions.

On the one hand, we can try to reduce the overproduction by solving the problem

$$\max \sum_{j \in J^c} y_j \quad \text{s.t.} \quad \sum_{j \in J^c} a^j y_j \leq \bar{b}, \quad y_j \in \mathbb{Z}_+, \quad j \in J^c.$$

If its solution  $y^*$  is nonzero, then  $\bar{x}_j - y_j^*, j \in J^c$  provides a better feasible integer solution than  $\bar{x}$ . The determination of  $y^*$  can be done, if too expensive, only approximately by a heuristic (see, for instance, [33]).

On the other hand, we can extend  $\underline{x}$  to become a feasible integer solution of (4.3) by solving the *residual problem*

$$\min \sum_{j \in J} y_j \quad \text{s.t.} \quad \sum_{j \in J} a^j y_j \geq \underline{b}, \quad y_j \in \mathbb{Z}_+, \quad j \in J.$$

In this case, since  $\underline{b}$  is relatively small, in general, only *proper* patterns should be considered instead of all represented by  $J$ . Then again, an approach based on the respective continuous relaxation could be applied. Due to the, in general, small magnitude of the components of  $\underline{b}$  the residual problem is rather a bin packing problem. Therefore, methods to solve the 1BPP exactly or approximately should be applied.

Depending on  $z^* - z(\underline{x})$  and  $z(\bar{x}) - z^*$ , one can prefer the first or the second variant. In some cases the number of different patterns in the solution has to be regarded because of setup costs. In the first variant, this number is not increased in comparison to  $x^c$  whereas applying a BPP heuristic generates further, in general, different patterns which can increase setup costs.

The here proposed strategy to tackle one-dimensional cutting stock problems can be summarized as follows: compute a solution of the continuous relaxation and construct an integer solution from it. Although this is an approximate one, in the majority of cases this method leads to optimal solutions, or those which are very close to optimal ones. In particular, up to now there is not known any instance of the 1CSP for which the difference between the optimal values of the integer problem and its continuous relaxation is larger than 1.2. This issue is considered in Sect. 4.11 in more detail.

## 4.5 A Branch-and-Bound Approach

Since the cutting stock problem belongs to the class of NP-hard problems, branch-and-bound or branch-and-cut approaches have to be applied in order to get a proved solution, in general. Here, we propose a concept of a branch-and-bound (b&b) algorithm which is based on the pattern-oriented standard model of the 1CSP. Lower bounds are obtained by the respective LP relaxations which are solved by the column generation technique.

Let  $E = (m, \ell, L, b)$  be an instance of the 1CSP, and let  $a^j, j \in J$  denote all feasible patterns of  $E$ . With  $\Pi$  we denote the set of all *active* subproblems within the solution process. Initially,  $\Pi$  is set to contain the problem

$$(\pi_0) : \quad z_0 := \min\left\{ \sum_{j \in J} x_j : \sum_{j \in J} a^j x_j \geq b, x_j \in \mathbb{Z}_+, j \in J \right\}.$$

For branching we consider a bisection strategy (or alternative concept), i.e., a current (sub) problem (if not pruned) is partitioned into two successor problems. To this end, we define two index sets  $J^\geq$  and  $J^\leq$  indicating those patterns which have to have frequencies in the LP solution at least or at most as large as a predefined bound, respectively.

Let  $x^k$  be a solution of the LP relaxation of subproblem  $\pi_k \in \Pi$  where  $k$  counts the subproblems. Furthermore, let

$$j_k := \arg \max\{x_j^k : x_j^k \notin \mathbb{Z}, j \in J\}. \quad (4.14)$$

Using  $u_{j_k} := \lceil x_{j_k}^k \rceil$ , we define two successor problems as follows:

$$\begin{aligned} (\pi_{k,1}) &:= (\pi_k) \wedge (x_{j_k} \geq u_{j_k}), \quad (J^\geq := J^\geq \cup \{j_k\}), \\ (\pi_{k,2}) &:= (\pi_k) \wedge (x_{j_k} \leq u_{j_k} - 1), \quad (J^\leq := J^\leq \cup \{j_k\}). \end{aligned} \quad (4.15)$$

**Branch-and-bound algorithm for the 1CSP**

Input:  $E = (m, \ell, L, b)$

Output:  $x^*, z^*$

- (1) Initialize  $\Pi := \{\pi_0\}$  and set  $z^* := \infty$ .
- (2) If  $\Pi = \emptyset$ , then stop:  $x^*$  is a solution.
- (3) Select a subproblem  $\pi_k \in \Pi$  and set  $\Pi := \Pi \setminus \{\pi_k\}$ .
- (4) Compute a solution  $x^k$  of the LP relaxation of  $\pi_k$  with value  $z_k$ .
- (5) If  $\lceil z_k \rceil \geq z^*$ , then go to (3).
- (6) If  $x^k$  is integer, then set  $x^* := x^k, z^* := z_k$ , and go to (3).
- (7) Determine  $j_k$  according to (4.14).
- (8) Define successor problems  $\pi_{k,1}$  and  $\pi_{k,2}$  according to (4.15).
- (9) Set  $\Pi := \Pi \cup \{\pi_{k,1}, \pi_{k,2}\}$  and go to (3).

**Fig. 4.3** Branch-and-bound algorithm for the 1CSP

Thus, any subproblem, which results in the branching process, has the form

$$(\pi_k) : \quad z_k := \min \left\{ \sum_{j \in J} x_j : \sum_{j \in J} a^j x_j \geq b, \quad x_j \geq u_j, j \in J^{\geq}, \right. \\ \left. x_j \leq u_j - 1, j \in J^{\leq}, \quad x_j \in \mathbb{Z}_+, j \in J \right\}.$$

It is obvious, the schema depicted in Fig. 4.3 provides only the general principle of a b&b algorithm. Most of the steps allow modifications and refinements. For instance, after step (6), heuristic approaches can be applied to obtain feasible integer solutions based on  $x^k$  (cf. Sect. 4.4). We refer to [5] and [6] for more detailed descriptions and alternatives.

In difference to the slave problems of the LP relaxation of  $\pi_0$  which are simply common knapsack problems, for subproblems with  $J^{\leq} \neq \emptyset$ , within the column generation process, we have to solve

$$\bar{c} = \min \{c_j - d^\top a^j : j \in J \setminus J^{\leq}\}$$

where  $d$  denotes a respective vector of simplex multipliers. This causes some additional difficulties—how to prevent the solution from belonging to  $J^{\leq}$ ? A common method consists in replacing each integer variable by a sum of Boolean variables, since it is known that two binary vectors  $\alpha, \beta \in \mathbb{B}^q$  are different if and only if

$$\sum_{i: \beta_i=1} (1 - \alpha_i) + \sum_{i: \beta_i=0} \alpha_i \geq 1.$$

Since  $a_i \leq k_i := \lfloor L/\ell_i \rfloor$ , we define  $\tau_i := \lceil \log(k_i + 1) \rceil$  variables  $\alpha_{ip} \in \mathbb{B}$ ,  $p = 1, \dots, \tau_i$  so that  $a_i = \sum_{p \in K_i} 2^{p-1} \alpha_{ip}$  where  $K_i := \{1, \dots, \tau_i\}$ ,  $i \in I$ . Thus, instead of  $m$  integer variables  $a_i$ , now we have  $q = \sum_{i \in I} \tau_i$  binary variables, and the slave problem can be written as

$$\begin{aligned} \bar{c} = \min\{ & c_j - \sum_{i \in I} \sum_{p \in K_1} 2^{p-1} d_i \alpha_{ip} : \sum_{i \in I} \sum_{p \in K_1} 2^{p-1} \ell_i \alpha_{ip} \leq L, \\ & \sum_{i \in I} \sum_{p \in K_1: \beta_{ip}^j=1} (1 - \alpha_{ip}) + \sum_{i \in I} \sum_{p \in K_1: \beta_{ip}^j=0} \alpha_{ip} \geq 1, \quad j \in J^{\leq}, \\ & \alpha_{ip} \in \mathbb{B}, \quad p \in K_i, \quad i \in I \} \end{aligned}$$

where  $\beta^j = (\beta_{ip}^j)$  is the reformulation of  $a^j, j \in J^{\leq}$ .

## 4.6 Lower Bounds

We consider an instance  $E = (m, \ell, L, b)$  of the 1CSP, and let  $I := \{1, \dots, m\}$ . Clearly, the consumption of material is not less than  $\sum_{i \in I} \ell_i b_i$ . Therefore, at least

$$lb_M = lb_M(E) := \lceil \sum_{i \in I} \ell_i b_i / L \rceil$$

pieces of raw material are necessary to fulfill all order demands. We call  $lb_M$  the *material bound* of instance  $E$ .

**Proposition 4.3** *For any instance  $E = (m, \ell, L, b)$  of the 1CSP with integer input data, we have*

$$lb_M(E) \leq z^*(E) \leq 2 \cdot lb_M(E)$$

where  $z^*(E)$  denotes the optimal value of instance  $E$ .

*Proof* Exercise 4.19

In case of only smaller item lengths, we can obtain a tighter interval for  $z^*$ .

**Lemma 4.4** *Let  $E = (m, \ell, L, b)$  be an instance of the 1CSP with  $\sum_{i \in I} \ell_i b_i \geq L$ . Furthermore, let  $q \in \mathbb{N}$ . If  $\ell_i \leq L/q$  for all  $i \in I$ , then there exists a pattern  $a \in \mathbb{Z}_+^m$  with*

$$\frac{q}{q+1}L < \sum_{i \in I} \ell_i a_i \leq L.$$

The proof is similar to that of Lemma 3.2.

**Corollary 4.5** *If  $\ell_i \leq L/q$  for all  $i \in I$  and  $q \geq 2$ ,  $q \in \mathbb{N}$ , then we have*

$$lb_M(E) \leq z^*(E) \leq \frac{q+1}{q} lb_M(E).$$

Hence, the absolute worst-case performance ratio of lower bound  $lb_M$  is bounded by  $(q+1)/q$ .

Another natural lower bound results from the total number of demanded pieces. Let

$$k^* := \max\{\lfloor L/\ell_i \rfloor : i \in I\}$$

denote the maximal number of items in a pattern. Then, obviously,

$$lb_d(E) := \left\lceil \frac{1}{k^*} \sum_{i \in I} b_i \right\rceil$$

defines a lower bound of  $z^*(E)$  since not more than  $k^*$  desired pieces can be obtained from one stock piece. It is easy to see, neither  $lb_M$  dominates  $lb_d$  nor  $lb_d$  dominates  $lb_M$ .

Similar to a lower bound proposed for the two-dimensional BPP by Martello and Toth [19], we can define another lower bound which is oriented on the occurrence of larger item types as follows: let  $p \in [0, L/2] \cap \mathbb{Z}$ . We define index sets  $I_1(p)$ ,  $I_2(p)$  and  $I_3(p)$  according to

$$\begin{aligned} I_1(p) &:= \{i \in I : L - p < \ell_i\}, \\ I_2(p) &:= \{i \in I : L/2 < \ell_i \leq L - p\}, \\ I_3(p) &:= \{i \in I : p \leq \ell_i \leq L/2\}. \end{aligned}$$

Then the lower bound  $lb_{MT}$  for the 1CSP, related to that of Martello and Toth, is the following:

$$lb_{MT}(E, p) := \sum_{i \in I_1(p) \cup I_2(p)} b_i + \max \left\{ 0, \left\lceil \frac{1}{L} \left( \sum_{i \in I_3(p)} \ell_i b_i - (L - \sum_{i \in I_2(p)} b_i - \sum_{i \in I_2(p)} \ell_i b_i) \right) \right\rceil \right\}.$$

Taking the maximal value, we obtain

$$lb_{MT}(E) := \max\{lb_{MT}(E, p) : p \in [0, L/2] \cap \mathbb{Z}\}.$$

As already mentioned in [19], in order to compute  $lb_{MT}(E)$ , only values  $p \in \{\ell_i : i \in I, \ell_i \leq L/2\}$  have to be considered since for the other ones the index sets do not change. In analogy to the concept of *reduced set of potential allocation points* (Sect. 2.7), the computational effort can be further decreased, see Sect. 3.2.

The bound  $lb_{MT}(E)$  can be computed in  $O(m)$  time if the item lengths are presorted. The worst-case performance ratio of  $lb_{MT}(E)$  is equal to 1.5 [19].

The bound  $lb_{MT}(E, p)$  is oriented on the number of *larger* items; *small* items are disregarded, and the lengths of items of  $I_3(p)$  are continuously distributed. Another lower bound can be obtained in analogy to  $lb_d$ . To this end, we consider the maximal number of items that can be packed into a pattern if an item of  $I_2(p)$  is already contained therein:

$$lb_s(E, p) := \sum_{i \in I_1(p) \cup I_2(p)} b_i + \max \left\{ 0, \left\lceil \frac{|I_3(p)| - \sum_{i \in I_2(p)} \lfloor b_i(L - \ell_i)/p \rfloor}{\lfloor L/p \rfloor} \right\rceil \right\},$$

$$lb_s(E) := \max \{lb_s(E, p) : p \in (0, L/2] \cap \mathbb{Z}\}. \quad (4.16)$$

Besides the presented lower bounds, the concept of *dual feasible functions*, described in Sect. 3.2.3, can be used to strengthen lower bounds.

## 4.7 Equivalence of Instances

Let  $E = (m, \ell, L, b)$  denote an instance of the 1CSP. Furthermore, let  $(a^j)_{j \in J^{max}}$  be the set of all *maximal* patterns of  $E$ , i.e., we have  $a^j \in \mathbb{Z}_+^m$  and  $L - \min\{\ell_i : i \in I\} < \ell^\top a^j \leq L$  for all  $j \in J^{max}$ . In order to define the equivalence of instances, we also consider minimal infeasible patterns. A vector  $\bar{a} \in \mathbb{Z}_+^m$  with  $\ell^\top \bar{a} > L$  is called *minimal infeasible pattern* if there does not exist any vector  $\tilde{a} \in \mathbb{Z}_+^m$  with  $\tilde{a} \neq \bar{a}$ ,  $\ell^\top \tilde{a} > L$  and  $\tilde{a}_i \leq \bar{a}_i$  for all  $i \in I$ . Let  $\bar{J}$  be an index set of all minimal infeasible patterns which we represent by  $\bar{a}^j, j \in \bar{J}$ .

**Definition 4.1** Instance  $E' = (m, \ell', L', b')$  with  $\ell' \in \mathbb{R}_{>}^m$ ,  $L' \in \mathbb{R}_{>}$  is *equivalent* to instance  $E = (m, \ell, L, b)$  of the 1CSP if

$$(\ell')^\top a^j \leq L' \quad \text{for all } j \in J, \quad (4.17)$$

$$(\ell')^\top \bar{a}^j > L' \quad \text{for all } j \in \bar{J}. \quad (4.18)$$

**Proposition 4.6** Let  $x^c = x^c(E)$  denote a solution of the LP relaxation of the standard model for the 1CSP,  $z^c$  its objective value, and let  $d \in \mathbb{R}_+^m$  be a corresponding vector of simplex multipliers. Then,  $E' := (m, \ell' = d, L' = 1, b)$  is an instance satisfying (4.17) with  $lb_M(E') = \lceil z^c \rceil$ .

*Proof* Since  $x^c$  is optimal, we have  $c_j - d^\top a^j \geq 0$  for all  $j \in J$ , and therefore,  $d^\top a^j \leq 1$  for all  $j \in J$ . Moreover, let  $x_B \in \mathbb{R}_+^m$  be the vector of basis variables of  $x^c$ ,  $A_B$  the basis matrix, and  $c_B$  the corresponding objective coefficients. Then

$$z^c = c_B^\top x_B = c_B^\top A_B^{-1} b = d^\top b$$

holds which proves the statement. ■

Note that  $E'$ , as defined in Proposition 4.6, is not necessarily an equivalent instance of  $E$  since  $E'$  can have more feasible patterns than  $E$ . For instance, vector  $(4, 0, 0, 0)^\top$  is not feasible for  $E$  in Example 4.3, but feasible for the instance  $E' = (4, (\frac{1}{4}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2})^\top, 1, b)$  which results from the LP solution.

A possibility to obtain equivalent instances consists in modifying single input data. In literature, such method is frequently called a *reduction method*. Either we aim to reduce  $L$  or to increase the length of some items while maintaining the feasibility of all patterns. By means of a modified (reduced) instance, possibly some bounds get a larger value. Let  $E = (m, \ell = (\ell_1, \dots, \ell_m)^\top, L, b)$  be an instance of the 1CSP (or the 1BPP). Throughout this section, let  $\bar{E} = (m, (\bar{\ell}_1, \dots, \bar{\ell}_m)^\top, \bar{L}, b)$  denote a reduced instance of  $E$  with  $\bar{L} \in \mathbb{Z}_+$  and  $\bar{\ell}_i \in \mathbb{Z}_+$  for all  $i \in I$ .

Clearly, if

$$L > L^* \quad \text{with} \quad L^* := \max \left\{ \sum_{i \in I} \ell_i a_i : \sum_{i \in I} \ell_i a_i \leq L, a_i \in \mathbb{Z}_+, I \in I \right\},$$

then we can set  $\bar{L} := L^*$ , maintaining all other input data, and obtain a stronger material bound.

In the reduction method, proposed by Boschetti and Mingozi [9] for the 2BPP, successively a single item size  $\ell_i$  is increased such that the waste in each pattern containing piece  $i$  is reduced. Here we adapt the approach to the 1CSP. Some or all values  $L_{ik}^*, k = 1, \dots, k_i := \lfloor L/\ell_i \rfloor, i \in I$ , have to be computed where

$$L_{ik}^* := k \cdot \ell_i + \max \left\{ \sum_{j \in I \setminus \{i\}} \ell_j a_j : \sum_{j \in I \setminus \{i\}} \ell_j a_j \leq L - k\ell_i, a_j \in \mathbb{Z}_+, j \in I \setminus \{i\} \right\}. \quad (4.19)$$

Since  $L_{ik}^*$  is the optimal value of a knapsack problem having the additional constraint  $a_i = k$ , it can be computed in pseudo-polynomial time. In case of  $L_{ik}^* < L$  for all  $k$ , then piece length  $\ell_i$  possibly can be increased. More precisely, if

$$\Delta_i := \min \left\{ \lfloor \frac{1}{k} (L - L_{ik}^*) \rfloor : k = 1, \dots, k_i \right\} > 0,$$

then we can define

$$\bar{\ell}_j := \begin{cases} \ell_i + \Delta_i, & j = i, \\ \ell_j, & \text{otherwise,} \end{cases} \quad j \in I,$$

and obtain an equivalent instance  $\bar{E}$  for  $E$ . Note that all other input data remain unchanged. After increasing an item length, further attempts can be performed for strengthening bounds. Several questions arise concerning the computational effort needed to compute all these values, or the strategy of choosing items to be increased.

**Proposition 4.7** Let  $E = (m, \ell, L, b)$  denote an instance of the 1CSP. The computation of all values  $L_{ik}^*$  according to (4.19) can be done in  $O(L(2m + \sum_{i \in I} k_i))$  time. In particular, the complexity is  $O(mL)$  for the 1BPP.

*Proof* For  $k \in I$  and  $y = 0, 1, \dots, W$  we define

$$F(k, y) := \max \left\{ \sum_{i=1}^k \ell_i a_i : \sum_{i=1}^k \ell_i a_i \leq y, a_i \in \mathbb{Z}_+, i \in \{1, \dots, k\} \right\}$$

and

$$\bar{F}(k, y) := \max \left\{ \sum_{i=k}^m \ell_i a_i : \sum_{i=k}^m \ell_i a_i \leq y, a_i \in \mathbb{Z}_+, i \in \{k, \dots, m\} \right\}.$$

The computation of all values  $F(k, y)$  and  $\bar{F}(k, y)$  requires  $O(m \cdot L)$  operations according to the recurrence formula of Gilmore and Gomory, Sect. 2.2. Let  $i \in \{2, \dots, m-1\}$  and  $k \in \{1, \dots, k_i\}$ , then  $L_{ik}^*$  can be computed according to

$$L_{ik}^* = k\ell_i + \max_{y=0, \dots, L-k\ell_i} \{F(i-1, y) + \bar{F}(i+1, L - k\ell_i - y)\}$$

in  $O(L)$  time, so that the determination of all values requires  $O(L \sum_{i \in I} k_i)$  time. Summing up, the asserted total effort results.

In the particular case of the 1BPP we have  $k_i = 1$  for all  $i \in I$  and, therefore, obtain  $O(m \cdot L)$ . ■

The strategy which piece type has to be chosen depends on the potential effect of improving bounds and the effort to compute the modified instance. The addressed methods are applicable to higher-dimensional C&P problems as we will see on appropriate places.

## 4.8 Alternative Models

Besides the standard model of the 1CSP which has, in general, an exponential number of variables, there exist models with a pseudo-polynomial number of variables: Kantorovich-type models, the arcflow model, and the one-cut model. We review here these models and discuss relationships between them and the standard model. Let again  $E = (m, \ell, L, b)$  denote an instance of the 1CSP.

### 4.8.1 A Nonlinear Model

An important disadvantage of the pattern-oriented standard model is the, in general, huge number of variables. Therefore, we consider the coefficients of the patterns

also as variables. Let  $n_0 \in \mathbb{Z}$  be sufficiently large so that a solution of  $E$  exists having not more than  $n_0$  different patterns, and let  $J_0 := \{1, \dots, n_0\}$ . ( $n_0$  can be obtained by a heuristic.) Then, a nonlinear model of the 1CSP is as follows:

### Nonlinear model of the 1CSP

$$z^{NL} = \min \sum_{j \in J_0} x_j \quad \text{s.t.} \quad (4.20a)$$

$$\sum_{i \in I} \ell_i a_{ik} \leq L, \quad j \in J_0, \quad (4.20b)$$

$$\sum_{j \in J_0} a_{ij} x_j \geq b_i, \quad i \in I, \quad (4.20c)$$

$$a_{ij} \in \mathbb{Z}_+, \quad i \in I, \quad x_j \in \mathbb{Z}_+, \quad j \in J_0. \quad (4.20d)$$

Due to the nonlinearity in (4.20c) and the integer constraints (4.20d), this model is in particular difficult to solve. One opportunity to overcome the nonlinearity consists in predefining all patterns as in the standard model: only the  $x_j$  remain as variables. Subsequently, we consider two further variants.

A first mathematical model of the 1CSP was proposed by Kantorovich [18] in 1939. Since there the coefficients  $a_{ij}$  of the patterns are variables as in (4.20c), we name the two linearizations of model (4.20) to be of Kantorovich-type.

### 4.8.2 Kantorovich-Type Models

In the first variant of linearization, the number  $n_0$  of possibly different patterns depends on the optimal value  $z^*$ , and therefore, on the order demands  $b_i$ ,  $i \in I$ . In fact, the variables  $x_j$  are substituted by binary variables.

Let  $\bar{z}$  and  $\underline{z}$  denote a finite upper and a lower bound of the optimal value of  $E$ , respectively. For instance,  $\bar{z}$  can be determined by a feasible solution obtained by a heuristic, whereas the *material bound*  $\underline{z} := lb_M = \lceil \sum_{i \in I} \ell_i b_i / L \rceil$  defines a natural lower bound for  $z^*$ . Since possibly more than  $\underline{z}$  pieces of raw material are needed, we define decision variables  $y_j$ ,  $j = \underline{z} + 1, \dots, n_0 := \bar{z}$ , to model whether the  $j$ th piece of length  $L$  is used. The variables  $a_{ij}$ ,  $i \in I$ ,  $j \in J_0 := \{1, \dots, \bar{z}\}$ , represent the frequency items of length  $\ell_i$  are obtained from the  $j$ th piece in stock.

#### Kantorovich-type Model 1 of the 1CSP

$$z^{K1} = \underline{z} + \min \sum_{j=\underline{z}+1}^{\bar{z}} y_j \quad \text{s.t.} \quad (4.21a)$$

$$\sum_{i \in I} \ell_i a_{ij} \leq L, \quad j = 1, \dots, \underline{z}, \quad (4.21b)$$

$$\sum_{i \in I} \ell_i a_{ij} \leq L y_j, \quad j = \underline{z} + 1, \dots, \bar{z}, \quad (4.21c)$$

$$\sum_{j \in J_0} a_{ij} \geq b_i, \quad i \in I, \quad (4.21d)$$

$$y_j \in \{0, 1\}, \quad j = \underline{z} + 1, \dots, \bar{z}, \quad a_{ij} \in \mathbb{Z}_+, \quad i \in I, \quad j \in J_0. \quad (4.21e)$$

Since at least  $\underline{z}$  patterns are needed to satisfy the order demands, additionally to the feasibility constraints (4.21b) for the first  $\underline{z}$  patterns, condition (4.21c) asks whether pattern  $a^j$  is needed at all. If not, then  $a_{ij} = 0$  follows for all  $i \in I$ . Clearly, inequalities (4.21d) can be replaced by equations if overproduction is not permitted.

The LP relaxation of the Kantorovich-type model (4.21) is obtained by substituting the integer demand in (4.21e) by  $0 \leq y_j \leq 1$  and  $a_{ij} \geq 0$  for all  $i$  and  $j$ . The lower bound  $z_{LP}^{K1}$  for  $z^* = z^{K1}$  provided by the LP relaxation is equal to  $\underline{z}$  if  $\underline{z} \geq lb_M$ . This can be seen by showing the feasibility of  $a_{ij} := b_i / \underline{z}$  for all  $i \in I$  and  $j = 1, \dots, \underline{z}$ . Hence, the gap  $z^* - z_{LP}^{K1}$  can rise when the order demands are increased. Because of the weak lower bound the applicability of the Kantorovich-type model is mostly restricted.

A further disadvantage of the model could be a high degree of symmetric solutions. In particular, there exists at least  $\underline{z}!$  equivalent solutions obtainable by permuting the indices of the patterns.

In order to avoid this kind of symmetry, we can search for a solution with lexicographically sorted patterns. Let  $u, v \in \mathbb{R}^m$ , then  $u$  is *lexicographically larger* than  $v$  if the implication

$$(\forall j \in \{1, \dots, i-1\} : v_j = u_j) \Rightarrow u_i \geq v_i$$

holds for all  $i \in I$ . Then, the sorting can be achieved by adding some binary variables and linear constraints. To this end, we consider the following lemma.

#### Lemma 4.8

- (a) Let  $u, v \in \mathbb{Z}_+$  with  $\max\{u, v\} \leq M - 1$  where  $M$  is a constant, and let  $\alpha \in \mathbb{B}$ . Then we have:

$$u > v \Leftrightarrow [u - v \leq M\alpha, u - v \geq 1 + (\alpha - 1)M \Rightarrow \alpha = 1].$$

- (b) Let  $u, v \in \mathbb{Z}_+^m$  with  $\max_{i \in I} \{u_i, v_i\} \leq M - 1$  where  $M$  is a constant, and let  $\alpha \in \mathbb{B}^m$ . Then we have:  $u$  is lexicographically larger than  $v$  if and only if there exists  $\alpha \in \mathbb{B}^m$  such that the system of inequalities

$$u_i - v_i \leq M\alpha_i, \quad u_i - v_i \geq 1 + (\alpha_i - 1)M, \quad v_i - u_i \leq M \sum_{k=1}^{i-1} \alpha_k$$

is satisfied for all  $i \in I$ .

A proof can be found in [20].

As a consequence of the previous lemma, in order to obtain a lexicographically sorted solution of the Kantorovich-type model (4.21), we have to add  $\bar{z}$  vectors  $\alpha^j = (\alpha_{1j}, \dots, \alpha_{mj})^\top \in \mathbb{B}^m$  and the inequalities

$$a_{ij} - a_{i,j+1} \leq M\alpha_{ij}, \quad a_{ij} - a_{i,j+1} \geq 1 + (\alpha_{ij} - 1)M, \quad a_{i,j+1} - a_{ij} \leq M \sum_{k=1}^{i-1} \alpha_{kj}$$

for all  $i \in I$  and  $j \in \{1, \dots, \bar{z} - 1\}$ . Obviously, the number of binary variables is increased which could be a disadvantage. Alternatively, the usage of the additional conditions  $y_j \geq y_{j+1}$  for  $j = \underline{z} + 1, \dots, \bar{z} - 1$  could be meaningful to avoid such symmetries.

A more practicable method consists in assigning a real number  $\kappa(a^j)$  to each pattern  $a^j, j = 1, \dots, \bar{z} - 1$  where

$$\kappa(a^j) := \sum_{i \in I} \mu^{m-i} a_{ij} \quad \text{with } \mu := \max\{\lfloor L/\ell_i \rfloor : i \in I\} + 1.$$

Then the restrictions

$$\kappa(a^j) \geq \kappa(a^{j+1}) \quad \text{for } j = 1, \dots, \bar{z} - 1$$

ensure a lexicographic ordering. Because of a finite mantissa length of computer numbers, the latter method cannot be considered as an exact method in applications.

In the second variant of linearization, the predefined number  $n_0$  of possibly different patterns does not depend on the optimal value  $z^*$ , but on the maximal number of different patterns in a solution. Let  $J_0 := \{1, \dots, n_0\}$ . The variables  $x_j$  are maintained and the unknown pattern coefficients  $a_{ij}$  are replaced by binary variables.

Since the magnitude of  $a_{ij}$  is restricted by  $k_i := \lfloor L/\ell_i \rfloor$ , we can represent  $a_{ij}$  by  $k_i$  binary variables  $\beta_{ik}^j$  as follows

$$a_{ij} = \sum_{k \in K_i} k \cdot \beta_{ik}^j \quad \text{with} \quad \sum_{k \in K_i} \beta_{ik}^j \leq 1 \quad \text{for } i \in I, j \in J_0,$$

where  $K_i := \{1, \dots, k_i\}$ . To this end, the term  $a^j \cdot x_j \in \mathbb{Z}_+^m$  has to be replaced by a vector

$$\alpha^j = (\alpha_{1,1}^j, \dots, \alpha_{1,k_1}^j, \alpha_{2,1}^j, \dots, \alpha_{m,k_m}^j) \in \mathbb{Z}^{n_1} \quad \text{with} \quad \alpha_{ik}^j = \beta_{ik}^j \cdot x_j$$

where  $n_1 := \sum_{i \in I} k_i$ .

### Kantorovich-type Model 2 of the 1CSP

$$z^{K2} = \min \sum_{j \in J_0} x_j \quad \text{s.t.} \tag{4.22a}$$

$$\sum_{i \in I} \ell_i \cdot \sum_{k \in K_i} k \cdot \beta_{ik}^j \leq L, \quad j \in J_0, \quad (4.22b)$$

$$\sum_{k \in K_i} \beta_{ik}^j \leq 1, \quad i \in I, j \in J_0, \quad (4.22c)$$

$$\alpha_{ik}^j \leq M \beta_{ik}^j, \quad k \in K_i, i \in I, j \in J_0, \quad (4.22d)$$

$$\sum_{j \in J_0} \sum_{k \in K_i} k \cdot \alpha_{ik}^j \geq b_i, \quad i \in I, \quad (4.22e)$$

$$\sum_{k \in K_i} \alpha_{ik}^j \leq x_j, \quad i \in I, j \in J_0, \quad (4.22f)$$

$$M \sum_{k \in K_i} \beta_{ik}^j - \sum_{k \in K_i} \alpha_{ik}^j + x_j \leq M, \quad i \in I, j \in J_0, \quad (4.22g)$$

$$\beta_{ik}^j \in \mathbb{B}, \alpha_{ik}^j \in \mathbb{Z}_+, x_j \in \mathbb{Z}_+, \quad k \in K_i, i \in I, j \in J_0. \quad (4.22h)$$

Here,  $M$  is a sufficiently large constant, e.g.  $M = \max\{b_i : i \in I\}$ . A similar model is considered, for instance, in [17]. Because of (4.22f) and (4.22g), all nonzero entries of  $\alpha^j$  have the same value  $x_j$ .

An advantage of this model consists in the aspect that the number  $n_0$  of different patterns can easily be controlled. However, a main problematic aspect is the weakness of its LP relaxation. The LP bound can be smaller than the material bound. We illustrate this by means of the following example.

*Example 4.4* Considering the 1CSP instance  $E = (m = 2, \ell = (6, 7), L = 10, b = (2, 2))$ , we obtain  $z^*(E) = 4$  and  $lb_M = 3$ . Since  $k_i = 1$  for  $i \in I = \{1, 2\}$ , we can omit the index  $k$  and related terms to simplify the model. For  $n_0 = 2$  and  $M = 2$  a feasible LP solution is given by

$$\beta_i^j = 0.5, \quad \alpha_i^j = 1, \quad x_j = 1, \quad j = 1, 2, \quad i = 1, 2,$$

which implies  $z_{LP}^{K_2} \leq 2$ . ■

Similar to model (4.21), a high degree of symmetric solutions is present due to permuting the pattern sequence. Besides the possibilities to overcome this issue as discussed above for model (4.21), in [17], a simple method is proposed demanding

$$x_j \geq x_{j+1} \quad \text{for } j = 1, \dots, n_0 - 1$$

which does not exclude all symmetries.

### 4.8.3 The One-Cut Model

In difference to the Gilmore/Gomory model where a variable is assigned to each pattern, in the approach proposed by Dyckhoff [13] a variable is related to a single

particular cut. More precisely, if raw material of length  $p$  is cut to produce a desired length  $q \in \mathbb{L} := \{\ell_1, \dots, \ell_m\}$  and some residual length  $p - q$ , then we assign the variable  $x_{pq}$ . The resulting residual length itself is either a desired length (if  $p - q \in \mathbb{L}$ ), or can be used to produce more desired pieces (if  $p - q > \ell_{min} := \min\{\ell_i : i \in I\}$ ), or is waste.

Let  $\mathbf{R}$  denote the set of all useful residual lengths, i.e.,

$$\mathbf{R} := \{r \in [\ell_{min}, L) : r = L - \sum_{i \in I} \ell_i a_i, a_i \in \mathbb{Z}_+, i \in I\}.$$

Then, the set  $\mathbf{M}$  of feasible cuts, considered in the model, is

$$\mathbf{M} := \{(p, q) : p \in \mathbf{R} \cup \{L\}, q \in \mathbb{L}, q < p\}.$$

Since we assign an integer variable  $x_{p,q}$  to each  $(p, q) \in \mathbf{M}$ , we obtain a model with a pseudo-polynomial number of position-indexed variables.

### One-Cut Model of the 1CSP

$$z^{OC} = \min \sum_{i \in I} x_{L, \ell_i} = \min \sum_{q \in \mathbb{L}} x_{L, q} \quad \text{s.t.} \quad (4.23a)$$

$$\sum_{p \in \mathbf{R} \cup \{L\}, q < p} x_{p,q} + \sum_{p+q \in \mathbf{R} \cup \{L\}, p \in \mathbf{L}} x_{p+q,p} \geq \sum_{p \in \mathbf{L}, q > p} x_{q,p} + N_q, \quad q \in \mathbf{R}, \quad (4.23b)$$

$$x_{p,q} \in \mathbb{Z}_+, \quad (p, q) \in \mathbf{M}, \quad (4.23c)$$

$$\text{where } N_q := \begin{cases} b_i, & \text{if } q = \ell_i \in \mathbb{L}, \\ 0, & \text{if } q \in \mathbf{M} \setminus \mathbb{L}. \end{cases}$$

By means of  $N_q$  two cases are combined within one set of restrictions. If the considered length  $q$  is a desired product length, then the order demand plus the times length  $q$  is further dissected, does not exceed the total number length  $q$  is obtained. If  $q$  is an intermediate length, then at least as many have to be obtained as are further cut. Note that the reduction of the number of variables, in comparison to the standard model, goes along with an increase of constraints ( $O(mL)$  and  $O(L)$ , respectively).

A more sophisticated formulation of a one-cut model, involving fewer variables and constraints, is proposed in [21]. Therein, the equivalence to the standard model also with respect to their LP relaxations is shown.

#### 4.8.4 The Arcflow Model

In order to formulate a graph-theoretical model as proposed by de Carvalho [11], we observe that any pattern  $a \in \mathbb{Z}_+^m$  can be seen as a path in a graph  $G = (V, E)$  with

vertex set  $V \subset \{0, 1, \dots, L\}$  and arc set  $E \subset V \times V$ . If, for instance,  $a_1 > 0$  and  $a_2 > 0$ , then we assign arcs  $(0, \ell_1), (\ell_1, 2\ell_1), \dots, ((a_1 - 1)\ell_1, a_1\ell_1)$ , and  $(a_1\ell_1, a_1\ell_1 + \ell_2)$  to the paths, and so on. Moreover, the frequency of using  $a$  can be considered as the flow value along the path. Combining all paths belonging to patterns, we obtain an arcflow formulation of the 1CSP.

It is obvious, the assumption that all input data are integers is important, in particular, in this subsection. Moreover, to keep the number of variables small, we assume, without loss of generality, that

$$L > \ell_1 > \dots > \ell_m > 0$$

holds. The vertex set  $V$  of  $G$  represents all total lengths of feasible patterns, i.e.

$$V := \{r \in [0, L) : r = \sum_{i \in I} \ell_i a_i, a_i \in \mathbb{Z}_+, i \in I\} \cup \{L\}.$$

Since, we have, in general, many possibilities to assign a path to a pattern  $a \in \mathbb{Z}_+^m$ , due to different sequences of the arcs, many equivalent solutions can exist. To keep this number small, we aim to consider only paths with nonincreasing lengths of their arcs. For that reason, let

$$\mu(p) := \begin{cases} 0, & \text{for } p = 0, \\ \min\{i \in I : p - \ell_i \in V, i \geq \mu(p - \ell_i)\}, & \text{for } p \in V \setminus \{0\}. \end{cases}$$

Then, the arc set  $E$  is defined as follows:

$$E := E_1 \cup E_2 \quad \text{where}$$

$$E_1 := \{(p, q) \in V \times V : \exists i \in I \text{ with } i \geq \mu(p), p + \ell_i = q\},$$

$$E_2 := \{(p, L) : p \in V \setminus \{L\}, p > L - \ell_m\}.$$

Obviously, the arcs in  $E_1$  represent by their lengths (= difference of vertex values) desired pieces. The arcs in  $E_2$  model the waste of a pattern and ensure that the path ends in vertex  $L$ . According to an arcflow problem, we assign a variable  $x_{pq} \in \mathbb{Z}_+$  to arc  $(p, q) \in E$  and obtain the

### Arcflow Model of the 1CSP

$$z^{AF} = \min \sum_{(0,q) \in E} x_{0,q} \quad \text{s.t.} \tag{4.24a}$$

$$\sum_{(p,q) \in E} x_{pq} = \sum_{(q,r) \in E} x_{qr}, \quad q \in V \setminus \{0, L\}, \tag{4.24b}$$

$$\sum_{p \in V, p + \ell_i \leq L} x_{p,p+\ell_i} \geq b_i, \quad i \in I, \quad (4.24c)$$

$$x_{pq} \in \mathbb{Z}_+, \quad (p, q) \in E. \quad (4.24d)$$

The target function (4.24a) minimizes the total flow value, and therefore, the number of stock material pieces needed. The flow conservation constraints (4.24b) ensure that patterns can be reconstructed from a solution of the arcflow model. Finally, constraints (4.24c) guarantee the satisfaction of all order demands. More details, in particular, the equivalence with the standard model and of their LP relaxations are addressed in [11] and [12]. Moreover, similar results are presented in [21] concerning the equivalence of the arcflow and one-cut model.

An application of the arcflow model is considered in Exercise 4.17.

## 4.9 Neighboring Problems

Besides the minimization of material usage, frequently further objectives are considered in applications.

In case of allowing overproduction it may be desirable to restrict the magnitude of it, i.e., restrictions of form

$$b_i \leq \sum_{j \in J} a_{ij} x_j \leq \bar{b}_i \quad \text{for all } i \in I$$

have to be considered where  $\bar{b}_i$  limits the overproduction of pieces of type  $i \in I$ . Note that such restrictions can be handled as *box constraints* for the slack variables  $s_i$  by demanding

$$0 \leq s_i \leq \bar{b}_i - b_i \quad \text{for all } i \in I.$$

Another issue is to regard *setup costs* which are meaningful sometimes. Within the production process, besides the material costs, the change from one pattern to a different pattern can cause additional costs, the setup costs, so that two cost components have to be combined in the target function. Depending on the underlying CSP model, different solution approaches are thinkable. Mostly, among all, such a solution of the CSP is searched having additionally a minimal number of different patterns. As proven in [7], where an efficient heuristic based on the *Sequential Value Correction* method is proposed (Sect. 7.4.2), allowing a small increase of the pattern number to be cut (e.g. cutting of  $z^* + 1$  pieces of raw material) can lead to a drastic reduction of the number of different patterns within the feasible solution. In respect to the *setup costs minimization problem* also the notation *pattern minimization problem* is frequently used.

As one possibility to handle setup costs in an ILP, we consider again the K2 model (4.22). Let  $c_m$  and  $c_s$  denote the cost coefficients per unit stock material and per setup, respectively. Introducing additional binary variables  $y_j, j \in J_0$ , indicating whether pattern  $j$  is used, we obtain the target function

$$z^{K2} = \min c_m \cdot \sum_{j \in J_0} x_j + c_s \cdot \sum_{j \in J_0} y_j.$$

According to the definition of the  $y_j$ -variables, we have to change restriction (4.22b) to

$$\sum_{i \in I} \ell_i \sum_{k \in K_i} k \cdot \beta_{ik}^j \leq L y_j, \quad j \in J_0,$$

and to add

$$x_j \leq M y_j, \quad j \in J_0.$$

Both restrictions ensure that in case of  $y_j = 0$  all variables related to pattern  $j$  become zero.

Technological restrictions, frequently appearing in applications, result from the restricted space around the cutting machine. In order to avoid many transportation tasks, it can be required that for each piece type a single *stack* is opened when the first item of this type is produced. This stack remains open until the last item of this type is obtained, and then it is closed. Thereafter the stack can be used for another piece type. If the number of available stacks is  $n_{os}$ , then a feasible solution of the CSP has to be found which satisfies this *open stack condition* with minimal material usage. An efficient heuristic for the open stack problem is proposed, for instance, in [7].

Another type of additional restrictions results from the occurrence of *due dates* meaning that all items of a certain type have to be obtained at the latest of the predefined due date.

In specific applications a *minimum allowable pattern run length*  $\underline{x}_0$  is considered meaning that, if a pattern  $a^j$  should be cut, then at least  $\underline{x}_0$  times, i.e., either  $x_j \geq \underline{x}_0$  or  $x_j = 0$  has to hold for all  $j$ . Sometimes,  $x_j = k \cdot \underline{x}_0$  is demanded with  $k \in \mathbb{Z}_+$ , or in a similar way a minimum number of applying a pattern is given.

All these additional restrictions, which can be of high relevance in application, lead to even harder problems to solve. The suitability of the above introduced models to handle such scenarios varies from case to case.

A further situation, called the *contiguous 1CSP*, is addressed in Sect. 7.2.3 as relaxation of a two-dimensional problem. The contiguity condition demands that if the cutting (production) of items of a certain piece type is started, then all items of that type have to be obtained without interruption. This kind of problem has importance not only in the field of C&P (for instance, for rectangle packing), but also in other fields as, for example, for scheduling problems.

## 4.10 The Cutting Stock Problem with Multiple Stock Lengths

Among the big variety of different practical problems related to cutting and packing, we are very often confronted with problems where various types of raw material are available. They are frequently called *Cutting Stock Problem with Multiple Stock Lengths* or *Multiple (Lengths) Cutting Stock Problem* (MCSP). In the one-dimensional case, pieces of lengths  $\ell_i, i \in I = \{1, \dots, m\}$  with order demands  $b_i \in \mathbb{N}$  have to be obtained from rods (stock lengths) of lengths  $L_1, \dots, L_q$ . Without loss of generality, we assume that the first  $p$  ( $p < q$ ) types of rods have a restricted availability, i.e., at most  $u_k$  rods of length  $L_k$  can be used,  $k = 1, \dots, p$ . For the remaining types of rods  $L_k, k = p + 1, \dots, q$ , we assume that a sufficiently large number is available and that  $\max\{\ell_i : i \in I\} \leq \max\{L_k : k \in \{p + 1, \dots, q\}\}$  holds.

*Remark 4.6* The last assumption is sufficient to guarantee solvability of the problem for any order demands  $b_i, i \in I$ . Nevertheless, if the existence of feasible solutions can be ensured in another way, modifications of the subsequent model can easily be derived.

For abbreviation, let  $K_q := \{1, \dots, q\}$  and  $K_p := \{1, \dots, p\}$ . To formulate a mathematical model for the MCSP, we need to define an appropriate target function, that means, a valuation by means of cost coefficients  $c_k, k \in K_q$  has to be provided for the different types of stock material. Setting  $c_k := L_k$  for all  $k \in K_q$ , the total usage of material can be minimized. Moreover, modifying the cost coefficients enables to regard priority rules.

In analogy to the case  $q = 1$ , we represent feasible patterns for rod length  $L_k, k \in K_q$ , by integer vectors

$$a^{jk} = (a_{1jk}, \dots, a_{mjk})^\top \in \mathbb{Z}_+^m \quad \text{satisfying} \quad \sum_{i \in I} \ell_i a_{ijk} \leq L_k, \quad j \in J_k$$

where  $J_k$  denotes an appropriate index set of patterns related to  $L_k, k \in K_q$ . As usual,  $a_{ijk}$  indicates how many pieces of type  $i$  are obtained when a single rod of length  $L_k$  is cut according to pattern  $a^{jk}$ . Note that in case of small order demands, as for instance in the *multiple bin packing problem*, i.e., a BPP with different bin sizes, the additional restriction

$$a_{ijk} \leq b_i, \quad i \in I, \quad j \in J_k, \quad k \in K_q,$$

should be taken into account. We denote the frequency of using pattern  $a^{jk}$  by  $x_{jk}$ .

### Model of the cutting stock problem with multiple stock lengths

$$z^{MCSP} = \min \sum_{k \in K_q} c_k \sum_{j \in J_k} x_{jk} \quad \text{s.t.} \tag{4.25a}$$

$$\sum_{k \in K_q} \sum_{j \in J_k} a_{ijk} x_{jk} \geq b_i, \quad i \in I, \quad (4.25b)$$

$$\sum_{j \in J_k} x_{jk} \leq u_k, \quad k \in K_p, \quad (4.25c)$$

$$x_{jk} \in \mathbb{Z}_+, \quad j \in J_k, \quad k \in K_q. \quad (4.25d)$$

To obtain an approximate solution of the integer problem (4.25), we propose, in analogy to Sect. 4.3.1, to solve the corresponding continuous relaxation and to apply appropriate heuristics based on its solution.

*Remark 4.7* Sometimes, it can be desirable to have equality in restrictions (4.25c), e.g., if some rod types should be removed from stock, then difficulties can arise with respect to solvability or constructing integer solutions from the LP solution. Here we do not consider this aspect in more detail.

The continuous relaxation of model (4.25) can be solved using the column generation technique. Based on a reformulation having equality constraints by means of  $m+p$  slack variables and an initial feasible basis solution, in each iteration step of the simplex method the following generation (slave) problems have to be solved:

$$\bar{c}_k := \min\{c_k - \sum_{i \in I} d_i a_i - d_{m+k} : \sum_{i \in I} \ell_i a_i \leq L_k, a_i \in \mathbb{Z}_+, i \in I\}, \quad k \in K_p, \quad (4.26)$$

$$\bar{c}_k := \min\{c_k - \sum_{i \in I} d_i a_i : \sum_{i \in I} \ell_i a_i \leq L_k, a_i \in \mathbb{Z}_+, i \in I\}, \quad k \in K_q \setminus K_p, \quad (4.27)$$

$$\bar{c}_0 := \min\{\min_{i \in I} d_i, -\max_{k \in K_p} d_{m+k}\}. \quad (4.28)$$

Here, vector  $d = (d_1, \dots, d_{m+p})^\top$  denotes again the respective vector of simplex multipliers.

Hence, solving the  $p$  slave problems in (4.26) and the  $q-p$  slave problems in (4.27) requires, altogether, the determination of  $q$  knapsack problem solutions. Fortunately, the  $q$  optimal values can be obtained by applying a dynamic programming based method to solve a single knapsack problem with right hand side  $\max_{k \in K_q} L_k$ . The simplex multipliers related to the slack variables yield  $\bar{c}_0$  in (4.28). In case of  $\min\{\bar{c}_k : k = 0, \dots, q\} \geq 0$  the optimality of the current LP solution is proved. Otherwise, at least a new column (pattern or negative unit vector) is found which can lead to an improved feasible solution of the LP relaxation.

Besides the generalization of the Gilmore/Gomory model, an adaptation of the arcflow and the one-cut model to the MCSP is straightforwardly possible, see e.g. [1].

The scenario of several different lengths occurs also for bin packing problems, frequently called *variable size bin packing problem*. Exact and heuristic approaches are proposed, for instance, in [3] where also additional practical restrictions are considered.

## 4.11 Integer Round Up and Modified Integer Round Up Property

A phenomenon which can be observed when solving instances of the 1BPP or the 1CSP is the always small difference between the respective optimal values of the integer problem and the corresponding continuous relaxation. In the following we investigate this issue in more detail, in particular, we study the *Integer Round-Up Property* (IRUP) and the *Modified Integer Round-Up Property* (MIRUP) of the one-dimensional cutting stock problem.

### 4.11.1 Definitions

Let  $E = (m, \ell, L, b)$  denote an instance of the 1CSP where  $m$  is the number of piece types,  $L$  the length of the raw material,  $\ell = (\ell_1, \dots, \ell_m)^\top \in \mathbb{Z}_+^m$  the vector of piece lengths, and  $b = (b_1, \dots, b_m)^\top \in \mathbb{Z}_+^m$  the vector of order demands. Furthermore, let  $I := \{1, \dots, m\}$ . Without loss of generality, we assume that  $b_i \geq 1$  for all  $i \in I$  and  $L \geq \ell_1 > \ell_2 > \dots > \ell_m > 0$  hold. Vector  $a^j = (a_{1j}, \dots, a_{mj})^\top \in \mathbb{Z}_+^m$  ( $j \in J$ ) represents a cutting pattern (hence  $\ell^\top a^j \leq L$ ) where  $J = \{1, \dots, n\}$  is an appropriate index set of all patterns of  $E$ . If  $x_j$  denotes the frequency how often  $a^j$  is used in the solution, then the standard (Gilmore/Gomory) model of the (one-dimensional) cutting stock problem can be stated as follows:

$$z^*(E) = \min \left\{ \sum_{j \in J} x_j : \quad \sum_{j \in J} a_{ij} x_j \geq b_i, \quad i \in I, \quad x_j \in \mathbb{Z}_+, \quad j \in J \right\}, \quad (4.29)$$

and the related continuous (LP) relaxation is:

$$z_c(E) = \min \left\{ \sum_{j \in J} x_j : \quad \sum_{j \in J} a_{ij} x_j \geq b_i, \quad i \in I, \quad x_j \in \mathbb{R}_+, \quad j \in J \right\}. \quad (4.30)$$

Defining  $A := (a^j)_{j \in J}$  and  $e := (1, \dots, 1)^\top \in \mathbb{R}^n$ , we can formulate problem (4.29) in a compact form:

$$z^*(E) = \min \{ e^\top x : Ax \geq b, x \in \mathbb{Z}_+^n \}. \quad (4.31)$$

Related to instance  $E = (m, \ell, L, b)$  of the 1CSP, the ILP instance  $E' = (m, n, A, c, b)$  (with  $c := e$ ) of model (4.31) is uniquely determined, but the converse does not hold.

For a general ILP of form (4.31) and its LP relaxation

$$z_c(E) = \min\{c^\top x : Ax \geq b, x \geq 0, x \in \mathbb{R}^n\},$$

Baum and Trotter [4] introduced the integer and the integer round-up property.

**Definition 4.2** An integer linear programming problem  $P$  possesses the

- *Integer Property (IP)* if  $z^*(E) = z_c(E)$  holds for all instances  $E \in P$ ;
- *Integer Round-Up Property (IRUP)* if  $z^*(E) = \lceil z_c(E) \rceil$  holds for all  $E \in P$ ;
- *Modified Integer Round-Up Property (MIRUP)*  
if  $z^*(E) \leq \lceil z_c(E) \rceil + 1$  holds for all  $E \in P$ .

It is known that problem  $P$  possesses the IP if and only if the coefficient matrix of each instance  $E$  of  $P$  is *total unimodular* [22]. For example, the *max-flow problem* and the *transportation problem* possess the IP.

In all numerical experiments concerning the 1CSP, it was observed that the difference between the optimal values  $z^*(E)$  of (4.29) and  $z_c(E)$  of (4.30) is smaller than 2. Up to date, there is not known any instance  $E$  of the 1CSP with *gap*

$$\Delta(E) := z^*(E) - z_c(E) \geq 2.$$

Therefore, already in 1992, the following conjecture was formulated in [29].

**Conjecture** *The one-dimensional cutting stock problem possesses the MIRUP.*

Consequently, an instance  $E$  has the IRUP if  $\Delta(E) < 1$ , and the MIRUP if  $\Delta(E) < 2$ .

Whereas the proof of optimality of a feasible solution  $x^*$  of a 1CSP instance  $E$  which has IRUP, can be done by help of the LP relaxation, since  $z^*(E) = \lceil z_c(E) \rceil$ , in the case of a non-IRUP instance proving optimality is much harder. To this end, branch-and-bound approaches, or cutting plane methods, or their combination, called branch-and-cut, have to be applied, see e.g. [5, 8].

In the sequel, we present some results concerning the IRUP and MIRUP issue of the 1CSP. Let  $\mathsf{P}$  denote the set of all instances of the 1CSP, and  $\mathsf{M}^* \subset \mathsf{P}$  and  $\mathsf{M} \subseteq \mathsf{P}$  the sets of instances having IRUP or MIRUP, respectively. Those instances which belong to the *divisible case* define a subproblem of the 1CSP which is important for our investigations. Instance  $E := (m, \ell, L, b)$  belongs to the divisible case if  $L/\ell_i \in \mathbb{Z}$  for all  $i \in I$ . We recall, a pattern  $a \in \mathbb{Z}_{+}^m$  is *maximal* if  $0 \leq L - \ell^\top a < \ell_m$ , and *elementary* if  $a$  is a positive multiple of a unit-vector. i.e., if only items of a single type are contained. Furthermore, pattern  $a$  is said to be *proper* with respect to a given demand vector  $b$  if  $a_i \leq b_i$  holds for all  $i \in I$ .

### 4.11.2 Transformations

As it is well known, model (4.31) of the 1CSP is equivalent (with respect to the optimal value) to model

$$z^* = \min\{e^\top x : Ax = b, x \in \mathbb{Z}_+^n\}, \quad (4.32)$$

if  $A$  contains all feasible patterns (Exercise 4.2). Helpful for our study of the 1CSP with respect to the IRUP and the MIRUP is the consideration of a *residual instance*. Let  $E = (m, \ell, L, b)$  be an instance of the 1CSP and  $x^c = x^c(E)$  a solution of the LP relaxation of (4.32), i.e. of

$$z_c = \min\{e^\top x : Ax = b, x \in \mathbb{R}_+^n\}. \quad (4.33)$$

Then, the instance  $\bar{E} := (m, \ell, L, b - A[x^c])$  constitutes a residual instance of  $E$ .

**Definition 4.3** Instance  $E = (m, \ell, L, b)$  of the 1CSP is called *residual instance* if a solution  $x^c$  of (4.33) exists with  $0 \leq x^c < e$ , i.e., with  $0 \leq x_j^c < 1$  for all  $j \in J$ .

Typically, a residual instance is an instance of the bin packing problem so that solution methods for the BPP can be used to solve it, for instance approximately by fast heuristics as presented in Sect. 4.4. Let  $P_{\text{res}}$  denote the set of residual instances.

**Proposition 4.9** Let  $E \in P$  and let  $\bar{E}$  be a residual instance of  $E$ . Then:

- (a)  $\bar{E} \in M^* \Rightarrow E \in M^*$ ,
- (b)  $\bar{E} \in M \Rightarrow E \in M$ .

Exercise 4.10 asks for the proof of the proposition.

**Corollary 4.10** In order to prove the MIRUP for a certain problem  $P$ , it is sufficient to show the MIRUP for all residual instances of  $P$ .

### 4.11.3 Subproblems Possessing the MIRUP

Without loss of generality, we assume in this subsection that every solution of the LP relaxation of a residual instance has only components less than 1. Related to the instance  $E = (m, \ell, L, b)$  of the 1CSP and a pattern  $a^j$ , we define some characteristic values:

$$k_i := \left\lfloor \frac{L}{\ell_i} \right\rfloor, \quad i \in I, \quad \kappa := \left( \frac{1}{k_1}, \dots, \frac{1}{k_m} \right)^\top, \quad \gamma := \kappa^\top b, \quad \omega_j := \kappa^\top a^j, \quad \varrho_j := \frac{\ell^\top a^j}{L}.$$

Obviously, the objective value for the 1CSP equals  $\gamma$  if only maximal elementary patterns are used. Therefore, we have

$$z_c(E) \leq \gamma \quad \text{for all } E \in \mathcal{P}.$$

The value  $\omega_j$  is called the *density* and  $\varrho_j$  the *degree of utilization* of pattern  $a^j, j \in J$ .

**Proposition 4.11** *Let positive integer numbers  $k_i$  and  $b_i, i \in I$ , be given with  $b_i \leq k_i, i \in I$ . Then, there exist  $\lceil \gamma \rceil + 1$  nonnegative integer vectors  $a^j = (a_{1j}, \dots, a_{mj})^T$ ,  $j = 1, \dots, \lceil \gamma \rceil + 1$  with*

$$\gamma = \sum_{i=1}^m \frac{b_i}{k_i}, \quad \sum_{j=1}^{\lceil \gamma \rceil + 1} a_{ij} = b_i, \quad i = 1, \dots, m, \quad \sum_{i=1}^m \frac{a_{ij}}{k_i} \leq 1, \quad j = 1, \dots, \lceil \gamma \rceil + 1.$$

*Proof* We prove the proposition using the induction principle for  $n = 1, \dots, m$ . Obviously, the proposition holds in the case  $n = 1$  with  $a_{11} = b_1, a_{12} = 0$ . Let

$$\gamma_n = \left\lceil \sum_{i=1}^n \frac{b_i}{k_i} \right\rceil \quad \text{for } n \in \{1, \dots, m\}.$$

Induction step „ $n \Rightarrow n + 1$ “:

**Case (a):** Let  $k_{n+1} = k_r$  for some  $r \in \{1, \dots, n\}$ . Then, we define  $b_r^{(1)} := b_r + b_{n+1}, b_i^{(1)} := b_i, i = 1, \dots, n, i \neq r$ , and  $b_{n+1}^{(1)} := 0$ . If  $b_r^{(1)} \leq k_r$ , then the proposition holds. Otherwise, we have  $b_r^{(1)} \leq 2k_r$ , and with  $b_r^{(2)} := b_r^{(1)} - k_r, b_r^{(2)} \leq k_r$ , we obtain  $\lceil \sum_{i=1, i \neq r}^n b_i/k_i + b_r^{(2)}/k_r \rceil = \gamma_{n+1} - 1$ . Because of  $b_i \leq k_i$  for all  $i \neq r$  and  $b_r^{(2)} \leq k_r$ , there exist  $\gamma_{n+1}$  nonnegative integer vectors  $a^j$  for the instance  $(m, \ell, L, b - k_r e^r)$ , and with  $a_{r, \gamma_{n+1}+1} := k_r, a_{i, \gamma_{n+1}+1} := 0, i \neq r$ , the assertion is proved for this case.

**Case (b):** Due to case (a), we can assume  $1 \leq k_1 < k_2 < \dots < k_{n+1}$ . Therefore,  $k_{n+1} \geq n + 1$ . Because of  $\gamma_n \leq \gamma_{n+1}$ , there exist  $\gamma_{n+1} + 1$  nonnegative  $n$ -dimensional integer vectors  $a^j = (a_{1j}, \dots, a_{nj})^T$  with

$$\sum_{j=1}^{\gamma_{n+1}+1} a_{ij} = b_i, \quad i = 1, \dots, n, \quad \sum_{i=1}^n \frac{a_{ij}}{k_i} \leq 1, \quad j = 1, \dots, \gamma_{n+1} + 1.$$

Now, we consider these vectors as  $(n + 1)$ -dimensional vectors setting  $a_{n+1,j} := \lfloor (1 - \sum_{i=1}^n a_{ij}/k_i) k_{n+1} \rfloor$ . Consequently, we obtain

$$1 - \frac{1}{k_{n+1}} < \sum_{i=1}^{n+1} \frac{a_{ij}}{k_i} \leq 1, \quad j = 1, \dots, \gamma_{n+1} + 1, \quad b_{n+1} \leq k_{n+1} \left( \gamma_{n+1} - \sum_{i=1}^n \frac{b_i}{k_i} \right).$$

Furthermore, we have

$$\begin{aligned}
b_{n+1} - \sum_{j=1}^{\gamma_{n+1}+1} a_{n+1,j} &\leq k_{n+1} \left( \gamma_{n+1} - \frac{1}{k_{n+1}} \sum_{j=1}^{\gamma_{n+1}+1} a_{n+1,j} - \sum_{i=1}^n \frac{b_i}{k_i} \right) \\
&= k_{n+1} \left( \gamma_{n+1} - \frac{1}{k_{n+1}} \sum_{j=1}^{\gamma_{n+1}+1} a_{n+1,j} - \sum_{i=1}^n \left( \frac{1}{k_i} \sum_{j=1}^{\gamma_{n+1}+1} a_{ij} \right) \right) \\
&= k_{n+1} \left( \gamma_{n+1} - \sum_{j=1}^{\gamma_{n+1}+1} \sum_{i=1}^{n+1} \frac{a_{ij}}{k_i} \right) \\
&< k_{n+1} \left( \gamma_{n+1} - (\gamma_{n+1} + 1) \left( 1 - \frac{1}{k_{n+1}} \right) \right) \\
&= k_{n+1} \left( -1 + \frac{\gamma_{n+1}}{k_{n+1}} + \frac{1}{k_{n+1}} \right) \\
&\leq 1.
\end{aligned}$$

Because of  $b_{n+1} \in \mathbb{Z}$  and  $a_{n+1,j} \in \mathbb{Z}$ , we obtain  $\sum_{j=1}^{\gamma_{n+1}+1} a_{n+1,j} \geq b_{n+1}$ . Hence, we can find appropriate nonnegative integers  $a'_{n+1,j}$ ,  $j = 1, \dots, \gamma_{n+1} + 1$  with  $a'_{n+1,j} \leq a_{n+1,j}$  and  $\sum_{j=1}^{\gamma_{n+1}+1} a'_{n+1,j} = b_{n+1}$ . ■

The previous proposition can be generalized straightforwardly:

**Proposition 4.12** *Let positive integer numbers  $k_i$  and  $b_i$ ,  $i \in I$ , be given. Then, there exist  $\lceil \gamma \rceil + 1$  nonnegative integer vectors  $a^j = (a_{1j}, \dots, a_{mj})^T$  with*

$$\sum_{j=1}^{\lceil \gamma \rceil + 1} a_{ij} = b_i, \quad i = 1, \dots, m, \quad \sum_{i=1}^m \frac{a_{ij}}{k_i} \leq 1, \quad j = 1, \dots, \lceil \gamma \rceil + 1, \quad \gamma = \sum_{i=1}^m \frac{b_i}{k_i}.$$

Exercise 4.9 asks for a proof of Proposition 4.12.

**Corollary 4.13** *If  $E$  is an instance of the divisible case, then its gap  $\Delta(E)$  is less than 2. With other words, the divisible case subproblem of the 1CSP possesses the MIRUP.*

Moreover, Rietz [25] has proved that  $\Delta(E) < 7/5$  holds for all instances  $E$  of the divisible case which constitutes a tightening of Proposition 4.11. Furthermore, in [27] it is shown that

$$\Delta(E) < \max\{2, (m+2)/4\}$$

holds for any instance of the 1CSP.

In the following, we list some results proposed in [25, 27, 30], but omit sometimes to prove them if that would require an expensive case by case analysis.

**Proposition 4.14** Let  $E$  be an instance of the 1CSP, and let  $\gamma$  be defined as above. If  $\lceil z_c(E) \rceil = \lceil \gamma \rceil$  holds, then  $E$  possesses the MIRUP, i.e.  $E \in \mathbf{M}$ .

*Proof* Let  $\alpha_i := \lfloor b_i/k_i \rfloor$  and  $\bar{b}_i := b_i - \alpha_i k_i$ . Then,

$$\lceil z_c(E) \rceil = \lceil \gamma \rceil = \lceil \kappa^\top b \rceil = \lceil \sum_{i=1}^m b_i/k_i \rceil = \sum_{i=1}^m \alpha_i + \lceil \sum_{i=1}^m \bar{b}_i/k_i \rceil =: \Gamma_1 + \Gamma_2.$$

Because of construction, there exist due to Proposition 4.11,  $\lceil \Gamma_2 \rceil + 1$  non-negative integer vectors  $a^j$  with  $\sum_{j=1}^{\lceil \Gamma_2 \rceil + 1} a_{ij} = \bar{b}_i$ ,  $i \in I$ , and  $\sum_{i=1}^m a_{ij}/k_i \leq 1$ ,  $j = 1, \dots, \lceil \Gamma_2 \rceil + 1$ . Using  $\alpha_i$  corresponding elementary patterns the assertion is proved. ■

For an instance  $E = (m, \ell, L, b)$  of the 1CSP, a natural lower bound for the optimal value is given by  $lb_0(E) := \ell^\top b/L \leq z_c(E)$ , the *material bound* without rounding-up.

**Proposition 4.15** Let  $E \in \mathbf{P}$  and let  $\bar{E} \in \mathbf{P}_{\text{res}}$  be a corresponding residual instance. If  $lb_0(\bar{E}) \in [0, 7/4] \cup (2, 5/2] \cup (3, 13/4]$  holds, then  $E \in \mathbf{M}^*$  follows.

In Exercise 4.11 the statement has to be proved for the case  $lb_0(E) \leq 3/2$ .

**Proposition 4.16** Let  $E \in \mathbf{P}$  and let  $\bar{E} \in \mathbf{P}_{\text{res}}$  be a corresponding residual instance. If  $m \leq 6$  or  $lb_0(\bar{E}) \in [0, 19/4] \cup (5, 11/2] \cup (6, 25/4]$  are fulfilled, then  $E$  possesses the MIRUP, i.e.  $E \in \mathbf{M}$ .

Let  $v = v(x) := \sum_j \text{sign}(x_j)$  where  $\text{sign}(t) = 1$  for  $t > 0$  and  $\text{sign}(t) = 0$  for  $t = 0$ . If  $x$  is a basis solution of the LP relaxation (obtained by the simplex method), then, obviously, we have  $v(E) \leq m$ .

**Proposition 4.17** Let  $E \in \mathbf{P}$  and let  $\bar{E} \in \mathbf{P}_{\text{res}}$  be a corresponding residual instance. If  $\lceil z_c(\bar{E}) \rceil \geq v(x^c(\bar{E})) - 1$  is satisfied, then  $E \in \mathbf{M}$  follows.

Exercise 4.12 asks for a proof of this proposition. Using the abbreviation  $\omega_j = \kappa^\top a^j$ , we have the following relation for an LP solution  $x^c \in \mathbb{R}_+^n$ :

$$\gamma - z_c(E) \leq \sum_{i \in I} \frac{1}{k_i} \sum_{j \in J} a_{ij} x_j^c - \sum_{j \in J} x_j^c = \sum_{j \in J} x_j^c (\omega_j - 1).$$

Furthermore,  $\mu = \mu(E) := \max \{ \omega_j : j = 1, \dots, n \} \leq 1.7$  holds (Exercise 4.13).

**Proposition 4.18** Let  $E \in \mathbf{P}$  and let  $\bar{E} \in \mathbf{P}_{\text{res}}$  be a corresponding residual instance. In case of  $\mu(\bar{E}) \leq 1$ , then  $E \in \mathbf{M}$  and  $z_c(\bar{E}) = \gamma(\bar{E})$  follow.

In Exercise 4.14 the proof of the proposition has to be done.

Lower bounds for  $z^*(E)$  with  $E = (m, \ell, L, b) \in \mathbf{P}$  can be obtained by aggregation as well. Let  $\lambda \in R_+^m$ , then

$$z_\lambda = \min \{ e^\top x : \lambda^\top A x \geq \lambda^\top b, x \geq 0 \} \quad (4.34)$$

is a relaxation of the 1CSP (Exercise 4.15), and hence,  $\lceil z_\lambda \rceil$  is a lower bound of  $z^*(E)$ . In particular, choosing  $\lambda := \ell/L$ , we obtain with  $\ell^\top A \leq L e^\top$  the material

bound (without rounding-up)

$$B_1(E) := lb_0(E) = \ell^\top b / L.$$

For  $\lambda := \kappa/\mu$  another bound results:

$$B_2(E) := \gamma/\mu \quad \text{with } \gamma = \kappa^\top b.$$

**Example 4.5** For the instance  $E = (2, (44, 33)^\top, 142, (2, 3)^\top)$  we have  $z_C(E) \geq B_1(E) = 187/142 \approx 1.317$ ,  $\mu = 1$  and  $z_C(E) \geq B_2(E) = 17/12 \approx 1.417$ .

Hence, for  $E' := (2, (44, 33)^\top, 142, (20, 30)^\top)$ , we obtain  $z^*(E') \geq \max\{14, 15\}$ . ■

**Proposition 4.19** *Let  $E \in \mathbf{P}$ . If  $\lceil \max\{B_1(E), B_2(E)\} \rceil = \lceil \gamma \rceil$  holds, then  $E \in \mathbf{M}$  is fulfilled.*

In Exercise 4.16 a proof of the proposition has to be found. Corresponding to  $E = (m, \ell, L, b) \in \mathbf{P}$  and  $z \in \mathbb{R}$  with  $z \geq k_1 = \lfloor L/\ell_1 \rfloor$ , we define  $\theta(z) := \max\{i : k_i \leq \lceil z \rceil, i \in I\}$ .

**Proposition 4.20** *Let  $E = (m, (\ell_1, \dots, \ell_m)^\top, L, (b_1, \dots, b_m)^\top) \in \mathbf{P}_{\text{res}}$  and let  $x^s$  be a feasible solution of the LP relaxation with value  $z$  and  $z \geq k_1$ . Moreover, let  $\sum_{i=\theta(z)+1}^m b_i > 0$ . Then we have:*

$$E \in \mathbf{M} \iff \bar{E} := (\theta(z), (\ell_1, \dots, \ell_{\theta(z)})^\top, L, (b_1, \dots, b_{\theta(z)})^\top) \in \mathbf{M}.$$

Moreover, let  $E = (m, \ell, L, b) \in \mathbf{P}$  and  $\bar{E} = (\theta(z), \bar{\ell}, \bar{L}, \bar{b}) \in \mathbf{P}$  be defined as in Proposition 4.20, and let  $z$  denote the objective value of a feasible solution  $x^s$  of the LP relaxation of  $E$ . As a consequence of Propositions 4.17 and 4.20, we obtain:

**Corollary 4.21** *For investigating the 1CSP with respect to the MIRUP it is sufficient to consider residual instances with  $k_i \leq \lceil z \rceil \leq m - 2$  for all  $i$  as well as the instances with  $k_1 > z_c(E)$ .*

#### 4.11.4 Further Relaxations

Besides the continuous relaxation of the 1CSP, stronger relaxations can be obtained by adding further restrictions. On the one hand, we can restrict the pattern set to only contain proper patterns, and on the other hand, we can add an upper bound  $u_j$  to the frequency  $x_j$  pattern  $a^j = (a_{1j}, \dots, a_{mj})^\top$  of  $E = (m, \ell, L, b)$  can be used where

$$u_j := \min \left\{ \lfloor b_i/a_{ij} \rfloor : a_{ij} > 0, i = 1, \dots, m \right\}, \quad j = 1, \dots, n.$$

Thus, we have  $u_j = 0$  if and only if  $a^j \not\leq b$ . This implies

$$z_c(E) \leq z_p(E) \leq z_u(E) \leq z^*(E) \quad (4.35)$$

where

$$z_p(E) := \min \left\{ e^\top x : \sum_{j \in J} a^j x_j = b, x_j \geq 0, a^j \not\leq b \Rightarrow x_j = 0, j \in J \right\} \quad (4.36)$$

and

$$z_u(E) := \min \left\{ e^\top x : \sum_{j \in J} a^j x_j = b, 0 \leq x_j \leq u_j, j \in J \right\}. \quad (4.37)$$

In particular, if  $E$  is a residual instance, we can observe some strict inequalities in (4.35). Problem (4.36) is known as the *proper relaxation* of the 1CSP. Since problem (4.36) is also an LP problem, it can be solved by the simplex method and the column generation technique. Therein, the condition  $a^j \leq b$  has to be regarded in the slave problems as well as when choosing the initial pattern pool.

Obviously, if all patterns of instance  $E = (m, \ell, L, b)$  are proper, then  $z_c(E) = z_p(E)$ . Let

$$\delta := L - \max\{\ell^\top a : \ell^\top a \leq L, a \leq b, a \in \mathbb{Z}_+^m\},$$

and let  $a^*$  be a proper pattern with  $\delta = L - \ell^\top a^*$ . Furthermore, let  $x^p$  be a solution of (4.36), i.e.  $z_p(E) = e^\top x^p$ . Then we have:

**Proposition 4.22** *Let  $z_0 := \lceil z_c(E) \rceil$ . If  $\ell^\top (b - a^*) > (z_0 - 1)L$  holds, then  $z_p(E) > z_0$  follows.*

A proof can be found in [23].

The *upper bound relaxation* (4.37) can also be solved using the simplex method and column generation if a special technique, called *upper bound technique* (handling of box constraints), is applied.

In order to illustrate the relaxations, we consider

**Example 4.6** Let  $E = (3, (15, 10, 6)^\top, 30, (1, 2, 4)^\top)$ . Then, we have  $z_c(E) = 59/30 < 2 < z_p(E) = z_u(E) = 2.2$  and  $\lceil z_u(E) \rceil = 3 = z^*(E)$ .

The slightly modified instance  $E(b')$  with  $b' = (3, 5, 9)^\top$  yields  $z_c(E) = 149/30 = z_p(E) < 5 < z_u(E) = 5.2$  and  $\lceil z_u(E) \rceil = 6 = z^*(E)$ .

Note that there remain instances  $E \notin M^*$  where the upper bound relaxation does not provide the optimal value of the CSP. Such an instance is  $E = (4, (150, 100, 60, 1)^\top, 302, (3, 5, 9, 3)^\top)$  with  $z_u(E) < 5$  and  $z^*(E) = 6$ . ■

### 4.11.5 MIRUP and Higher-Dimensional Cutting Stock Problems

For the two- and three-dimensional cutting stock problem (2CSP and 3CSP) only few results concerning the quality (tightness) of relaxations are known so far. For the 2-stage two-dimensional guillotine cutting stock problem, exact case (E2-CSP, Sect. 6.3), where additionally the cuts of the first stage either have to be all horizontal, or all are vertical, referred to as RE2-CSP, it is shown in [28] that

$$\Delta(\text{RE2-CSP}) > m/3$$

holds. That means, there exists a sequence  $(E_m)_{m \in \mathbb{N}}$  of instances of the RE2-CSP such that the gap  $\Delta(E_m)$  increases with the number  $m$  of item types.

Moreover, in [28] instances of the nonexact case of the 2-stage guillotine CSP (N2-CSP) are presented having a gap larger than 2, but smaller than 3.

Considering the 2-stage two-dimensional guillotine cutting stock problem as a combination of two independent one-dimensional cutting stock problems, for which no instance with  $\Delta > 1.2$  is known, the following conjecture is motivated:

**Conjecture** *For each instance  $E$  of the N2-CSP we guess:  $z^*(E) \leq \lceil z_c(E) \rceil + 2$ .*

Investigations in this respect and concerning related issues still have to be done.

### 4.11.6 Resume

Summarizing, we have to state that a lot of questions remain open with respect to an absolute bound for the gaps. Concerning the 1CSP, the largest gap  $\Delta(E) = z^*(E) - z_c(E)$  among all instances  $E$  of the *divisible case* is equal to  $137/132 < 1.0379$  obtained for the instance  $E = (3, (44, 33, 12)^\top, 132, (2, 3, 6)^\top)$  where  $z^*(E) = 3$  and  $z_c(E) = 259/132$ .

Moreover, 1CSP-instances having a larger gap and methods to construct them are presented in [25, 26]. Up to now, there is not known any instance  $E$  of the 1CSP having a gap larger than 1.2. An instance  $E^*$  with  $\Delta(E^*) = 1.2$ , proposed in [25], is the following:

$$m = 32, \quad L = 1500, \quad b_{12} = 2, b_{21} = 5, b_i = 1 \text{ otherwise},$$

$$\ell = (1214, 1210, 1208, 910, 906, 904, 774, 770, 768, 504, 503, 500, 498, 494, 368,$$

$$366, 365, 364, 363, 362, 300, 298, 297, 296, 295, 294, 148, 146, 145, 144, 143, 142)^\top.$$

## 4.12 Exercises

**Exercise 4.1** Show that the minimization of the number of stock lengths needed in an optimal solution of the 1CSP model (4.3a) is equivalent to the minimization of waste provided that overproduction is not allowed (surplus pieces are considered to be waste).

**Exercise 4.2** Let index set  $J$  represent all feasible (cutting) patterns. Prove the equivalence of models (4.4) and (4.5) with respect to optimal values. Moreover, construct an instance of the 1CSP which proves that the equivalence does not hold if this condition is violated.

**Exercise 4.3** Develop a recurrence formula to determine the number of all patterns of the 1CSP.

**Exercise 4.4** Prove that  $\bar{c} = \min_{j \in J} \{c_j - d^\top a^j\}$  holds if  $\bar{c} = \min_{j \in N} \{c_j - d^\top a^j\} \leq 0$  is fulfilled.

**Exercise 4.5** As an extension of Example 4.2 (p. 81), determine all patterns and solve the enlarged problem with the revised simplex method. Note that as initial matrix one can use the optimal basis matrix obtained in the example.

**Exercise 4.6** Show that instance  $E = (m, \ell, L, b)$  of the 1CSP has a solution if and only if  $\ell_i \leq L$  for  $i = 1, \dots, m$  is fulfilled.

**Exercise 4.7** Verify that the instance of the 1CSP with input data  $L = 30$ ,  $m = 3$ ,  $\ell = (15, 10, 6)^\top$ , and  $b = (1, 2, 4)^\top$  does not have a solution with only two patterns. What is the optimal value of the LP relaxation?

**Exercise 4.8** Solve the instance of the 1CSP with the input data  $L = 132$ ,  $m = 3$ ,  $\ell = (44, 33, 12)^\top$ , and  $b = (2, 3, 6)^\top$ .

**Exercise 4.9** Prove: Let positive integer numbers  $k_i$  and  $b_i$ ,  $i \in I = \{1, \dots, m\}$ , be given. Then, there exist  $\lceil \gamma \rceil + 1$  nonnegative integer vectors  $a^j = (a_{1j}, \dots, a_{mj})^\top$  with

$$\sum_{j=1}^{\lceil \gamma \rceil + 1} a_{ij} = b_i, \quad i = 1, \dots, m, \quad \sum_{i=1}^m \frac{a_{ij}}{k_i} \leq 1, \quad j = 1, \dots, \lceil \gamma \rceil + 1, \quad \gamma = \sum_{i=1}^m \frac{b_i}{k_i}.$$

**Exercise 4.10** Prove Proposition 4.9 (p. 109).

**Exercise 4.11** Prove Proposition 4.15 (p. 112) for the case  $lb_0(\bar{E}) \leq 3/2$ .

**Exercise 4.12** Prove Proposition 4.17 (p. 112).

**Exercise 4.13** Show:  $\max \{\omega_j : j = 1, \dots, n\} \leq 1.7$ .

**Exercise 4.14** Prove Proposition 4.18 (p. 112).

**Exercise 4.15** Show that problem (4.34) is a relaxation of the 1CSP for  $\lambda \in \mathbb{R}_+^m$  (p. 112).

**Exercise 4.16** Prove Proposition 4.19 (p. 113).

**Exercise 4.17** Construct the graph  $G'$  of the arcflow model for the instance in Example 4.1 (p. 77) and compute the flow value for each arc in a solution of the 1CSP.

**Exercise 4.18** Compute the bound  $lb_s(E)$  in (4.16) for the instance  $E$  of Example 3.1 (p. 53).

**Exercise 4.19** Proof Proposition 4.3: For any instance  $E = (m, \ell, L, b)$  of the 1CSP with integer input data, we have  $lb_M(E) \leq z^*(E) \leq 2 \cdot lb_M(E)$  where  $z^*(E)$  denotes the optimal value of instance  $E$ .

## 4.13 Solutions

**To Exercise 4.1** Let  $\bar{x}$  denote a solution of model (4.3), i.e., we have  $\sum_{j \in J} a_{ij}\bar{x}_j \geq b_i$  for  $i \in I$  and  $\sum_{j \in J} \bar{x}_j \leq \sum_{j \in J} x_j$  for all feasible solutions  $x$ . We consider the target function  $z_A(x)$  of waste minimization, then

$$\begin{aligned} z_A(x) &= \sum_{j \in J} \left( L - \sum_{i \in I} \ell_i a_{ij} \right) x_j + \sum_{i \in I} \ell_i \left( \sum_{j \in J} a_{ij} x_j - b_i \right) = L \sum_{j \in J} x_j - \sum_{i \in I} \ell_i b_i \\ &\geq L \sum_{j \in J} \bar{x}_j - \sum_{i \in I} \ell_i b_i = z_A(\bar{x}). \end{aligned}$$

Therefore, vector  $\bar{x}$  is a solution for waste minimization, too. This does not hold, in general, if the surplus pieces are not considered to be waste.

**To Exercise 4.2** If problem  $(P_=)$ , defined in (4.5), is solvable (this holds if and only if  $\ell_i \leq L$  for all  $i \in I$ ), then problem  $(P_\geq)$ , defined in (4.4), is solvable, too. Contrarily, let  $x$  be a feasible solution of  $(P_\geq)$ . Then we can obtain a feasible solution  $\bar{x}$  of  $(P_=)$  by the following construction: If  $Ax \neq b$ , then there exists an index  $k \in I$  with  $[Ax]_k \geq b_k + 1$  and an index  $p \in J$  with  $a_{kp}x_p \geq 1$ . ( $J$  represents all columns of  $A$ .) Since  $A$  consists of all patterns, there exists  $q \in J$  with  $a^q = a^p - e^k$  where  $e^k$  denotes the  $k$ th unit vector.

Since  $x_p \geq 1$ , we can define  $\bar{x}_q := x_q + 1$ ,  $\bar{x}_p := x_p - 1$ ,  $\bar{x}_j := x_j$  for  $j \in J \setminus \{p, q\}$  and obtain a feasible solution of  $(P_\geq)$  with  $[A\bar{x}]_i = [Ax]_i \geq b_i$  for  $i \neq k$  and  $[A\bar{x}]_k = [Ax]_k - 1 \geq b_k$ .

By means of a finite number of such steps, finally we obtain a solution  $\bar{x}$  with  $[A\bar{x}]_i = b_i$  for all  $i \in I$ . Thus,  $(P_=)$  is solvable, too. Obviously, the two optimal values are equal.

Let  $L = 10$ ,  $\ell = (9, 1)^\top$  and  $b = (1, 2)^\top$ .

For  $A = \begin{pmatrix} 1 & 0 \\ 1 & 10 \end{pmatrix}$ , we have:  $(P_=)$  is not solvable, but  $(P_\geq)$  has optimal value 2.

**To Exercise 4.3** Let  $\ell_1, \dots, \ell_m$  and  $L$  be given. By  $\lambda_i(y)$  we denote the number of all nonnegative integer vectors  $a$  with  $\ell^\top a \leq y$  and  $a_j = 0$  for  $j > i$ ,  $i \in I$ ,  $y = 0, 1, \dots, L$ , including the zero-vector. Thus, we have  $\lambda_1(y) = \lfloor y/\ell_1 \rfloor + 1$  for  $y = 0, 1, \dots, L$ . Setting  $a_i := j$  with  $j = 0, \dots, \lfloor y/\ell_i \rfloor$ , we obtain different patterns. Hence, the total number of patterns results to  $\lambda_i(y) = \sum_{j=0}^{\lfloor y/\ell_i \rfloor} \lambda_{i-1}(y - j\ell_i)$  for  $y = 0, 1, \dots, L$ ,  $i = 2, \dots, m$ .

**To Exercise 4.4** Let  $A_B = (a^j)_{j=1, \dots, m}$  (without loss of generality we use an appropriate renumbering of the columns). For each column  $a^j$  of  $A_B$ , i.e.  $j \in J \setminus N$ , we have  $A_B^{-1} a^j = e^j$  where  $e^j$  denotes the  $j$ th unit vector. Then we obtain  $\bar{c}_j = c_j - c_B^\top A_B^{-1} a^j = c_j - c_B^\top e^j = c_j - c_j = 0 \geq \bar{c}$  for the transformed (reduced) cost coefficient of  $a^j$ .

**To Exercise 4.5** Additionally, we obtain the maximal patterns  $a^{12} = (0, 0, 0, 2)^\top$  and  $a^{13} = (0, 0, 1, 1)^\top$  as well as further non-maximal patterns. Their reduced cost coefficients are  $\bar{c}_{12} = -1$  and  $\bar{c}_{13} = -1/2$ .

| $ST_2$   | $A_B^{-1}$ |    |      |   | $A_B^{-1} b$ | $-A_B^{-1} a^{12}$ | $ST_3$   | $A_B^{-1}$ |      |      |     | $A_B^{-1} b$ |
|----------|------------|----|------|---|--------------|--------------------|----------|------------|------|------|-----|--------------|
| $x_1$    | 0          | 0  | 1/2  | 0 | 15           | 0                  | $x_1$    |            | 1/2  |      |     | 15           |
| $s_2$    | -1         | -1 | 1/2  | 2 | 195          | -4                 | $x_{12}$ | -1/4       | -1/4 | 1/8  | 1/2 | 195/4        |
| $x_6$    | 1/2        | 0  | -1/4 | 0 | 15/2         | 0                  | $x_6$    | 1/2        |      | -1/4 |     | 15/2         |
| $x_2$    | -1/2       | 0  | 1/4  | 1 | 225/2        | -2                 | $x_2$    |            | 1/2  |      |     | 15           |
| $d^\top$ | 0          | 0  | 1/2  | 1 | 135          | -1                 | $d^\top$ | 1/4        | 1/4  | 3/8  | 1/2 | 345/4        |

Tableau  $ST_3$  already provides a solution since now all reduced cost coefficients are nonnegative.

**To Exercise 4.6** If the 1CSP is solvable, then there exists at least one feasible pattern for each item type. Hence, we have  $\ell_i \leq L$  for all  $i = 1, \dots, m$ . Otherwise, if  $\ell_i \leq L$  for  $i = 1, \dots, m$ , then there exists a feasible solution of the 1CSP, e.g., determined by elementary patterns. Since the target function is bounded from below and the optimal value has to be integer, the solvability of the problem results.

**To Exercise 4.7** Because of  $\sum_{i=1}^3 \ell_i b_i = 59$ , we have  $z^* \geq 2$ .

Supposing there exists a solution with  $z = 2$ , then it has to contain one trimless pattern and one with 1 unit waste. It is easy to verify that a pattern  $a \in \mathbb{Z}_+^3$  with  $29 \leq \ell^\top a \leq 30 = L$  and  $a \leq b$  does not exist. Therefore, more than 2 patterns are needed, i.e.  $z^* \geq 3$ .

In the LP relaxation, also the patterns  $(2, 0, 0)^\top$ ,  $(0, 3, 0)^\top$ , and  $(0, 0, 5)^\top$  are available which form its solution with value  $\frac{1}{2} + \frac{2}{3} + \frac{4}{5} = 59/30 < 2$ .

For  $b' = b + k \cdot (2, 3, 5)^\top$ ,  $k \in \mathbb{Z}_+$ , we obtain the same gap between  $z^*(b') = 3 + 3k$  and the LP bound  $\sum_{i=1}^3 \ell_i b'_i / L = 59/30 + 3k$ .

**To Exercise 4.8** Matrix  $A_B = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 11 \end{pmatrix}$  represents an optimal basis matrix since the knapsack problem

$$z = \max\left\{\frac{1}{132}(44a_1 + 33a_2 + 12a_3) : 44a_1 + 33a_2 + 12a_3 \leq 132, a_i \in \mathbb{Z}_+, i \in I\right\}$$

has the optimal value 1 (thus  $\bar{c} = 0$ ). We have  $z = c_B^\top A_B^{-1} b = 259/132$ . Therefore, at least 2 stock lengths are necessary. Because of  $2L - \ell^\top b = 5$ , a solution with  $z = 2$  would have exactly 5 units of waste. Since there does not exist any pattern  $a \in \mathbb{Z}_+^3$  with  $\ell^\top a \leq L$ ,  $a \leq b$ , and  $L - \ell^\top a \leq 5$ , we obtain,  $z^* \geq 3$ .

**To Exercise 4.9** Because of  $b_i = \lfloor b_i/k_i \rfloor k_i + \bar{b}_i$  with  $\bar{b}_i < k_i$ , the assumptions of Proposition 4.11 are fulfilled for  $\bar{b}_1, \dots, \bar{b}_m$ . Consequently, there exist  $\lceil \bar{\gamma} \rceil + 1$  nonnegative integer vectors  $a^j = (a_{1j}, \dots, a_{mj})^\top$  with  $\sum_{j=1}^{\lceil \bar{\gamma} \rceil + 1} a_{ij} = \bar{b}_i$ ,  $i = 1, \dots, m$  and  $\sum_{i=1}^m a_{ij}/k_i \leq 1$ ,  $j = 1, \dots, \lceil \bar{\gamma} \rceil + 1$ , where  $\bar{\gamma} = \sum_{i=1}^m \bar{b}_i/k_i$ . Since  $\lceil \gamma \rceil = \sum_{i=1}^m \lfloor b_i/k_i \rfloor + \lceil \bar{\gamma} \rceil$ , the assertion follows.

**To Exercise 4.10** Let  $x^c$  be a solution of the LP relaxation of  $E$ , and let  $\underline{x} := \lfloor x^c \rfloor$ ,  $e = (1, \dots, 1)^\top$ . Then we have

$$z^*(E) \leq e^\top \underline{x} + z^*(\bar{E}) \leq e^\top \underline{x} + \lceil z_c(\bar{E}) \rceil + 1 = \lceil e^\top \underline{x} + z_c(\bar{E}) \rceil + 1 = \lceil z_c(E) \rceil + 1.$$

**To Exercise 4.11** If there exists a piece  $i$  with  $\ell_i \geq L/2$ , then it defines the first pattern. The remaining pieces define the second pattern. Otherwise, some pieces can be combined with total length at least  $L/2$ . They define the first and the remaining the second pattern. Since more than one stock length is needed, the optimal value is equal to 2.

**To Exercise 4.12** We simply obtain an integer solutions by rounding up.

**To Exercise 4.13** For  $k = 1, 2, \dots$  let  $I_k := \{i \in I : \lfloor L/\ell_i \rfloor = k\}$ . Thus, for  $i \in I_k$  we have  $L/(k+1) < \ell_i \leq L/k$ . We consider now the optimization problem

$$\omega = \max \sum_{k=1}^{\infty} \sum_{i \in I_k} \frac{a_i}{k} \quad \text{s. t.} \quad \sum_{k=1}^{\infty} \sum_{i \in I_k} \ell_i a_i \leq L, \quad a_i \in \mathbb{Z}_+ \quad \forall i.$$

In the case  $\sum_{i \in I_1} a_i = \sum_{i \in I_2} a_i = 1$ , we obtain  $a_i = 0$  for  $i \in I_3 \cup I_4 \cup I_5 \cup I_6$  and

$$\begin{aligned}\omega &= \max 1 + \frac{1}{2} + \sum_{k=7}^{\infty} \sum_{i \in I_k} \frac{a_i}{k} \quad \text{s. t. } \sum_{k=7}^{\infty} \sum_{i \in I_k} \ell_i a_i < L - \frac{L}{2} - \frac{L}{3} = \frac{L}{6}, \quad a_i \in \mathbb{Z}_+ \quad \forall i. \\ \frac{L}{6} &> \sum_{k=7}^{\infty} \sum_{i \in I_k} \ell_i a_i = L \sum_{k=7}^{\infty} \sum_{i \in I_k} \frac{a_i}{L/\ell_i} > L \sum_{k=7}^{\infty} \sum_{i \in I_k} \frac{a_i}{k+1} = L \sum_{k=7}^{\infty} \sum_{i \in I_k} \frac{a_i}{k} \frac{k}{k+1} \\ &\geq L \sum_{k=7}^{\infty} \sum_{i \in I_k} \frac{a_i}{k} \cdot \frac{7}{8} \quad \Rightarrow \quad \sum_{k=7}^{\infty} \sum_{i \in I_k} \frac{a_i}{k} < \frac{1}{6} \cdot \frac{8}{7} < 0.2.\end{aligned}$$

Hence,  $\omega < 1.7$  is valid in this case. Analogously, we can show that the assertion holds also in the two cases  $\sum_{i \in I_1} a_i = 1$ ,  $\sum_{i \in I_2} a_i = 0$  and  $\sum_{i \in I_1} a_i = 0$ .

**Tip** Compare this assertion with those concerning the *Salzer-series* in Sect. 7.6.1.

**To Exercise 4.14** Because of

$$\gamma - z_c(E) \leq \sum_i \frac{1}{k_i} \sum_j a_{ij} x_j^c - \sum_j x_j^c = \sum_j x_j^c \left( \sum_i \frac{a_{ij}}{k_i} - 1 \right) = \sum_j x_j^c (\omega_j - 1)$$

we have  $\gamma - z_c(E) \leq 0$ . With  $z_c(E) \leq \gamma$  the assumptions of Proposition 4.14 are satisfied.

**To Exercise 4.15** Since  $x \geq 0$  and  $A \geq 0$ , we have  $Ax \geq 0$ . Therefore, because of  $Ax \geq b$  and  $\lambda \geq 0$ , it follows that any feasible solution  $x$  of the 1CSP fulfills the inequality  $\lambda^\top Ax \geq \lambda^\top b$ .

**To Exercise 4.16** Choosing  $\lambda := \ell/L$  we obtain with  $\ell^\top A \leq Le^\top$  the material bound  $B_1(E) := \ell^\top b/L$ . If we take  $\lambda := \kappa/\mu$ , then we obtain the bound  $B_2(E) := \gamma/\mu$  with  $\gamma = \kappa^\top b$ . Therewith, the assumptions of Proposition 4.14 are satisfied.

**To Exercise 4.17** All vertices  $p \in V$  (all potential allocation points) as well as all arcs  $(p, q) \in E'$  are stated within the table. The entry of an arc represents its flow value. No entry means, that the arc does not belong to  $E'$ .

| $k$      | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $p$      | 0  | 20 | 22 | 25 | 26 | 40 | 42 | 44 | 45 | 46 | 47 | 48 | 50 | 51 | 52 | 60 | 62 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| $\mu(p)$ | 0  | 4  | 3  | 2  | 1  | 4  | 4  | 3  | 4  | 4  | 3  | 3  | 2  | 2  | 1  | 4  | 4  | 4  | 4  | 3  | 4  | 4  | 3  | 3  |
| 26       | 72 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 25       |    | 15 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       |    | 0  |    |    | 0  | 15 |    |    |    | 0  |    |    | 0  | 15 |    |    |    |    |    |    |    |    |    |    |
| 20       |    | 0  | 0  | 0  | 0  | 8  | 0  | 0  | 0  | 0  | 8  | 0  | 0  | 15 |    |    |    |    |    |    |    |    |    |    |
| $E_2$    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 49 | 0  | 0  | 0  | 0  | 8  | 0  | 0  | 0  |

**To Exercise 4.18** We obtain  $lb_s(E) = 5 = lb_s(E, 6)$ .

**To Exercise 4.19** Let  $x_j^*, a^j, j \in J^*$  represent a solution of  $E$ , i.e.  $z^* = \sum_{j \in J^*} x_j^*$ . Then, all but one pattern  $a^j$  have a length utilization of more than  $L/2$  since otherwise, a contradiction to the optimality of  $x^*$  would arise. Therefore, we have

$$\sum_{i \in I} \ell_i b_i = \sum_{j \in J^*} \sum_{i \in I} \ell_i a_{ij} x_j^* > \frac{L}{2} \left( \sum_{j \in J^*} x_j^* - 1 \right) = \frac{L}{2} (z^* - 1).$$

Hence,  $z^* < \frac{2}{L} \sum_{i \in I} \ell_i b_i + 1$ . Finally, since  $z^* \in \mathbb{Z}$ , we obtain  $z^* \leq \lceil \frac{2}{L} \sum_{i \in I} \ell_i b_i \rceil \leq 2lb_M$ .

## References

1. C. Alves, J.M.V. de Carvalho, A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Comput. Oper. Res.* **35**, 1315–1328, 2008.
2. B.S. Baker, A new proof for the first-fit decreasing bin-packing algorithm. *J. Algorithms* **6**, 49–70 (1985)
3. J. Bang-Jensen, R. Larsen, Efficient algorithms for real-life instances of the variable size bin packing problem. *Comput. Oper. Res.* **39**(11), 2848–2857 (2012)
4. S. Baum, L.E. Trotter Jr., Integer rounding for polymatroid and branching optimization problems. *SIAM J. Algebr. Discrete Methods* **2**(4), 416–425 (1981)
5. G. Belov, G. Scheithauer, A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *Eur. J. Oper. Res.* **141**(2), 274–294 (2002)
6. G. Belov, G. Scheithauer, A branch-and-cut-and-price algorithm for one- and two-dimensional two-staged cutting. *Eur. J. Oper. Res.* **171**(1), 85–106 (2006)
7. G. Belov, G. Scheithauer, Setup and open-stacks minimization in one-dimensional stock cutting. *INFORMS J. Comput.* **19**(1), 27–35 (2007)
8. J.A. Bennell, X. Song, A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Comput. Oper. Res.* **35**, 267–281 (2006)
9. M.A. Boschetti, A. Mingozzi, The two-dimensional finite bin packing problem. part I: new lower bounds for the oriented case. *4OR* **1**, 27–42 (2003)
10. G.B. Dantzig, P. Wolfe, Decomposition principle for linear programs. *Oper. Res.* **8**(1), 101–111 (1960)
11. J.M.V. de Carvalho, Exact solution of cutting stock problems using column generation and branch-and-bound. *Int. Trans. Oper. Res.* **5**, 35–44 (1998)
12. J.M.V. de Carvalho, LP models for bin packing and cutting stock problems. *Eur. J. Oper. Res.* **141**(2), 253–273 (2002)
13. H. Dyckhoff, A new linear approach to the cutting stock problem. *Oper. Res.* **29**(6), 1092–1104 (1981)
14. A.A. Farley, A note on bounding a class of linear programming problems, including cutting stock problems. *Oper. Res.* **38**(5), 922–923 (1990)
15. M.R. Garey, D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
16. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting-stock problem (Part I). *Oper. Res.* **9**, 849–859 (1961)
17. R.E. Johnston, E. Sadinlija, A new model for complete solutions to one-dimensional cutting stock problems. *Eur. J. Oper. Res.* **153**(1), 176–183 (2004)

18. L.V. Kantorovich, Mathematical methods of organising and planning production. *Manag. Sci.* **6**, 366–422 (1939)
19. S. Martello, P. Toth, *Knapsack Problems* (Wiley, Chichester, 1990)
20. J. Martinovic, G. Scheithauer, Integer linear programming models for the skiving stock problem. *Eur. J. Oper. Res.* **251**(2), 356–368 (2016)
21. J. Martinovic, G. Scheithauer, V. de Carvalho, A comparative study of the arcfow model and the one-cut model for one-dimensional cutting stock problems. Preprint MATH-NM-03-2017, Technische Universität Dresden (2017)
22. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988)
23. C. Nitsche, G. Scheithauer, J. Terno, Tighter relaxations for the cutting stock problem. *Eur. J. Oper. Res.* **112/3**, 654–663 (1999)
24. J. Nocedal, S.J. Wright, *Numerical Optimization* (Springer, New York, 1999)
25. J. Rietz, Untersuchungen zu MIRUP für Vektorpackprobleme. PhD thesis, Technische Universität Bergakademie Freiberg (2003)
26. J. Rietz, S. Dempe, Large gaps in one-dimensional cutting stock problems. *Discrete Appl. Math.* **156**(10), 1929–1935 (2008)
27. J. Rietz, G. Scheithauer, J. Terno, Tighter bounds for the gap and non-IRUP constructions in the one-dimensional cutting stock problem. *Optimization* **51**(6), 927–963 (2002)
28. G. Scheithauer, On the MAXGAP problem for cutting stock problems. *J. Inf. Process. Cybern.* **30**, 111–117 (1994)
29. G. Scheithauer, J. Terno, About the gap between the optimal values of the integer and continuous relaxation of the one-dimensional cutting stock problem, in *Operations Research Proceedings* (Springer, Berlin/Heidelberg, 1992), pp. 439–444
30. G. Scheithauer, J. Terno, The properties IRUP and MIRUP for the cutting stock problem, in *Decision Making under Conditions of Uncertainty (Cutting-Packing Problems)* (Ufa State Aviation Technical University, Ufa, 1997), pp. 16–31
31. A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, New York, 1986)
32. D. Simchi-Levi, New worst-case results for the bin-packing problem. *Naval. Res. Logist.* **41**, 579–585 (1994)
33. J. Terno, R. Lindemann, G. Scheithauer, *Zuschnittprobleme und ihre praktische Lösung* (Verlag Harri Deutsch, Thun und Frankfurt/Main, 1987)

# Chapter 5

## Orthogonal Packing Feasibility, Two-Dimensional Knapsack Problems

The *Orthogonal Packing Feasibility Problem* and the *Orthogonal Knapsack Problem* are basic problems in two- and higher-dimensional cutting and packing. For given dimensionality  $d \geq 2$ , we consider a set of  $d$ -dimensional rectangular items (pieces) that need to be packed into the given container. The input data describe the container sizes  $W_k \in \mathbb{Z}_+$  for  $k \in D := \{1, \dots, d\}$ , the item sizes  $w_i^k \in \{1, \dots, W_k\}$  for  $k \in D$ , and, in case of a knapsack problem, the profit (value) coefficient  $p_i$  of any item  $i \in I := \{1, \dots, m\}$ .

The *d-dimensional Orthogonal Packing Problem* (*dOPP*) [10, 11] is the feasibility problem: *decide whether all the  $m$  pieces can orthogonally be packed into the container without rotations*. The *d-dimensional Orthogonal Knapsack Problem* (*dOKP*) asks for a subset  $I^*$  of items of maximal total profit  $\sum_{i \in I^*} p_i$  which can orthogonally be packed into the container without rotations.

For short, we denote an instance of the *dOPP* and the *dOKP* by a tuple  $E = (m, W, w)$  or  $E = (m, W, w, p)$ , respectively, where  $m \in \mathbb{N}$ ,  $W \in \mathbb{N}^d$ ,  $w \in \mathbb{N}^{dm}$ , and  $p \in \mathbb{N}^m$ . Since the OPP and the OKP are strongly related, we examine both problems together in appropriate cases. If not stated otherwise, in case of an OPP instance, we assign  $p_i := \prod_{k \in D} w_i^k$ ,  $i \in I$ , to obtain a related OKP instance.

The OPP is a subproblem in some solution methods for orthogonal *bin packing problems* (BPP) and *knapsack problems* (OKP) [2, 11, 28]. It is polynomially equivalent to the orthogonal *strip packing problem* (SPP) [1, 18].

In the following, we describe a basic nonlinear and some integer linear programming models (for simplicity, frequently for  $d = 2$ ). Afterwards we discuss some necessary conditions for the feasibility of an OPP instance. We continue with a short description of the graph-theoretical approach of Fekete and Sheeppers [11]. Moreover, we also present some results concerning statements on the packability of a set of rectangular items into a container. Finally, we propose a (general) branch-and-bound algorithm based on the contour concept which allows to regard various additional restrictions and to construct fast heuristics.

## 5.1 Nonlinear Basic Models

Let  $(x_i^1, \dots, x_i^d)^\top$  denote the allocation point of piece  $i \in I$ . Thus, piece  $i$  covers the rectangular region

$$R_i(x_i) := \{x \in \mathbb{R}^d : x_i^k \leq x_i < x_i^k + w_i^k, k \in D\}.$$

Then, the nonlinear *natural model* [10, 28] just states the containment and non-overlapping conditions as follows:

### Basic model of the OPP

*Find a set of coordinates  $x_i^k$ ,  $k \in D, i \in I$ , satisfying*

$$0 \leq x_i^k \leq W^k - w_i^k, \quad k \in D, i \in I, \tag{5.1a}$$

$$x_i^k + w_i^k \leq x_j^k \quad \text{or} \quad x_j^k + w_j^k \leq x_i^k \quad \text{for at least one } k \in D, i < j, i, j \in I, \tag{5.1b}$$

*or prove that it does not exist.*

Constraints (5.1a) are *containment conditions* stating that each item should lie in the big box. Constraints (5.1b) are *non-overlapping conditions* enforcing that any two items  $i$  and  $j$  should be separated along at least one coordinate axis; in particular, we can specify *item orderings*: “item  $i$  lies completely before item  $j$  along axis  $k$ ”. This very simple model can immediately be passed to *constraint programming* software and, supplemented by bounds such as *conservative scales* or a *sweep line* technique, leads to efficient solution algorithms [10, 28] (see below).

Using the definition of the OPP, we can formulate the OKP as follows:

### Basic model of the OKP

*Find a subset  $I^*$  of  $I$  with maximal total profit  $\sum_{i \in I^*} p_i$  such that  $I^*$  defines a*

*feasible OPP instance.*

Obviously, if the optimal value of the OKP equals  $\sum_{i \in I} p_i$ , i.e., if all items are packed, then the OKP instance is a feasible OPP instance. For that reason, models and methods addressed to the OKP can straightforwardly be applied to the OPP.

## 5.2 Integer Linear Programming Models

Several ILP models for the OPP are known in literature, e.g. [2, 3, 6, 27]. Their exact solution is difficult, in general, because of the weak LP bounds of some models [27], the quadratic number of binary variables, and, in some models, the pseudo-polynomial number of position-indexed variables [2, 3]. We will formulate these approaches with respect to the OKP if appropriate.

### 5.2.1 The Beasley-Type Model

For the sake of simplicity in formulation, we consider here the two-dimensional case. We denote the size parameters of the container by  $W$  and  $H$ , and those of piece  $i$  by  $w_i$  and  $h_i$ ,  $i \in I$ . Obviously, we may assume that  $w_i \leq W$ ,  $h_i \leq H$ ,  $i \in I$ , and  $\sum_{i \in I} w_i \cdot h_i \leq WH$  hold since, otherwise, the OPP instance cannot possess a solution. However, in case of an OKP instance, the latter condition does not need to be satisfied, in general.

In the Beasley-type model, decision variables are used to indicate the *allocation point* (bottom-left corner)  $(x_i, y_i)$  of piece  $i \in I$ . In order to keep the number of binary variables small, we consider the sets of possible allocation coordinates

$$S(w, W) := \{r_0, \dots, r_\alpha\} = \{r : r = \sum_{i \in I} w_i a_i \leq W, a_i \in \{0, 1\}, i \in I\}$$

with  $0 = r_0 < r_1 < \dots < r_\alpha \leq W$  and

$$S(h, H) := \{s_0, \dots, s_\beta\} = \{s : s = \sum_{i \in I} h_i a_i \leq H, a_i \in \{0, 1\}, i \in I\}$$

with  $0 = s_0 < s_1 < \dots < s_\beta \leq H$  (cf. Sect. 2.7). Let  $J := \{0, \dots, \alpha - 1\}$  and  $K := \{0, \dots, \beta - 1\}$  denote corresponding index sets, respectively. The region covered by rectangle  $i$ , if it gets the *allocation point*  $(x_i, y_i)$ , is denoted by

$$R_i(x_i, y_i) := \{(x, y) \in \mathbb{R}^2 : x_i \leq x < x_i + w_i, y_i \leq y < y_i + h_i\}.$$

Based on the allocation points  $(x_i, y_i)$  for  $i \in \tilde{I} \subseteq I$ , we define the corresponding *arrangement or pattern*

$$A(\tilde{I}) = \{R_i(x_i, y_i) : i \in \tilde{I}\}.$$

According to the approach of Beasley [4], we define a binary variable  $x_{ipq}$  for all  $i \in I$ , for all  $p \in J_i := \{j \in J : r_j \leq W - w_i\}$ , and for all  $q \in K_i := \{k \in K : s_k \leq H - h_i\}$  as follows:

$$x_{ipq} := \begin{cases} 1, & \text{if item } i \text{ has allocation point } (x_i, y_i) = (r_p, s_q), \\ 0, & \text{otherwise.} \end{cases}$$

In difference to [4] where each point  $(p, q)$  with  $p \in \{0, \dots, W - 1\}$  and  $q \in \{0, \dots, H - 1\}$  could be a potential allocation point, we apply a possible reduction of the model size which is caused by considering only appropriate potential coordinates. We call a pattern whose allocation points belong to  $S(w, W) \times S(h, H)$  *normalized*. Obviously, patterns which are named *bottom-left justified* in the related literature, are normalized. A bottom-left justified pattern is obtained if all placed items cannot be shifted down- or leftwards. In the Beasley-type model the non-overlapping condition of placed pieces is formulated using the coefficients

$$a_{ipqjk} := \begin{cases} 1, & \text{if } x_{ipq} = 1, (r_j, s_k) \in R_i(r_p, s_q), \\ 0, & \text{otherwise,} \end{cases} \quad i \in I, p, j \in J, q, k \in K,$$

indicating that point  $(r_j, s_k)$  is covered if item  $i$  is allocated at  $(r_p, s_q)$ , or not. We obtain the following model of the OKP:

### Beasley-type model of the OKP

$$z^{Bea} = \max \sum_{i \in I} p_i \sum_{p \in J_i} \sum_{q \in K_i} x_{ipq} \quad \text{s.t.} \quad (5.2a)$$

$$\sum_{i \in I} \sum_{p \in J_i} \sum_{q \in K_i} a_{ipqjk} x_{ipq} \leq 1, \quad j \in J, k \in K, \quad (5.2b)$$

$$\sum_{p \in J_i} \sum_{q \in K_i} x_{ipq} \leq 1, \quad i \in I, \quad (5.2c)$$

$$x_{ipq} \in \{0, 1\}, \quad i \in I, p \in J_i, q \in K_i.$$

Restrictions (5.2b) ensure that at most one piece covers point  $(r_j, s_k)$ . Because of (5.2c) any piece is placed at most once. The set  $I^*$  of placed pieces can easily be identified by  $I^* = \{i \in I : \sum_{p \in J_i} \sum_{q \in K_i} x_{ipq} = 1\}$ .

According to the objective function (5.2a), the corresponding OPP is feasible if and only if  $I^* = I$ , or using  $p_i = 1$  for all  $i \in I$ , if and only if the optimal value is equal to  $m = |I|$ .

We notice that the non-overlapping condition (5.2b) can be formulated without the 5-index coefficients:

$$\sum_{i \in I} \sum_{p: r_j - w_i < r_p \leq r_j} \sum_{q: s_k - h_i < q \leq s_k} x_{ipq} \leq 1, \quad j \in J, k \in K.$$

Besides the application of sets of potential allocation points also the reduced sets can be used to further save some binary variables (see Sect. 2.7).

In order to avoid symmetric solutions (resulting by reflection on the vertical or horizontal symmetry axis) in the case of the OPP, we can add for a particular item  $i$  the condition

$$\sum_{p \in J_i : r_p \leq (W - w_i)/2} \sum_{q \in K_i : s_q \leq (H - h_i)/2} x_{ipq} = 1, \quad (5.3)$$

demanding that the largest part of item  $i$  lies in the bottom-left quarter of the container. Choosing a piece with largest area has been proven to be favorable.

### 5.2.2 The Padberg-Type Model

To obtain a model with polynomial number of binary variables (polynomial in  $m$ ), we define decision variables

$$\delta_i := \sum_{p \in J_i} \sum_{q \in K_i} x_{ipq}, \quad i \in I,$$

where the  $x_{ipq}$  are the decision variables of the Beasley-type model. Thus,  $\delta_i = 1$  holds if and only if piece  $i$  is allocated.

In the Padberg-type model the coordinates of the allocation point  $(x_i, y_i)$  of piece  $i$  are considered to be real variables. It is easy to see, their relation to the Beasley-type model is given by

$$x_i := \sum_{p \in J_i} r_p \sum_{q \in K_i} x_{ipq}, \quad y_i := \sum_{q \in K_i} s_q \sum_{p \in J_i} x_{ipq}.$$

Furthermore, we define binary variables  $u_{ij}$  and  $v_{ij}$  to model the non-overlapping of pieces  $i$  and  $j$  for  $i, j \in I$ ,  $i \neq j$  as follows:

- if  $u_{ij} = 1$  holds, then item  $j$  is placed right to item  $i$ , i.e.,  $x_j \geq x_i + w_i$  is satisfied,
- if  $v_{ij} = 1$  holds, then item  $j$  is placed above item  $i$ , i.e.,  $y_j \geq y_i + h_i$  is satisfied.

#### Padberg-type model of the OKP

$$z^{Pad} = \max \sum_{i \in I} p_i \delta_i \quad \text{s.t.} \quad (5.4a)$$

$$0 \leq x_i \leq (W - w_i)\delta_i, \quad 0 \leq y_i \leq (H - h_i)\delta_i, \quad i \in I, \quad (5.4b)$$

$$x_i + w_i \delta_i \leq x_j + W(\delta_i - u_{ij}), \quad i, j \in I, \quad i \neq j, \quad (5.4c)$$

$$y_i + h_i \delta_i \leq y_j + H(\delta_i - v_{ij}), \quad i, j \in I, \quad i \neq j, \quad (5.4d)$$

$$u_{ij} + u_{ji} + v_{ij} + v_{ji} \leq \delta_i, \quad i, j \in I, \quad i < j, \quad (5.4e)$$

$$\delta_i + \delta_j \leq 1 + u_{ij} + u_{ji} + v_{ij} + v_{ji}, \quad i, j \in I, \quad i < j, \quad (5.4f)$$

$$x_i, y_i \in \mathbb{R}, \quad \delta_i, \quad u_{ij}, \quad v_{ij} \in \{0, 1\}, \quad i, j \in I, \quad i \neq j. \quad (5.4g)$$

According to objective function (5.4a) the OPP is feasible if and only if the optimal value is  $m$  if  $p_i = 1$  is used for all  $i \in I$ . Restrictions (5.4b) ensure that piece  $i$  is completely placed within the container if  $\delta_i = 1$ . Otherwise, if  $\delta_i = 0$ , then  $x_i = y_i = 0$  is enforced. Conditions (5.4c) and (5.4d) model the interrelation of two placed pieces. If  $\delta_i = 1$  and  $\delta_j = 1$ , then, because of (5.4f), at least one of the variables  $u_{ij}$ ,  $u_{ji}$ ,  $v_{ij}$ , and  $v_{ji}$  gets the value 1. If, for example,  $u_{ij} = 1$ , then  $x_i + w_i \leq x_j$  becomes a relevant constraint due to (5.4c). Hence, the two rectangles do not overlap. The same holds for  $v_{ij} = 1$ . Because of (5.4e), if  $\delta_i = 0$ , then all  $u$ - and  $v$ -variables related to  $i$  are 0. Otherwise, exactly one of these variables gets value 1 [because of (5.4f)]. If  $\delta_i = 0$ , then  $u_{ij} = 0$  and  $x_i = 0$  reduce (5.4c) to  $x_j \geq 0$ .

Although model (5.4) has a polynomial number of binary variables, it is hard to solve because of a weak LP relaxation (see [27]). Therefore, additional restrictions to strengthen the LP relaxation are considered in [27]. To avoid symmetric solutions (related to reflections) in the case of the OPP, we can easily add the constraints

$$x_i \leq (W - w_i)/2, \quad y_i \leq (H - h_i)/2$$

for some item  $i$  similar to (5.3).

As already proposed in [30], instead of the four variables  $u_{ij}$ ,  $u_{ji}$ ,  $v_{ij}$ , and  $v_{ji}$ , only two binary variables  $\tilde{u}_{ij}$  and  $\tilde{v}_{ij}$  are sufficient to describe the four possible mutual positions of rectangles  $i$  and  $j$ . In detail, all restrictions (5.4c)–(5.4g) can be replaced by

$$\begin{aligned} x_i + w_i \delta_i &\leq x_j + W(2 - \tilde{u}_{ij} - \tilde{v}_{ij}), \\ x_j + w_j \delta_j &\leq x_i + W(1 - \tilde{u}_{ij} + \tilde{v}_{ij}), \\ y_i + h_i \delta_i &\leq y_j + H(1 + \tilde{u}_{ij} - \tilde{v}_{ij}), \\ y_j + h_j \delta_j &\leq y_i + H(\tilde{u}_{ij} + \tilde{v}_{ij}), \\ \delta_i \in \{0, 1\}, \quad i \in I, \quad \tilde{u}_{ij}, \tilde{v}_{ij} \in \{0, 1\}, \quad i, j \in I, \quad i < j. \end{aligned} \quad (5.5)$$

If both rectangles,  $i$  and  $j$ , are packed, i.e.  $\delta_i = \delta_j = 1$ , then exactly one of the four inequalities is non-trivial which ensures the non-overlapping of the two items. Moreover, in the case that, e.g., rectangle  $i$  is not placed, but  $j$  is, i.e.,  $\delta_j = 1$  and  $\delta_i = 0$  enforcing  $x_i = y_i = 0$  due to restriction (5.4b), then the inequalities  $x_j + w_j \leq x_i + W(1 - \tilde{u}_{ij} + \tilde{v}_{ij})$  and  $y_j + h_j \leq y_i + H(\tilde{u}_{ij} + \tilde{v}_{ij})$  become redundant for  $\tilde{v}_{ij} = 1$  so that the feasibility of the model is guaranteed.

The technique used in (5.5) possesses computational advantages in comparison to the Padberg-type model, in general, which result from the smaller number of binary variables and a stronger LP relaxation. Such a formulation is successfully applied

within *sequence-pair* approaches widely used for solving VLSI module placement and floor planning problems [24, 26], and for other packing problems [14].

## 5.3 Necessary Conditions for Feasibility

To facilitate the OPP (in)feasibility decision we can employ bounds. To prove that an instance is feasible it is sufficient to provide a packing pattern, whereas relaxations of the problem can be employed to show infeasibility. Since we model the OPP as a maximization problem, upper bounds for the optimal value could be meaningful. For instance, if an upper bound  $ub$  of  $z^{B\!ea}$  or  $z^{P\!ad}$  is less than  $m$  (if  $p_i = 1$  is used), then the OPP instance is infeasible. Otherwise, if  $ub = m$ , then we do not have any decision. Upper bounds are also useful in solution approaches like branch-and-bound.

### 5.3.1 Natural Criteria for Feasibility

As already mentioned, a 2OPP instance  $E = (m, W, H, w, h)$  is obviously infeasible if at least one of the following conditions is satisfied:

$$\max_{i \in I} w_i > W, \quad \max_{i \in I} h_i > H, \quad \sum_{i \in I} w_i \cdot h_i > WH.$$

It is obvious that the last condition relates to the natural lower bound

$$lb_A(E) := \left\lceil \frac{1}{W} \sum_{i \in I} w_i \cdot h_i \right\rceil.$$

Clearly, if  $lb_A(E) > H$  holds, then not all items of  $E$  can be packed into the rectangle  $W \times H$ . This kind of a lower bound is known as *area* or *material bound*, or as *volume bound* in higher dimensions.

Another condition, easy to verify, is as follows: if there exist two items  $i \neq j$ , with  $w_i + w_j > W$  and  $h_i + h_j > H$ , then infeasibility of  $E$  follows. This criterion can easily be generalized to more than two items.

In case of  $W > r_\alpha$  or  $H > s_\beta$ , where  $r_\alpha$  and  $s_\beta$  are defined in Sect. 5.2.1, we can apply some reduction of the container size leading to a possibly stronger decision criterion (see Sect. 3.3). Besides these reduction methods the size parameters of single items can be modified by, for instance, *dual feasible functions* (see Sect. 3.2.3) or *conservative scales* which are as well modified item sizes. Conservative scales can be obtained by dual-feasible functions or some LP approaches (see below).

### 5.3.2 The Bar Relaxation

The *bar relaxation* can formally be derived as follows: we consider again the Beasley-type model (5.2) and take  $J = \{0, \dots, W - 1\}$  and  $K = \{0, \dots, H - 1\}$ . That means, any integer lattice point is considered as a potential allocation point. The aggregation of restrictions (5.2b) leads to new (weaker) restrictions

$$\sum_{i \in I} \sum_{p \in J_i} \sum_{q \in K_i} \sum_{j \in J} a_{ipqjk} x_{ipq} \leq W, \quad k \in K. \quad (5.6)$$

Applying the substitution

$$a_{i,k+1} := \frac{1}{w_i} \sum_{p \in J_i} \sum_{q \in K_i} \sum_{j \in J} a_{ipqjk} x_{ipq}, \quad k \in K,$$

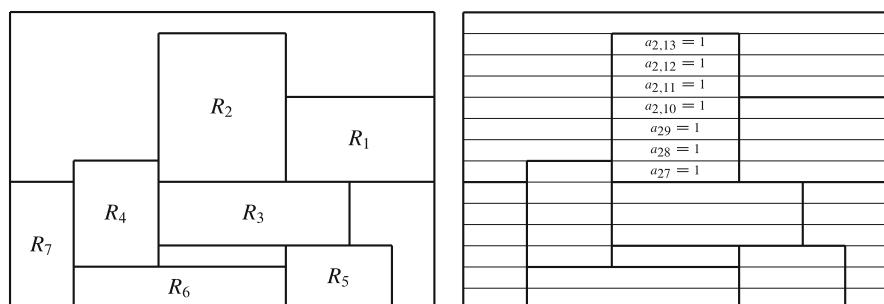
we obtain the compressed restrictions

$$\sum_{i \in I} w_i a_{i,k+1} \leq W, \quad k \in K,$$

from (5.6). It is obvious that if  $x_{ipq} = 1$  holds, then

$$a_{ik} = \begin{cases} 1, & \text{for } k = q + 1, \dots, q + h_i, \\ 0, & \text{otherwise} \end{cases}$$

results. To illustrate what happens, we consider Fig. 5.1. The container  $W \times H$  is dissected into  $H$  strips (*bars*)  $W \times 1$  which are numbered with  $1, \dots, H$ . To each piece  $i$  we assign  $H$  binary variables  $a_{ik}$ . At most  $h_i$  of them can get value 1, namely if and only if piece  $i$  is (partially) covered by strip (bar)  $\{(x, y) : 0 \leq x \leq W, k - 1 \leq y < k\}$ . In each (horizontal) bar  $k$  the total width of contained pieces is not larger than



**Fig. 5.1** Horizontal bar relaxation for rectangle packing

$W$ . Hence, vector  $a^k = (a_{1k}, \dots, a_{mk})^T$  represents a *one-dimensional* binary pattern in  $W$ -direction.

With other words, each piece  $i \in I$  is cut into  $h_i$  strips of size  $w_i \times 1$ , and  $H$  one-dimensional (horizontal, binary) patterns  $a^k$  are assigned to a two-dimensional arrangement of the pieces. This leads us to a relaxation of the two-dimensional packing problem with  $mH$  binary variables. We call it the (*horizontal*) *bar relaxation of Kantorovich-type* if the  $a_{ik}$  are considered to be variables. In the following, let  $K = \{1, \dots, H\}$ .

### Horizontal bar relaxation of Kantorovich-type

$$z^{Kan} = \max \sum_{i \in I} \frac{p_i}{h_i} \sum_{k \in K} a_{ik} \quad \text{s.t.} \quad (5.7a)$$

$$\sum_{i \in I} w_i a_{ik} \leq W, \quad k \in K, \quad (5.7b)$$

$$\sum_{k \in K} a_{ik} \leq h_i, \quad i \in I, \quad (5.7c)$$

$$a_{ik} \in \{0, 1\}, \quad i \in I, k \in K.$$

Restrictions (5.7b) ensure the feasibility of the one-dimensional strip (bar) patterns. Furthermore, condition (5.7c) guarantees that the two-dimensional piece  $i$  is packed at most once. Note that in this relaxation it is no longer guaranteed that the items of one piece type occur in consecutive strip patterns. Moreover, the position in  $W$ -direction of parts of item  $i$  can be different within the strips in which they are contained.

Because of the dissection of the two-dimensional piece  $i$  into  $h_i$  strips (bars) of width  $w_i$  and height 1, we have modified profit coefficients in the objective function (5.7a). Concerning the OPP, we take  $p_i := w_i h_i$  for all  $i \in I$ . Then, according to the target function (5.7a), the OPP instance is infeasible if  $z^{Kan} < \sum_{i \in I} w_i h_i$ . Let  $z_{LP}^{Kan}$  denote the optimal value of the LP relaxation of (5.7). As easily can be seen (Exercise 5.7), if  $p_i := w_i h_i$  for all  $i \in I$ , then we have

$$z_{LP}^{Kan} = \min\{WH, \sum_{i \in I} w_i h_i\}. \quad (5.8)$$

That means,  $z_{LP}^{Kan}$  does not lead to a decision of feasibility for an OPP instance. Since the LP relaxation of (5.7) is weak, hereinafter we consider the variables  $a_{ik}$  as known coefficients of a one-dimensional pattern  $a^k$ , and we denote an appropriate index set of all feasible strip patterns by  $\bar{K}$ , i.e., we have  $a^k \in \mathbb{B}^m$  and  $\sum_{i \in I} w_i a_{ik} \leq W$  for all  $k \in \bar{K}$ . Then, we define integer variables  $t_k$  to indicate the frequency pattern  $a^k$  is

used which leads to a CSP-based relaxation of the OKP

### Horizontal bar relaxation of the 2OKP

$$z^{hor} = \max \sum_{k \in \bar{K}} \sum_{i \in I} \frac{p_i}{h_i} a_{ik} t_k \quad \text{s.t.} \quad (5.9a)$$

$$\sum_{k \in \bar{K}} t_k \leq H, \quad (5.9b)$$

$$\sum_{k \in \bar{K}} a_{ik} t_k \leq h_i, \quad i \in I, \quad (5.9c)$$

$$t_k \in \mathbb{Z}_+, \quad k \in \bar{K}.$$

Because of restrictions (5.9c) not more than  $h_i$  items of piece type  $i$  are packed in not more than  $H$  patterns, condition (5.9b). According to (5.9a), assuming again  $p_i = w_i h_i$  for all  $i \in I$ , an OPP instance is infeasible if  $z^{hor} < \sum_{i \in I} w_i h_i$  according to (5.9a).

Clearly, both optimal values  $z^{Kan}$  and  $z^{hor}$  coincide, but their computation is expensive in general, in particular, due to the integrality conditions. Therefore, to obtain computable bounds, we relax the integer constraint by considering the LP relaxations of models (5.7) and (5.9).

In contrast to  $z_{LP}^{Kan}$ , the LP relaxation of the horizontal bar relaxation provides, in general, better bounds, justified by the MIRUP observation for the 1CSP (see Sect. 4.11). We illustrate this aspect by the following example.

*Example 5.1* Let  $W = H = 2n$  for some  $n \in \mathbb{N}$ ,  $n \geq 3$ , and let two rectangular pieces with  $w_i = h_i = n + 1$ ,  $i = 1, 2$ , be given. Obviously, the OPP instance is infeasible. As solution of the LP relaxation of model (5.7) we obtain  $z_{LP}^{Kan} = 2(n+1)^2$  determined by bar patterns  $a_{1k} = 1$ ,  $a_{2k} = (n-1)/(n+1)$ ,  $k = 1, \dots, n+1$ , and  $a_{1k} = 0$ ,  $a_{2k} = 1$ ,  $k = n+2, n+3$ ,

Since we have only two bar patterns,  $a^1 = (1, 0)^\top$  and  $a^2 = (0, 1)^\top$  in model (5.9), we obtain  $z_{LP}^{hor} = 2n(n+1) < 2(n+1)^2$  due to restriction (5.9b). This proves the infeasibility of the OPP instance. ■

As a consequence of our considerations, in the case of the OPP we can also use the optimal value  $z^{hbCSP}$  of a 1CSP with binary patterns, capacity  $W$ , item sizes  $w_i$  and demands  $h_i$ .

### Horizontal binary 1CSP relaxation of the 2OPP

$$z^{hbCSP} = \min \sum_{k \in \bar{K}} t_k \quad \text{s.t.} \quad (5.10a)$$

$$\sum_{k \in \bar{K}} a_{ik} t_k \geq h_i, \quad i \in I, \quad (5.10b)$$

$$t_k \in \mathbb{Z}_+, \quad k \in \bar{K}.$$

If  $z^{hbCSP} > H$  holds, then the OPP instance  $E = (m, W, H, w, h)$  is infeasible.

Concerning again the OKP, besides the relaxation with horizontal bar patterns, we can define vertical bar patterns. Let  $\bar{J}$  denote an appropriate index set of all feasible vertical bar patterns  $A^j = (A_{1j}, \dots, A_{mj})^T$ , i.e., we have  $A^j \in \mathbb{B}^m$  and  $\sum_{i \in I} h_i A_{ij} \leq H$  for all  $j \in \bar{J}$ . Using integer variables  $\tau_j$  indicating the frequency of bar pattern  $A^j$  within a solution, we obtain a model of the

### Vertical bar relaxation of the 2OKP

$$z^{ver} = \max \sum_{j \in \bar{J}} \sum_{i \in I} \frac{p_i}{w_i} A_{ij} \tau_j \quad \text{s.t.} \quad (5.11a)$$

$$\sum_{j \in \bar{J}} \tau_j \leq W, \quad (5.11b)$$

$$\sum_{j \in \bar{J}} A_{ij} \tau_j \leq w_i, \quad i \in I, \quad (5.11c)$$

$$\tau_j \in \mathbb{Z}_+, \quad j \in \bar{J}.$$

The meaning of restrictions (5.11b)–(5.11c) is similar to that of conditions (5.9b)–(5.9c), but now in different direction. Similar to the horizontal binary 1CSP formulation, we define the

### Vertical binary 1CSP relaxation of the 2OPP

$$z^{vbCSP} = \min \sum_{j \in \bar{J}} \tau_j \quad \text{s.t.} \quad (5.12a)$$

$$\sum_{j \in \bar{J}} A_{ij} \tau_j \geq w_i, \quad i \in I, \quad (5.12b)$$

$$\tau_j \in \mathbb{Z}_+, \quad j \in \bar{J}.$$

It is remarkable, the optimal values  $z^{hor}$  and  $z^{ver}$  are different, in general (Exercise 5.8). The same holds for the LP bounds  $z_{LP}^{hor}$  and  $z_{LP}^{ver}$  and also for the values obtained by the CSP relaxations.

The combination of the horizontal and the vertical bar relaxation leads to an even stronger relaxation of the OKP:

### Combined bar relaxation of the 2OKP

$$z^{cbr} = \max \sum_{k \in \bar{K}} \sum_{i \in I} \frac{p_i}{h_i} a_{ik} t_k \quad \text{s.t.} \quad (5.13a)$$

$$\sum_{k \in \bar{K}} t_k \leq H, \quad (5.13b)$$

$$\sum_{k \in \bar{K}} a_{ik} t_k \leq h_i, \quad i \in I, \quad (5.13c)$$

$$\sum_{j \in \bar{J}} \tau_j \leq W, \quad (5.13d)$$

$$\sum_{j \in \bar{J}} A_{ij} \tau_j \leq w_i, \quad i \in I, \quad (5.13e)$$

$$\sum_{k \in \bar{K}} w_i a_{ik} t_k = \sum_{j \in \bar{J}} h_i A_{ij} \tau_j \quad i \in I, \quad (5.13f)$$

$$t_k \in \mathbb{Z}_+, \quad k \in \bar{K}, \quad \tau_j \in \mathbb{Z}_+, \quad j \in \bar{J}.$$

In difference to an independent consideration of the previous bar relaxations, the additional constraints (5.13f) ensure that in both, horizontal and vertical, directions the same (part of) area of each piece is used.

*Example 5.2* We consider a rectangular container of size  $10 \times 8$ . Which maximal area can be covered if 5 pieces  $4 \times 4$  and 6 pieces  $5 \times 3$  are available?

Concerning the horizontal bar relaxation, the input data are  $W = 10$ ,  $w = (4, 5)^\top$ , and supply  $u = (20, 18)^\top$  for a ‘compact’ formulation. We obtain  $z^{hor} = 80 = w_2 a_{12} t_1 = 5 \cdot 2 \cdot 8$  using pattern  $a^1 = (0, 2)^\top$ .

In case of the vertical bar relaxation, the input data are  $H = 8$ ,  $h = (4, 3)^\top$ , and supply  $u = (20, 30)^\top$  for a ‘compact’ formulation. We obtain  $z^{ver} = 80 = h_1 A_{11} \tau_1 = 4 \cdot 2 \cdot 10$  using pattern  $A^1 = (2, 0)^\top$ .

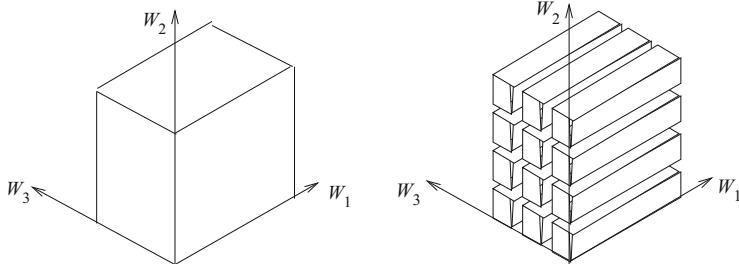
However, the combined bar relaxation possesses the optimal value  $z^{cbr} = 70$ :

$t_1 = 3$ ,  $a^1 = (0, 2)^\top$ ,  $t_2 = 5$ ,  $a^2 = (2, 0)^\top$ ,  $\tau_1 = 5$ ,  $A^1 = (2, 0)^\top$ ,  $\tau_2 = 5$ ,  $A^2 = (0, 2)^\top$ . ■

Note that the one-dimensional *bar relaxation* is also considered for  $d$ -dimensional packing problems with  $d \geq 3$  [6, 7, 29]. The three-dimensional case is addressed in more detail in relation to the *Container Loading Problem*, Chap. 11. For a  $d$ -dimensional OPP instance with container  $W_1 \times \dots \times W_d$  and pieces  $w_i^1 \times \dots \times w_i^d$ ,  $i \in I$ , and a certain dimension  $k \in \{1, \dots, d\}$ , the  $k$ th *bar relaxation* asks: is it possible to pack all  $\prod_{j=1, j \neq k}^d w_i^j$  one-dimensional items of length  $w_i^k$  for all  $i \in I$  into  $\prod_{j=1, j \neq k}^d W_j$  one-dimensional containers of length  $W_k$ ? A schema of the bar relaxation for  $d = 3$  and  $k = 1$  is shown in Fig. 5.2.

### 5.3.3 The Contiguous Relaxation

As already mentioned, within the above relaxations, it is not guaranteed that the one-dimensional items of the same type (resulting from a certain rectangular piece) occur in consecutive bar patterns (or that there exists a permutation of the bar patterns



**Fig. 5.2** A 3D container and the 1<sup>st</sup> bar relaxation (along  $W_1$ )

leading to the desired property). For that reason, we add an additional demand, the so-called

#### (horizontal) contiguous condition

There exists a sequence (ordering) of the (horizontal) bar patterns such that each (one-dimensional) item  $i$  occurs in consecutive bar patterns until its demand  $h_i$  is reached.

With this additional condition we obtain the *horizontal contiguous bar relaxation* for the 2OKP and the 2OPP. It was firstly used for the 2OPP by Monaci [25] within a branch-and-bound algorithm to get stronger bounds. Notice that only a verbal formulation of the contiguous condition was given in [25] where it was considered for the first time.

Indeed, the horizontal contiguous relaxation can provide a stronger bound than the horizontal bar relaxation.

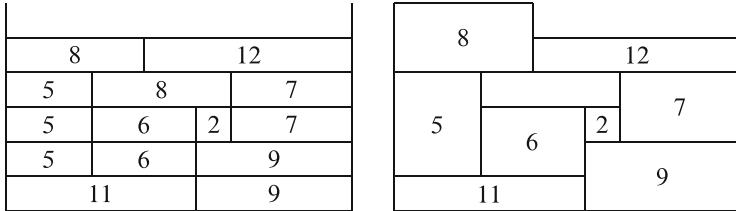
*Example 5.3* We consider the following 2OPP instance: is it possible to pack three pieces of size  $1 \times 2$  into a rectangle of widths  $W = 2$  and height  $H = 3$ ? Obviously, it is not possible. Using bar patterns  $(1, 1, 0)^\top$ ,  $(1, 0, 1)^\top$ , and  $(0, 1, 1)^\top$ , each with frequency one, we obtain  $z^{hor} = 6 \leq WH$  so that infeasibility is not discovered. In contrast, any solution of the horizontal contiguous relaxation packs at most two items proving the infeasibility of the OPP instance. ■

Before we discuss an ILP model of the stronger relaxation (in comparison to the horizontal bar relaxation), we show by means of an example that the horizontal contiguous relaxation does not provide an exact model of the OPP.

*Example 5.4* Is it possible to pack all the items  $2 \times 2$ ,  $11 \times 2$ ,  $12 \times 2$ ,  $6 \times 4$ ,  $7 \times 4$ ,  $8 \times 4$ ,  $9 \times 4$ , and  $5 \times 6$  into a containing rectangle of width  $W = 20$  and height  $H = 10$ ? Figure 5.3 shows a solution of the contiguous relaxation. But the OPP instance is infeasible. An optimal two-dimensional packing needs height 12. ■

To formulate an ILP model of the horizontal contiguous bar relaxation of the 2OKP, we define binary variables  $t_{ik}$  for each possible position  $k \in K_i := \{1, \dots, H - h_i + 1\}$  and for all pieces  $i \in I$  according to

$$t_{ik} = 1 \Leftrightarrow \text{the first item of type } i \text{ lies in bar } \{(x, y) : 0 \leq x \leq W, k - 1 \leq y < k\}.$$



**Fig. 5.3** The horizontal contiguous bar relaxation does not provide an OPP solution in general

That means, the  $y$ -coordinate of the allocation point of item  $i$  is  $k - 1$  if and only if  $t_{ik} = 1$ . However, the  $x$ -coordinate is not specified. For simplicity of description, we do not regard here a reduction of the number of variables with respect to the set of potential allocation coordinates (cf. Sect. 2.7). Let  $K(i, k) := \{\max\{1, k - h_i + 1\}, \dots, k\}$  denote the index set of horizontal strips in which the first item of type  $i$  can lie if an item of type  $i$  is in bar  $k$ .

### Horizontal contiguous relaxation

$$z^{hcr} := \max \sum_{i \in I} \sum_{k \in K_i} p_i t_{ik} \quad \text{s.t.} \quad (5.14a)$$

$$\sum_{k \in K_i} t_{ik} \leq 1, \quad i \in I, \quad (5.14b)$$

$$\sum_{i \in I} w_i \sum_{s \in K(i, k)} t_{is} \leq W, \quad k = 1, \dots, H, \quad (5.14c)$$

$$t_{ik} \in \{0, 1\}, \quad i \in I, \quad k \in K_i.$$

To avoid symmetric solutions (reflection on a horizontal symmetry axis), we can add the restriction  $\sum_{k=\lfloor(H-h_i)/2\rfloor+1}^{H-h_i+1} t_{ik} = 0$  for some (large) piece  $i$ . Note that, in general, it is not known in advance whether that item is contained in a solution.

In a straightforward manner, we also define the *vertical contiguous relaxation* of the 2OKP. To this end, we introduce binary variables  $\tau_{ij}$  for each possible position  $j \in J_i := \{1, \dots, W - w_i + 1\}$  and for all pieces  $i \in I$  according to

$$\tau_{ij} = 1 \Leftrightarrow \text{the first item of type } i \text{ lies in bar } \{(x, y) : j - 1 \leq x < j, 0 \leq y \leq H\}.$$

That means, the  $x$ -coordinate of the allocation point of item  $i$  is  $j - 1$  if and only if  $\tau_{ij} = 1$ . However, the  $y$ -coordinate is not specified in this case. For short, let

$J(i,j) := \{\max\{1, j - w_i + 1\}, \dots, j\}$  denote an appropriate index set similar to  $K(i,k)$ .

### Vertical contiguous relaxation

$$z^{vcr} := \max \sum_{i \in I} \sum_{j \in J_i} p_i \tau_{ij} \quad \text{s.t.} \quad (5.15a)$$

$$\sum_{j \in J_i} \tau_{ij} \leq 1, \quad i \in I, \quad (5.15b)$$

$$\sum_{i \in I} h_i \sum_{s \in J(i,j)} \tau_{is} \leq H, \quad j = 1, \dots, W, \quad (5.15c)$$

$$\tau_{ij} \in \{0, 1\}, \quad i \in I, j \in J_i.$$

Similar to above, to avoid symmetric solutions (reflection on a vertical symmetry axis), we can add the restriction  $\sum_{j=\lfloor (W-w_i)/2 \rfloor + 1}^{W-w_i+1} \tau_{ij} = 0$  for some (large) piece  $i$ .

The combination of horizontal and vertical contiguous relaxation allows to formulate a further model of the 2OKP and the 2OPP with a pseudo-polynomial number of binary variables:

### Contiguous relaxations based model of the 2OKP

$$z^{crm} := \max \sum_{i \in I} \sum_{k \in K_i} p_i t_{ik} \quad \text{s.t.} \quad (5.16a)$$

$$\sum_{k \in K_i} t_{ik} \leq 1, \quad i \in I, \quad (5.16b)$$

$$\sum_{i \in I} w_i \sum_{s \in K(i,k)} t_{is} \leq W, \quad k = 1, \dots, H, \quad (5.16c)$$

$$\sum_{j \in J_i} \tau_{ij} \leq 1, \quad i \in I, \quad (5.16d)$$

$$\sum_{i \in I} h_i \sum_{s \in J(i,j)} \tau_{is} \leq H, \quad j = 1, \dots, W, \quad (5.16e)$$

$$\sum_{k \in K_i} t_{ik} = \sum_{j \in J_i} \tau_{ij}, \quad i \in I, \quad (5.16f)$$

$$\sum_{s \in K(i,k)} t_{is} + \sum_{s \in K(q,k)} t_{qs} + \sum_{s \in J(i,j)} \tau_{is} + \sum_{s \in J(q,j)} \tau_{qs} \leq 3, \quad \begin{array}{l} i, q \in I, i < q, \\ k \in K, j \in J, \end{array} \quad (5.16g)$$

$$t_{ik} \in \{0, 1\}, \quad k \in K_i, \quad \tau_{ij} \in \{0, 1\}, \quad j \in J_i, \quad i \in I.$$

Restrictions (5.16f) combine the two directions and ensure that the same pieces are placed within the relaxations. Removing constraints (5.16g) we obtain a relaxation for the 2OKP which dominates the horizontal and the vertical contiguous relaxations. Note that the validity of this model to be a true model of the 2OKP can be shown by using Theorem 5.1 (see below).

Note that the number of variables can be decreased if the principle of (reduced) sets of potential allocation points is applied (cf. Sect. 2.7).

## 5.4 The Interval-Graph Approach

The interval-graph approach proposed by Fekete and Schepers [11] is designed for the  $d$ -dimensional OPP,  $d \geq 2$ . Therefore, we change back to the notations introduced in the beginning of this chapter. We follow here the description of the basic principles given in [5].

### 5.4.1 The Interval-Graph Model and a Simplification

The *interval-graph model*, proposed in [11], operates directly with *overlapping* and *separateness relations* of item projections on each axis. From that, respective item orderings and coordinates can be deduced. The overlapping relations, expressed as edges of a (non-oriented) graph, produce a so-called *interval graph* [11, 15]. The complement of such a graph possesses transitive orientations which correspond to orderings of the items along the axis. This model seems to be the most general one. In particular, it removes some of the symmetries in guillotine structured layouts. Therefore, an exact algorithm based on the interval-graph model has been proved to be an efficient algorithm for the OPP [10, 12].

The following theorem, presented in [11], defines and establishes the validity of the interval-graph model.

**Theorem 5.1 (Fekete and Schepers)** *An OPP instance  $(m, W, w) \in \mathbb{Z} \times \mathbb{Z}^d \times \mathbb{Z}^{dm}$  has a feasible solution if and only if there exists a system of graphs  $G_k = (I, E_k)$ ,  $k \in D = \{1, \dots, d\}$ , satisfying the properties*

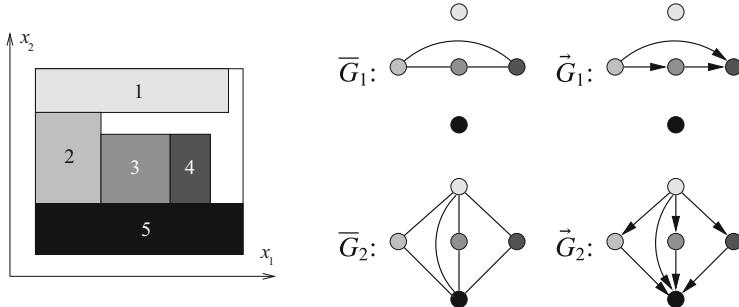
**P1** (Intervalness): *Each  $G_k$  is an interval graph, i.e., its edges correspond to overlapping relations of a set of intervals on a line.*

**P2** (Containment): *Each stable set (anti-clique)  $S$  of  $G_k$  is  $x_k$ -feasible, i.e.,*

$$\sum_{i \in S} w_i^k \leq W_k.$$

**P3** (Non-overlapping):  $\bigcap_{k=1}^d E_k = \emptyset$ .

*Proof Necessity.* We consider the following system of graphs: edge  $e = (i, j)$  belongs to  $G_k$  if and only if the projections of  $i$  and  $j$  on axis  $k$  strictly overlap, The



**Fig. 5.4** An exemplary packing pattern, the complement intersection graphs and their transitive orientations

so-called *intersection graphs* represent overlapping relations of item projections on the coordinate axes, see Fig. 5.4 illustrating the complements  $\bar{G}_1$ ,  $\bar{G}_2$  of  $G_1$ ,  $G_2$  for an exemplary packing, respectively.

The graphs  $G_k$ ,  $k \in D$ , are the *intersection graphs* of item projections onto the axes, which corresponds to the definition of intervalness [15], implying P1. Properties P2 and P3 can be shown straightforwardly.

*Sufficiency.* P1 ensures transitive orientability of the complements  $\bar{G}_k$  of the graphs  $G_k$ ,  $k \in D$ . Each transitive orientation  $\vec{G}_k$  of  $\bar{G}_k$  (transitivity means:  $(i, j) \in \vec{G}_k \wedge (j, q) \in \vec{G}_k \Rightarrow (i, q) \in \vec{G}_k$ ) corresponds to a partial ordering of items along axis  $k$ . Longest-path trees in the oriented graphs provide us with item coordinates. Due to P2, the containment condition (5.1a) holds. The last property P3 guarantees non-overlapping. ■

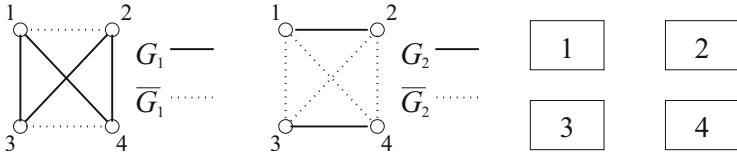
Property P1 arises naturally from feasible packings as a necessary condition. Let us consider the opposite direction, namely the question: *when does a system of graphs  $G_k = (I, E_k)$ ,  $k \in D$  represent a feasible packing?* Looking at the sufficiency proof, we see that we need, in addition to P2 and P3, only the transitive orientability of the complements  $\bar{G}_k$ . Transitively orientable graphs are called *comparability graphs* [15]. In fact, a given graph  $G$  is an interval graph if and only if its complement is a comparability graph and  $G$  does not contain any induced chordless cycle  $C_4$  [15]. Thus, the graphs  $G_k$  do not have to be interval, which allows to extend the set of candidate graphs:

**Theorem 5.2** *An OPP instance  $(m, W, w) \in \mathbb{Z} \times \mathbb{Z}^d \times \mathbb{Z}^{dm}$  has a feasible solution if and only if there exists a system of graphs  $G_k = (I, E_k)$ ,  $k \in D$ , satisfying the properties*

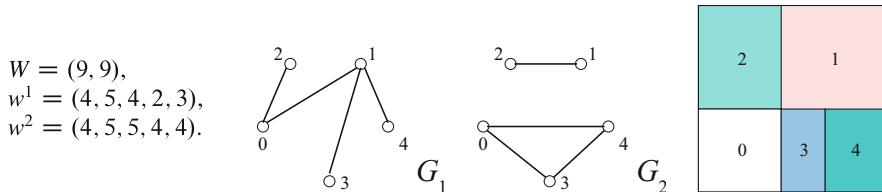
**P1'** (Orientability): *Each graph  $G_k$  is a complement of a comparability graph.*

**P2** (Containment): *Each stable set  $S$  of  $G_k$  is  $x_k$ -feasible, i.e.,  $\sum_{i \in S} w_i^k \leq W_k$ .*

**P3** (Non-overlapping):  $\bigcap_{k=1}^d E_k = \emptyset$ .



**Fig. 5.5** Example of a feasible graph system where  $G_1$  is non-interval



**Fig. 5.6** A ‘degenerate’ 2OPP example from [13]

An example is given in Fig. 5.5. In this example,  $G_1$  is non-interval (just a chordless  $C_4$ ), but  $\overline{G}_1$  has transitive orientations. We can select the orientations of  $\overline{G}_1$  and  $\overline{G}_2$  corresponding to the packing on the right.

A consequence of Theorem 5.2 is that we can have more graphs representing a solution, which might simplify the search.

The proofs of Theorems 5.1 and 5.2 show an interesting property of the graph model: given a feasible system of graphs  $G_k = (I, E_k)$ ,  $k \in D$ , and restoring a corresponding packing layout, the fact  $e = (i, j) \in E_k$  does not necessarily mean that items  $i$  and  $j$  strictly overlap in projection on axis  $k$ . An illustrating example was given in [13], where such cases are called *degenerate*, Fig. 5.6.

Note that in the problem setting of [12] it is impossible to demand that edges of  $G_k$  imply any minimal overlapping because item sizes are assumed to be real values.

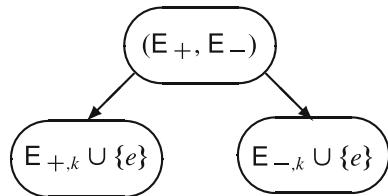
#### 5.4.2 The Algorithm of Fekete and Schepers

The exact algorithm of [12] has the goal to either obtain a system of graphs satisfying P1–P3 or to prove that none exists. The following issues are of importance for the design of the algorithm.

- **Branching Rule**

The algorithm enumerates the graphs edge by edge, fixing some edges to be *included* or *excluded*. Each fixing is called *augmentation* and is denoted by a triple

**Fig. 5.7** The branching schema: enumerating by edges



$(e, \delta, k)$  where

- $e = (i, j)$  is the edge,
- $\delta \in \{+, -\}$  means that the edge is included (+) or excluded (-) in  $G_k$ ,
- $k \in \{1, \dots, d\}$  is the dimension.

Each node of the search tree contains the fixed edges in the *search information*  $E = (E_+, E_-) = (E_{+,k}, E_{-,k})_{k=1}^d$ , see Fig. 5.7.

#### • Pruning Criteria and Bounds

To prune the current node because of infeasibility, we need a proof that its search information  $E$  cannot be augmented to feasible graphs. This is clearly the case when  $E$  contains fixed configurations which violate at least one of the properties P1–P3. Another possibility, employed in [12], is to consider the restrictions on item combinations dictated by  $E$ , which may allow an enlargement of item sizes and a stronger volume bound. This idea is generalized in [5].

#### • Solution Candidates and Feasibility Test

To obtain a feasible system of graphs, the authors take the graphs consisting of all included edges ( $G_k = (I, E_{+,k})$ ,  $k \in D$ ) and test them for feasibility. This is done in every node. The test is performed as follows (for each  $k \in D$ ):

1. Test P1': Check comparability of the complement  $\overline{G}_k$ .
2. Test P2: To check the feasibility of the largest stable set, solve the *maximal weighted clique problem* in the complement. This problem, generally NP-hard, is polynomially solvable in comparability graphs.
3. Test P1: Look for induced chordless  $C_4$  graphs.

Note that P3 is not tested because it is always satisfied by  $E_+$  due to preprocessing.

#### • Branching Edges

The edges to branch on are selected with the goal to eliminate infeasibilities in one of the subnodes. If the feasibility test discovers a forbidden subgraph  $\widetilde{G}$  in  $G_k$ , let  $A$  be the set of all free edges in this subgraph. If  $A$  is empty, the node can be pruned; if  $|A| = 1$ , the corresponding edge can be fixed (see preprocessing); otherwise, one of the free edges is selected for branching.

- **Preprocessing**

Preprocessing is modification of the problem with the goal to simplify the solution process [22], e.g., by tightening some of the constraints. Sometimes it is possible to modify item sizes in order to strengthen the volume bound. For two-dimensional packing problems, examples of preprocessing procedures are given, e.g., in [2] (see also Sect. 4.7). In the context of constraint programming, preprocessing is called *constraint propagation*. In the interval-graph algorithm, preprocessing is performed by extending the search information  $\mathbf{E}$  and is called *feasible augmentations*.

In the root node, all infeasible stable sets of two items are fixed at  $\mathbf{E}_+$ . In subnodes, feasible augmentations are performed by the principle outlined in the feasibility test: if just one edge is missing to complete a forbidden subgraph, it is fixed in the opposite way. This is done for the constraints P2, P3, and induced  $C_4$  graphs. Moreover, some fixings can be done on *indistinguishable edges* (i.e., edges whose corresponding end nodes are equal-sized items with equal search information).

It is to notice that many algorithms for the 2OPP are based on the observation that a feasible packing represents a feasible *non-preemptive cumulative-resource schedule* [6, 10, 31] or, equivalently, *contiguous one-dimensional cutting* [1, 21] along each axis. These algorithms follow the so-called *two-stage strategy*: first they enumerate the coordinates in one dimension, and then, for each feasible non-preemptive schedule, they enumerate the coordinates in the second dimension (see Sect. 5.7). It can be shown that the original interval-graph algorithm has a similar property:

**Theorem 5.3** *Suppose that in some node of the search tree, all graphs up to dimension  $k_0 \in \{1, \dots, d\}$  are feasible, i.e., graphs  $G_k = (I, \mathbf{E}_{+,k})$  for  $k \in D$  satisfy P1 and P2. Then all  $G_k$ ,  $k \in D$  will remain the same in the subtree of that node.*

*Proof* The graphs  $G_k$ ,  $k \in D$ , will always pass the feasibility test. Thus, no branching will be performed in dimensions  $k \leq k_0$ . Moreover, the preprocessing routines will not change the search information  $\mathbf{E}_{+,k}$ ,  $k \in D$ , because it contains no forbidden subgraphs; the preprocessing for P3 can only fix some ‘–’ edges. ■

Properties P1 and P2 for a certain  $k \in D$  do not imply that the non-preemptive schedule (obtained from a transitive orientation of  $G_k$ ) should satisfy the cumulative-resource constraint. For example, if all items are placed at the left margin of the container (i.e., starting at time zero), then the height of the corresponding packing can be too large. This situation can be checked in polynomial time as the *maximal stable set problem* in comparability graphs [15]; however this possibility was not mentioned in the original algorithm [12].

## 5.5 Sufficient Conditions for Feasibility

Besides necessary conditions for the feasibility of OPP instances, we can also look from the opposite side: which sufficient condition obviates the solution of the OPP? With other words, which circumstances guarantee a packing of all given pieces within the container? Some results are known; mostly for the two-dimensional case.

### 5.5.1 Rectangular Pieces

Let an instance  $E = (m, W, H, w, h)$  of the 2OPP be given where the container has size  $W \times H$  and the piece dimensions are  $w_i \times h_i$ ,  $i \in I = \{1, \dots, m\}$ . For short, let

$$A(I) := \sum_{i \in I} w_i h_i$$

denote the total area of all pieces. Furthermore, let

$$w_{\max} := \max\{w_i : i \in I\}, \quad h_{\max} := \max\{h_i : i \in I\},$$

and let  $(x)_+ := \max\{0, x\}$  for any real number  $x$ .

Some known results suggest that without any or only under weak assumptions the feasibility of a 2OPP instance  $E$ , i.e., the packability of all rectangular items into a single containing rectangle, can be guaranteed if  $A(I)$  is not too large, but near to 50 percent.

**Theorem 5.4 (Meir and Moser [23])** *Any set of rectangular items  $w_i \times h_i$ ,  $i \in I$ , satisfying  $w_i \geq h_i$  for all  $i \in I$  and total area  $A(I)$  can be packed into any rectangle of size  $W \times H$  if  $W \geq w_{\max}$  and*

$$WH \geq 2A(I) + W^2/8.$$

*This result is best possible.*

As a consequence of Theorem 5.4, we obtain the following corollary used, for instance, by Li and Cheng [20].

**Corollary 5.5 (Meir and Moser [23])** *A set of rectangles with  $w_i \geq h_i$  for all  $i \in I$  can be packed into a unit square if  $w_{\max} \leq 1$  and  $A(I) \leq 7/16$ .*

Another, stronger result on packability of rectangles was given by Steinberg in 1997:

**Theorem 5.6 (Steinberg [32])** *If for a set of rectangles  $w_i \times h_i$ ,  $i \in I$ , and a containing rectangle of size  $W \times H$  the following inequalities hold,*

$$w_{\max} \leq W, \quad h_{\max} \leq H, \quad 2A(I) \leq WH - (2w_{\max} - W)_+ \cdot (2h_{\max} - H)_+,$$

*then it is possible to pack all items into the container.*

The proof of this theorem is done by providing a polynomial time algorithm with worst-case performance ratio 2. This ratio is best-possible which can be seen by means of the following example: let a container of size  $W \times H$  and three rectangles of sizes  $\frac{1}{2}W \times H$ ,  $\varepsilon \times H$ , and  $\frac{1}{2}W \times \varepsilon$  be given with  $\varepsilon > 0$ . Then the assertion follows for  $\varepsilon \rightarrow 0$ .

Observe, if  $w_{max} \leq W/2$  or  $h_{max} \leq H/2$ , then all items can be packed if their total area is at most 50 percent of the container area. Consequently, if  $A(I) \leq WH$ , then all items fit in a rectangular container of size  $2W \times H$  or  $W \times 2H$ , but we cannot conclude that all items can be packed into two rectangles of size  $W \times H$ , since a guillotine cut bisecting the containing rectangle is not guaranteed. For instance, three items of size  $(\frac{1}{2}W + \varepsilon) \times (\frac{1}{2}H + \varepsilon)$  cannot be packed into two bins if  $\varepsilon > 0$ . However, the algorithm of Steinberg provides a guillotine pattern, that means, all pieces can be obtained using only guillotine cuts.

Jansen and Zhang improved the result of Steinberg for the special case that there are no wide items in  $E$ , i.e.,  $\{i \in I : w_i > W/2, h_i \leq H/2\} = \emptyset$ .

**Theorem 5.7 (Jansen and Zhang [17])** *If  $A(I) \leq \frac{1}{2}WH$ , and if the set  $I$  does not contain wide items, then all items of  $E$  can be packed, i.e.,  $E$  is a feasible OPP instance.*

Note that the theorem also holds if  $E$  contains a large item, i.e., one item  $i$  with  $w_i > W/2$  and  $h_i > H/2$ .

### 5.5.2 Quadratic Pieces

Meir and Moser published in 1968 also a result on the packability of  $d$ -dimensional cubes within a rectangular parallelepiped.

**Theorem 5.8 (Meir and Moser [23])** *Any set of  $d$ -dimensional cubes of sides  $w_1 \geq w_2 \geq \dots \geq w_m$  with total volume  $V = \sum_{i \in I} w_i^d$  can be packed into any  $d$ -dimensional rectangular parallelepiped of size  $W_1 \times W_2 \times \dots \times W_d$  if  $W_k > w_1$  for  $k = 1, \dots, d$  and*

$$w_1^d + (W_1 - w_1)(W_2 - w_1) \dots (W_d - w_1) \geq V.$$

*In certain cases this result is best possible.*

For the two-dimensional case, the following results are known:

**Theorem 5.9 (Meir and Moser [23])** *Any set of squares of sides  $w_1 \geq w_2 \geq \dots \geq w_m$  with total area  $A$  can be packed into any rectangle of size  $W \times H$  if  $\min\{W, H\} \geq w_1$  and  $WH \geq 2A$ .*

**Theorem 5.10 (Kleitman and Krieger [19])** *Any set of squares with total area  $A$  can be packed into a rectangle of size  $\sqrt{2A} \times \sqrt{4A}/3$ .*

As a consequence of the theorem, we have that any set of squares with total area 1 can be packed into a rectangle of area at most  $\sqrt{8/3} < 1.633$ . This result was improved by Hougaard:

**Theorem 5.11 (Hougaard [16])** *Any set of squares with total area 1 can be packed into a rectangle of area  $\frac{2867}{2048} < 1.4$ .*

It is remarkable that the proof of this theorem is computer-based. In [8] the following result is stated:

**Theorem 5.12** *Any set  $I$  of squares with sides  $w_i \leq \frac{1}{2}$  for all  $i \in I$  and total area  $A \leq 5/9$  can be packed into a unit square.*

This result is best-possible since 5 squares  $(\frac{1}{3} + \varepsilon) \times (\frac{1}{3} + \varepsilon)$  cannot be packed in the unit square for any  $\varepsilon > 0$ .

A generalization of Theorem 5.12 of the form

*Let  $p \in \mathbb{N}$ . Any set  $I$  of squares with sides  $w_i \leq \frac{1}{p}$  for all  $i \in I$  and total area*

*$A(I) \leq \frac{p^2 + 1}{(p + 1)^2}$  can be packed into a unit square.*

is obvious, but not proved so far for  $p > 2$ .

## 5.6 A Constructive Branch-and-Bound Algorithm

Within this section we describe a branch-and-bound algorithm which is based on the so-called *contour concept*. Thereby the rectangles are placed successively above the current *contour*. This approach allows to regard various placement constraints and to derive fast heuristics. Frequently, such kind of algorithm is named *skyline algorithm*.

The following b&b algorithm is formulated to be applied to a 2OKP, but it can straightforwardly be used for the OPP as well.

### 5.6.1 The Contour Concept

We assume again that any rectangular piece can be packed at most once. To immediately identify inevitable waste, we consider the reduced set of potential allocation points. Similar to Sect. 2.7, we initially define sets of potential allocation points

$$S(w, W) := \{r : r = \sum_{i \in I} w_i a_i \leq W, a_i \in \{0, 1\}, i \in I\},$$

and

$$S(h, H) := \{r : r = \sum_{i \in I} h_i a_i \leq H, a_i \in \{0, 1\}, i \in I\}.$$

Assigned to these sets, let

$$p_w(s) := \max\{r \in S(w, W) : r \leq s\}, \quad p_h(s) := \max\{r \in S(h, H) : r \leq s\}.$$

Then, the reduced sets of potential allocation points are as follows:

$$\begin{aligned}\widetilde{S}(w, W) &:= \{p_w(W - r) : r \in S(w, W)\}, \\ \widetilde{S}(h, H) &:= \{p_h(H - r) : r \in S(h, H)\}.\end{aligned}$$

Moreover, let

$$\widetilde{p}_w(x) := \max\{r \in \widetilde{S}(w, W) : r \leq x\}, \quad \widetilde{p}_h(y) := \max\{s \in \widetilde{S}(h, H) : s \leq y\}.$$

In the following, we assume  $W = p_w(W)$  and  $H = p_h(H)$  which can easily be derived by replacing  $W$  by  $p_w(W)$  and  $H$  by  $p_h(H)$ . For a given pattern

$$A(\widetilde{I}) = \bigcup_{i \in \widetilde{I}} R_i(x_i, y_i) = \bigcup_{i \in \widetilde{I}} \{(x, y) : x_i \leq x \leq x_i + w_i, y_i \leq y \leq y_i + h_i\}$$

with  $\widetilde{I} \subseteq I$  its *contour area* is defined by

$$U(A(\widetilde{I})) := \{(x, y) \in \mathbb{R}_+^2 : \exists i \in \widetilde{I} \text{ with } x \leq x_i + w_i, y \leq y_i + h_i\}.$$

Note that  $U(A(\widetilde{I}))$  encloses the inner waste. Then, the curve

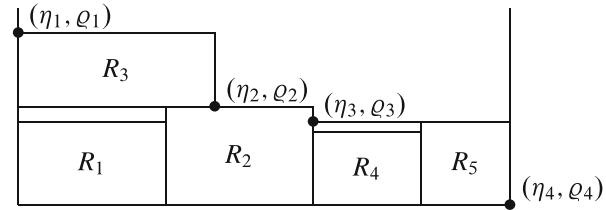
$$K(A(\widetilde{I})) := \text{fr} (\text{cl} (\{(x, y) : 0 \leq x \leq W, 0 \leq y\} \setminus U(A(\widetilde{I}))))$$

is named the *contour* (or *skyline*) of pattern  $A(\widetilde{I})$  where  $\text{fr}(\cdot)$  denotes the frontier and  $\text{cl}(\cdot)$  the closure of a set. As a convention, we use  $A(\emptyset) = \emptyset$ ,  $U(A(\emptyset)) = \emptyset$ , and  $K(\emptyset) = \{(x, y) : 0 \leq x \leq W, 0 \leq y, x(W - x)y = 0\}$ . According to the approach described in the following, the region  $U(A(\widetilde{I})) \setminus A(\widetilde{I})$  is considered as waste.

We say, contour area  $U_1$  covers contour area  $U_2$  if  $U_2 \subseteq U_1$  holds. Contour area  $U(A)$  can be described by means of the sequence

$$\{(\eta_i, \varrho_i) : i = 1, \dots, n + 1\}$$

**Fig. 5.8** Monotone packing according to the contour concept



of  $n + 1$  points (where  $n = n(A)$ ,  $n \geq 1$ ) in a unique manner according to  $0 = \eta_1 < \eta_2 < \dots < \eta_n < \eta_{n+1} = W$ ,  $\varrho_1 > \varrho_2 > \dots > \varrho_n \geq \varrho_{n+1} = 0$ , and

$$U(A) = \bigcup_{i=2}^{n+1} \{(x, y) \in \mathbb{R}_+^2 : x \leq \eta_i, y \leq \varrho_{i-1}\}.$$

The points  $(\eta_i, \varrho_i)$  are also named *contour points*.

A pattern  $A_k = \{R_{ij}(x_{ij}, y_{ij}) : j = 1, \dots, k\}$  with  $k$  items, placed in the sequence  $i_1, \dots, i_k$ , is called *monotone* if

$$R_{ij}(x_{ij}, y_{ij}) \cap \text{int } U(A_{j-1}) = \emptyset, \quad j = 2, \dots, k,$$

holds. Figure 5.8 shows a monotone pattern.

The *contour concept* consists in the construction of sequences of monotone and normalized patterns. To this end, further pieces are allocated (packed) at the contour points. This strategy is based on

**Theorem 5.13** *For every feasible packing  $A = \{R_i(x_i, y_i) : i = 1, \dots, k\}$  of  $k$  rectangles  $R_i$ ,  $i = 1, \dots, k$ , within a containing rectangle (or a rectangular strip) there exists a permutation  $(i_1, \dots, i_k) \in \Pi(1, \dots, k)$  such that  $A_r = \{R_{ij}(x_{ij}, y_{ij}) : j = 1, \dots, r\}$  is monotone for  $r = 1, \dots, k$ .*

The theorem can be proved by induction. Hence, to compute an optimal pattern it suffices to consider monotone normalized patterns.

## 5.6.2 A Branch-and-Bound Algorithm

Next, we describe a branch-and-bound algorithm designed for the 2OKP which is based on the contour concept (Fig. 5.9). The proposed algorithm can immediately be applied to the OPP using  $p_i = w_i h_i$  for all  $i \in I$ . The definition of subproblems (*branching*) is oriented on the contour points and the pieces which can be placed there. Bounds for subproblems and other fathoming tests are addressed later. To simplify the description of the algorithm we apply the *LIFO* strategy (*Last In First Out*) also known as *depth first search*. An adaptation to the *best bound search* or other branching rules can be done straightforwardly. The number  $k$  denotes the

**Branch-and-bound algorithm for the 2OKP**

- (1) **Initialization** Initialize the empty contour  $K_0$  and the branching tree  $I_0 := \emptyset$ . Set  $k := 1$ .
- (2) **Branching with respect to allocation points** (After packing  $k - 1$  pieces, an allocation point for the  $k$ -th piece has to be chosen.)  
 If all allocation points of contour  $K_{k-1}$  have already been considered, then *back track*:  $k := k - 1$ , if  $k = 0$ , then stop, otherwise go to (3).  
 Otherwise, choose an allocation point  $(\eta_k, \varrho_k)$  of  $K_{k-1}$ , not considered so far, for the  $k$ -th piece.
- (3) **Branching with respect to pieces** (After fixing the next allocation point, a next piece has to be selected.)  
 If all pieces of  $I \setminus I_{k-1}$  have been considered as  $k$ -th piece for allocation point  $(\eta_k, \varrho_k)$ , then *back track*: go to (2).  
 Otherwise, choose a feasible piece  $i_k \in I \setminus I_{k-1}$ , not considered so far as  $k$ -th piece, and place it at allocation point  $(\eta_k, \varrho_k)$ : set  $I_k := I_{k-1} \cup \{i_k\}$ ,  $A_k(I_k) := A_{k-1}(I_{k-1}) \cup \{R_{i_k}(\eta_k, \varrho_k)\}$ .  
 If an improved solution is constructed, save it.  
 If  $k = m$ , then stop.  
 Compute the new contour  $K_k$  and set  $k := k + 1$ .
- (4) **Bounding, equivalence and dominance tests**  
 If the current subproblem can be fathomed by some bounding, equivalence, or dominance test, then *back track*: set  $k := k - 1$  and go to (3). Otherwise, go to (2).

**Fig. 5.9** Branch-and-bound algorithm for the 2OPP

branching depth which is equal to the number of placed pieces. Moreover, index set  $I_k$  represents the pieces already packed.

Appropriate techniques to compute upper bounds are considered in the next subsection. Some motivation to use equivalence and dominance tests is given in Sect. 5.6.4.

The proposed branch-and-bound algorithm represents one variant to search for an optimal pattern based on the contour concept. Another possibility results if at first a piece is chosen and thereafter appropriate allocation points are selected.

### 5.6.3 Upper Bounds

Within the b&b algorithm, proposed in Sect. 5.6.2, various upper bounds for the maximal value  $z^*$  can be used. Their performance is, in general, proportional to the computational effort needed for their calculation.

Let  $A_k$  be a packing of the rectangles  $R_i$ ,  $i \in I_k = \{i_1, \dots, i_k\}$ . We denote the area of  $U(A_k)$  by  $\alpha_k$  and the value of  $A_k$  by  $z_k := \sum_{i \in I_k} c_i$ . A natural bound is the

material or area bound  $ub_A$  which is defined by the 0/1 knapsack problem

$$ub_A(A_k) := z_k + \max\left\{ \sum_{i \in I \setminus I_k} c_i a_i : \sum_{i \in I \setminus I_k} w_i h_i a_i \leq WH - \alpha_k, a_i \in \{0, 1\}, i \in I \setminus I_k \right\}.$$

A weaker bound, but obtainable by less computational effort, results from its continuous relaxation

$$ub_0(A_k) := z_k + \max\left\{ \sum_{i \in I \setminus I_k} c_i a_i : \sum_{i \in I \setminus I_k} w_i h_i a_i \leq WH - \alpha_k, 0 \leq a_i \leq 1, i \in I \setminus I_k \right\}.$$

When allocating piece  $R_i(x_i, y_i)$  we can possibly assign an enlarged area to  $R_i$  due to the usage of the reduced set of potential allocation points. In that case, modified coordinates of the ‘upper right corner’  $(\tilde{x}_i, \tilde{y}_i)$  can be defined according to

$$\tilde{x}_i := W - \tilde{p}_w(W - x_i - w_i), \quad \tilde{y}_i := H - \tilde{p}_h(H - y_i - h_i).$$

Then,  $R_i(x_i, y_i)$  can be replaced by the possibly enlarged rectangle

$$\tilde{R}_i(x_i, y_i) := \{(r, s) \in \mathbb{R}^2 : x_i \leq r \leq \tilde{x}_i, y_i \leq s \leq \tilde{y}_i\}$$

which can lead to a modified pattern  $\tilde{A}(\tilde{I}) = \{\tilde{R}_i(x_i, y_i) : i \in \tilde{I}\}$  and increased contour area

$$U(\tilde{A}(\tilde{I})) := \{(x, y) \in \mathbb{R}_+^2 : \exists i \in I \text{ with } x \leq \tilde{x}_i, y \leq \tilde{y}_i\}$$

for some  $\tilde{I} \subseteq I$ . Because of the possibly enlarged area  $\tilde{\alpha}_k$  of  $U(\tilde{A}(I_k))$ , we obtain the improved bound  $\tilde{ub}_A(A_k) := ub_A(\tilde{A}_k)$ .

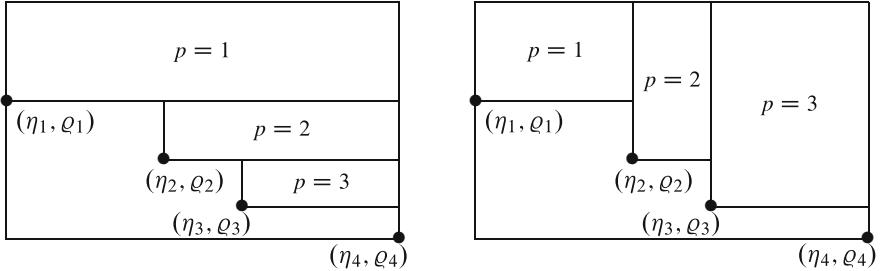
*Example 5.5* Consider a rectangle of width  $W = 15$  and height  $H = 13$  and quadratic pieces  $R_i$  with edge lengths  $w_i = 6, i \in \{1, 2, 3\}$ ,  $w_i = 5, i \in \{4, 5, 6\}$ , and  $w_i = 4, i \in \{7, 8, 9\}$ . We aim to find a pattern with maximal area utilization. As example, let us consider the packing  $A_2 = \{R_1(0, 0), R_2(6, 0)\}$  which has the contour points  $(\eta_1, \rho_1) = (0, 6)$ ,  $(\eta_2, \rho_2) = (12, 0)$ , and  $(\eta_3, \rho_3) = (15, 0)$ . Then, we obtain

$$ub_A(A_2) = 2 \cdot 36 + \max\left\{ \sum_{i=3}^9 w_i^2 a_i : \sum_{i=3}^9 w_i^2 a_i \leq 123, a_i \in \{0, 1\}, i = 3, \dots, 9 \right\} = 195.$$

The reduced sets of potential allocation coordinates are the following:

$$\tilde{S}(w, 15) = \{0, 4, 5, 6, 9, 10, 11, 15\} \quad \text{and} \quad \tilde{S}(w, 13) = \{0, 4, 5, 6, 8, 9, 13\}.$$

Because of  $\tilde{p}_w(W - 12) = 0$  and  $\tilde{p}_h(H - 6) = 6$ , we obtain the new contour points  $(\tilde{\eta}_1, \tilde{\rho}_1) = (0, 7)$ , and  $(\tilde{\eta}_2, \tilde{\rho}_2) = (15, 0)$  of  $\tilde{A}_2$ . Therefore, instead of the upper



**Fig. 5.10** Horizontal and vertical bar relaxation

bound  $ub_A(A_2)$  the improved bounds  $ub_0(\tilde{A}_2) = 162$  and

$$ub_A(\tilde{A}_2) = 2 \cdot 36 + \max\left\{\sum_{i=3}^9 w_i^2 a_i : \sum_{i=3}^9 w_i^2 a_i \leq 90, a_i \in \{0, 1\}, i = 3, \dots, 9\right\} = 158$$

are available. ■

A more costly, but in general stronger, upper bound can be obtained using the bar relaxation as proposed in Sect. 5.3.2 leading to a *binary multi cutting stock problem*. We consider a packing  $A_k = A(I_k)$  with contour points  $\{(\eta_i, \varrho_i) : i = 1, \dots, n+1\}$ .

The remaining unused area is partitioned horizontally and vertically as illustrated in Fig. 5.10. Associated to the partitions, we define horizontal bar patterns  $a^{pk}$  according to

$$\sum_{i \in I} w_i a_{pk} \leq W - \eta_p, \quad k \in \bar{K}_p, \quad p = 1, \dots, n,$$

and vertical bar patterns  $A^{pj}$  according to

$$\sum_{i \in I} h_i A_{pj} \leq H - \varrho_p, \quad j \in \bar{J}_p, \quad p = 1, \dots, n,$$

as well as nonnegative integer variables  $t_{pk}$ ,  $k \in \bar{K}_p$ , and  $\tau_{pj}$ ,  $j \in \bar{J}_p$ ,  $p = 1, \dots, n$ , respectively, and obtain a model adapted to  $A(I_k)$ :

#### Combined bar relaxation for OKP subproblems

$$z = \max \sum_{i \in I_k} c_i + \sum_{p=1}^n \sum_{k \in \bar{K}_p} \sum_{i \in I \setminus I_k} \frac{c_i}{h_i} a_{ipk} t_{pk} \quad \text{s.t.} \quad (5.17a)$$

$$\sum_{p=1}^q \sum_{k \in \bar{K}_p} t_{pk} \leq H - \varrho_q, \quad \sum_{p=q}^n \sum_{j \in \bar{J}_p} \tau_{pj} \leq W - \eta_q, \quad q = 1, \dots, n, \quad (5.17b)$$

$$\sum_{p=1}^n \sum_{k \in \bar{K}_p} a_{ipk} t_{pk} \leq h_i, \quad i \in I \setminus I_k, \quad (5.17c)$$

$$\sum_{p=1}^n \sum_{k \in \bar{K}_p} w_i a_{ipk} t_{pk} = \sum_{p=1}^n \sum_{j \in \bar{J}_p} h_i A_{ipj} \tau_{pj}, \quad i \in I \setminus I_k, \quad (5.17d)$$

$$t_{pk} \in \mathbb{Z}_+, \quad k \in \bar{K}_p, \quad \tau_{pj} \in \mathbb{Z}_+, \quad j \in \bar{J}_p, \quad p = 1, \dots, n.$$

The adaptation of model (5.13) on a nontrivial contour leads to a reformulation of the restrictions, in particular of (5.17b). Due to the reduced lengths a reduction of the number of patterns can appear. The other restrictions are similar to those in model (5.13). The computation of the related LP bound can be done using the column generation technique (Sect. 4.3.3).

#### 5.6.4 Equivalence and Dominance

Besides the application of appropriate bounds the usage of equivalence and dominance considerations can be meaningful to fathom subproblems within a b&b approach.

In order to save computational effort several equivalence and dominance relations are considered within the relevant literature. Here we address only two examples for motivation. Elaborated investigations in this respect can be found in [29].

Let us consider two sequences  $R_1, R_2, R_3$  and  $R_1, R_3, R_2$  to allocate three rectangular pieces where  $R_2$  is placed *right to* and  $R_3$  *above*  $R_1$ . Such a scenario can arise, e.g., within a b&b algorithm.

If  $h_2 \leq h_1$  and  $w_3 \leq w_1$ , then we obtain the patterns  $A_3 = \{R_1(0, 0), R_2(w_1, 0), R_3(0, h_1)\}$  and  $\tilde{A}_3 = \{R_1(0, 0), R_3(0, h_1), R_2(w_1, 0)\}$ . Both patterns contain the same rectangles and possess the same contour. Hence, both define the same subproblem in a b&b algorithm. Obviously, only one of the two subproblems should be considered in an efficient b&b algorithm, and the second should be fathomed because it is equivalent to the first one.

If  $h_2 > h_1$  and  $w_3 \leq w_1$ , then we obtain the patterns  $A_3 = \{R_1(0, 0), R_2(w_1, 0), R_3(0, h_2)\}$  and  $\tilde{A}_3 = \{R_1(0, 0), R_3(0, h_1), R_2(w_1, 0)\}$ . In this case, we have  $U(A_3) \supset U(\tilde{A}_3)$ . Hence, to each feasible packing of  $k \geq 4$  pieces which contains  $A_3$  as a sub-packing, there exists a pattern with non-smaller value which contains  $\tilde{A}_3$  as a subpattern. In this sense,  $A_3$  *dominates*  $\tilde{A}_3$ . Hence, there is no need to consider the subproblem which is defined by  $A_3$ . Summarizing we have that only dominant patterns should be used in a b&b algorithm.

## 5.7 Further Approaches for the OPP

We briefly address some further approaches for solving the OPP.

### A two-Step b&b Method

Clautiaux et al. [9] proposed a b&b algorithm for the 2OPP being a two-step b&b method based on a kind of contiguous relaxation. In the first step (*outer* b&b method), all solutions of a relaxed problem are enumerated. For each solution found, a second enumeration method is run (*inner* b&b method) to seek a solution of the initial problem corresponding to the current solution. The relaxation consists in considering each item  $w_i \times h_i$  as  $h_i$  strips of unit height and width  $w_i$  similar to the bar relaxation. A constraint states that all strips pertaining to a given item have to be packed at the same  $x$ -coordinate, even if they are not vertically contiguous. Thus, the resulting problem cannot be solved as a 1BPP. Using this relaxation, the problem consists in seeking a suitable set of  $x$ -coordinates (*outer* b&b method). When a solution is found for the relaxed problem (i.e., a valid list of  $x$ -coordinates for the items), the *inner* b&b method searches for a set of  $y$ -coordinates to obtain a solution for the initial two-dimensional problem.

### A Constraint Programming-Based b&b Algorithm

*Constraint programming* is a programming paradigm aimed at solving combinatorial optimization problems that can be described by a set of variables, a set of possible values for each variable, and a set of constraints between the variables. The set of possible values of a variable is called the *variable domain*. A constraint between variables expresses which combination of values for the variables are allowed. The question to be answered is whether there exists an assignment of values to variables, such that all constraints are satisfied. The power of a constraint programming method lies mainly in the fact that constraints can be used in an active process termed *constraint propagation* where certain deductions are performed in order to reduce the computational effort. Constraint programming removes values from the domains, deduces new constraints, and detects inconsistencies.

For instance, in [10] such a constraint programming based b&b algorithm is proposed. Let  $(x_i, y_i)$  denote the allocation point of item  $i$ ,  $i \in I$ . The variable domains of  $x_i$  and  $y_i$  are denoted by

$$D(x_i) := [\underline{x}_i, \bar{x}_i] \quad \text{and} \quad D(y_i) := [\underline{y}_i, \bar{y}_i],$$

respectively. Initially, we have

$$D(x_i) = [0, W - w_i] \quad \text{and} \quad D(y_i) = [0, H - h_i].$$

According to the nonlinear OPP model, Sect. 5.13, we associate each pair of items  $i$  and  $j$  with the following constraint:

$$(\bar{x}_i + w_i \leq \underline{x}_j) \vee (\bar{x}_j + w_j \leq \underline{x}_i) \vee (\bar{y}_i + h_i \leq \underline{y}_j) \vee (\bar{y}_j + h_j \leq \underline{y}_i).$$

The model is sufficient to ensure that the solution is valid, once the domains of variables have been reduced to only one value such that all constraints are satisfied (cf. the Padberg-type model). According to respective branching rules, the variable domains can be updated. For example, if  $(x_i + w_i \leq x_j)$  is demanded, then

$$\bar{x}_i := \min\{\bar{x}_i, \bar{x}_j - w_i\} \quad \text{and} \quad \underline{x}_j := \max\{\underline{x}_j, \underline{x}_i + w_i\}$$

follows. For a detailed description, we refer to [10] where a cumulative resource-constraint scheduling relaxation (a contiguous relaxation) is used to define bounds.

## 5.8 Exercises

**Exercise 5.1** Show, if the packing pattern  $A(I) = \{R_i(x_i, y_i) : i \in I\}$  is bottom-left justified, then we have  $x_i \in S(w, W)$  and  $y_i \in S(h, H)$  for all  $i \in I$  where  $S(w, W)$  and  $S(h, H)$  denote the sets of potential allocation points for  $w = (w_1, \dots, w_m)^T$  and  $h = (h_1, \dots, h_m)^T$ , respectively. Thus, the pattern is normalized.

**Exercise 5.2** Consider the 2OPP instance  $E$  with  $W = 15$ ,  $H = 13$ , and eight quadratic pieces:  $w_i = 6$  for  $i \in \{1, 2\}$ ,  $w_i = 5$ ,  $i \in \{3, 4, 5\}$ , and  $w_i = 4$  for  $i \in \{6, 7, 8\}$ . Compute the optimal value of the (LP) relaxation of the horizontal bar relaxation. Additionally, compute the optimal value of the (LP) relaxation of the vertical bar relaxation.

**Exercise 5.3** Let  $E = (m, W, H, w, h)$  be an instance of the 2OPP. Furthermore, let

$$W^* := \max\left\{\sum_{i \in I} w_i a_i : \sum_{i \in I} w_i a_i \leq W, a_i \in \mathbb{B}, i \in I\right\},$$

$$H^* := \max\left\{\sum_{i \in I} h_i a_i : \sum_{i \in I} h_i a_i \leq H, a_i \in \mathbb{B}, i \in I\right\}.$$

Show, if  $E$  is feasible, then the unused area is not smaller than  $WH - W^*H^*$ .

**Exercise 5.4** Determine, how many rectangles of size  $8 \times 3$  can be placed without overlap on a square  $11 \times 11$ ? Rotation of items is permitted.

**Exercise 5.5** Show that  $p_i = w_i h_i$  or  $p_i = 1$  for all  $i \in I$  can be used in the target function (5.2a) of the Beasley-type model (5.2) to get a decision criterion for the 2OPP.

**Exercise 5.6** Show that  $p_i = w_i h_i$  or  $p_i = 1$  for all  $i \in I$  can be used in the target function (5.4a) of the Padberg-type model (5.4) to get a decision criterion for the 2OPP.

**Exercise 5.7** Compute the optimal value  $z_{LP}^{Kan}$  of the LP relaxation of model (5.7) if  $p_i = w_i h_i$  for all  $i \in I$  is assumed.

**Exercise 5.8** Solve the horizontal and vertical bar relaxation for the OPP instances  $E$  and  $E'$  with container  $W = 16$  and  $H = 12$ . Both instances possess two items  $6 \times 6$ , two items  $4 \times 4$ , and two items  $8 \times 3$ . Moreover,  $E$  has four items  $5 \times 2$  whereas  $E'$  has one item  $5 \times 8$ .

## 5.9 Solutions

**To Exercise 5.1** If the pattern  $A(I)$  is bottom-left justified, then no item can be shifted ‘down-’ or ‘leftward’ since it either is tangent to the boundary of the container or it is supported from below or left by another item. Therefore, for the allocation point  $(x_i, y_i)$  of item  $i$  we have  $x_i = 0$  or  $x_j + w_j = x_i$  for some item  $j \neq i$ , and similar,  $y_i = 0$  or  $y_k + h_k = y_i$  for some item  $k \neq i$ . Since this is true for all pieces  $i \in I$ ,  $x_i$  (and  $y_i$ ) is a combination of item widths (heights). Thus,  $x_i \in S(w, W)$  and  $y_i \in S(h, H)$ .

**To Exercise 5.2** Note that  $\sum_{i \in I} w_i^2 = WH = 195$  holds. According to the horizontal bar relaxation, we obtain a one-dimensional problem with  $W = 15$ ,  $w = (6, 5, 4)^T$ , and demand vector  $b = (12, 15, 12)^T$ . Using twelve times the widths combination  $6 + 5 + 4$  and once  $5 + 5 + 5$  (this is possible because of three squares  $5 \times 5$ ), we obtain  $z_{LP}^{hor} = z^{hor} = 195$ . Hence, the bound does not provide a decision. But it is easy to see that the OPP instance is infeasible.

For the vertical bar relaxation we have to consider (binary) bar patterns of maximum length  $H = 13$ . Since there does not exist any combination of pieces having total length 13 which includes  $w_1 = 6$ , waste must necessarily occur. Therefore, the LP bound is larger than  $W = 15$  and the infeasibility of  $E$  is proven.

**To Exercise 5.3** For every arrangement  $A(I)$  there exists a normalized pattern  $A'(I)$ . The largest width used in  $A'(I)$  is not larger than  $W^*$ , and the largest height not higher than  $H^*$ . Therefore, the unused area is at least  $WH - W^*H^*$ .

**To Exercise 5.4** Obviously, four items can be packed. Because of the area (material) bound  $\lfloor 121/24 \rfloor = 5$  another criterion (e.g. the combined bar relaxation) has to be used to show that 4 is maximal.

**To Exercise 5.5** Obviously, a 2OPP instance is feasible if and only if  $I^* = I$  which means that the optimal value has to be equal to the total piece area (if  $p_i = w_i h_i$ ), or to the total number ( $m$ ) of items (if  $p_i = 1$ ).

**To Exercise 5.6** Similar to Exercise 5.5.

**To Exercise 5.7** Because of  $p_i = w_i h_i$  for all  $i \in I$  and restriction (5.7b), we have

$$\sum_{i \in I} \frac{p_i}{h_i} \sum_{k \in K} a_{ik} = \sum_{i \in I} \sum_{k \in K} w_i a_{ik} \leq \sum_{k \in K} W = WH.$$

In case of  $A = \sum_{i \in I} w_i h_i \leq WH$ , we set  $a_{ik} := h_i/H$  for all  $i$  and  $k$ . Then restrictions (5.7b) and (5.7c) are fulfilled, and the objective value results to  $A$  being maximal.

If  $A > WH$ , then we define  $a_{ik} := h_i W/A$  for all  $i$  and  $k$ . It is easy to verify that this setting is feasible with value  $WH$  equal to the upper bound. Thus, formula (5.8) is proved.

**To Exercise 5.8** We use a ‘compact’ writing of the patterns regarding the frequency a piece type is available. Because of given instance  $E$  with  $W = 16$ ,  $H = 12$ , two items  $6 \times 6$ , two items  $4 \times 4$ , two items  $8 \times 3$ , four items  $5 \times 2$ , the input data for the horizontal bar relaxation result to  $L = 16$ ,  $\ell = (6, 4, 8, 5)^\top$ ,  $b = (12, 8, 6, 8)^\top$ . Moreover, we additionally have to regard  $a_i \leq 2$  for  $i = 1, 2, 3$ , and  $a_4 \leq 4$  (because of the ‘compact’ consideration of patterns). An optimal solution is as follows: four times  $(2, 1, 0, 0)^\top$ , four times  $(1, 0, 0, 2)^\top$ , four times  $(0, 2, 1, 0)^\top$ , and once  $(0, 0, 2, 0)^\top$ .

The input data for the vertical bar relaxation are  $L = 12$ ,  $\ell = (6, 4, 3, 2)^\top$ ,  $b = (12, 8, 16, 20)^\top$ , and we have to regard  $a_i \leq 2$  for  $i = 1, 2, 3$ , and  $a_4 \leq 4$ . Then, a corresponding optimal solution is as follows: eight times  $(1, 0, 2, 0)^\top$ , four times  $(1, 0, 0, 3)^\top$ , and four times  $(0, 2, 0, 2)^\top$ .

In both cases, all items are placed—ineasibility is not discovered.

Considering  $E'$  with  $W = 16$ ,  $H = 12$ , two items  $6 \times 6$ , two items  $4 \times 4$ , two items  $8 \times 3$ , one item  $5 \times 8$ , the input data for the horizontal bar relaxation are  $L = 16$ ,  $\ell = (6, 4, 8, 5)^\top$ ,  $b = (12, 8, 6, 8)^\top$ , and we have to regard  $a_i \leq 2$  for  $i = 1, 2, 3$ , and  $a_4 \leq 1$ . Because of  $a_4 \leq 1$  there does not exist any trimless combination of pieces with total length 16 and  $a_4 = 1$ . Therefore, the instance is infeasible and the optimal value of the horizontal bar relaxation is less than  $WH$ .

The input data for the vertical bar relaxation are  $L = 12$ ,  $\ell = (6, 4, 3, 8)^\top$ ,  $b = (12, 8, 16, 5)^\top$ , and we have to regard  $a_i \leq 2$  for  $i = 1, 2, 3$  and  $a_4 \leq 1$ . Pattern  $(0, 1, 0, 1)^\top$  can be used at most five times and, because of  $a_2 \leq 2$ , the remaining three pieces of length 4 cannot be combined with other items without causing trim loses. Therefore, the instance is infeasible and the optimal value of the vertical bar relaxation is less than  $WH$ .

In both cases infeasibility is discovered.

## References

1. R. Alvarez-Valdes, F. Parreno, J.M. Tamarit, A branch and bound algorithm for the strip packing problem. OR Spektrum **31**, 431–459 (2009)
2. R. Baldacci, M.A. Boschetti, A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. Eur. J. Oper. Res. **183**(3), 1136–1149 (2007)
3. J.E. Beasley, Bounds for two-dimensional cutting. J. Oper. Res. Soc. **36**(1), 71–74 (1985)
4. J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure. Oper. Res. **33**(1), 49–64 (1985)
5. G. Belov, H. Rohling, LP bounds in an interval-graph algorithm for orthogonal-packing feasibility. Oper. Res. **61**(2), 483–497 (2013)

6. G. Belov, V. Kartak, H. Rohling, G. Scheithauer, One-dimensional relaxations and LP bounds for orthogonal packing. *Int. Trans. Oper. Res.* **16**, 745–766 (2009)
7. G. Belov, V. Kartak, H. Rohling, G. Scheithauer, Conservative scales in packing problems. *OR Spektrum* **35**(2), 505–542 (2013). Online 2011
8. T. Buchwald, G. Scheithauer, A 5/9 theorem on packing squares into a unit square. Preprint MATH-NM-04-2016, Technische Universität Dresden (2016)
9. F. Clautiaux, J. Carlier, A. Moukrim, A new exact method for the two-dimensional bin-packing problem with fixed orientation. *OR Lett.* **35**, 357–364 (2007)
10. F. Clautiaux, A. Jouplet, J. Carlier, A. Moukrim, A new constraint programming approach for the orthogonal packing problem. *Comput. Oper. Res.* **35**(3), 944–959 (2008)
11. S.P. Fekete, J. Schepers, A general framework for bounds for higher-dimensional orthogonal packing problems. *Math. Methods Oper. Res.* **60**(2), 311–329 (2004)
12. S.P. Fekete, J. Schepers, J. van der Veen, An exact algorithm for higher-dimensional orthogonal packing. *Oper. Res.* **55**(3), 569–587 (2007)
13. E.P. Ferreira, J.F.C. Oliveira, Graph based algorithm for the non-guillotinable two-dimesional packing problem. Technical report, Universidad do Porto (2008)
14. K. Fujiyoshi, Ch. Kodama, A. Ikeda, A fast algorithm for rectilinear block packing based on selected sequence-pair. *Integration* **40**(3), 274–284 (2007)
15. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, vol. 57 Annals of Discrete Mathematics (Elsevier, Amsterdam, 2004)
16. S. Hougaard, On packing squares into a rectangle. *Comput. Geom.* **44**(8), 456–463 (2011)
17. K. Jansen, G. Zhang, Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica* **47**, 323–342 (2007)
18. M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, H. Nagamochi, Exact algorithms for the two-dimensional strip packing problem with and without rotations. *Eur. J. Oper. Res.* **198**, 73–83 (2009)
19. D.J. Kleitman, M.M. Krieger, An optimal bound for two dimensional bin packing, in *Proc. 16th Ann. Symp. on Found. of Comp. Sci., IEEE Long Beach* (1975), pp. 163–168
20. K. Li, K.H. Cheng, On three-dimensional packing. *SIAM J. Comput.* **19**(5), 847–867 (1990)
21. S. Martello, M. Monaci, D. Vigo, An exact approach to the strip packing problem. *INFORMS J. Comput.* **15**(3), 310–319 (2003)
22. A. Martin, General mixed integer programming: computational issues for branch-and-cut algorithms, in *Computational Combinatorial Optimization*, ed. by M. Jünger, D. Naddef (Springer, Berlin/Heidelberg, 2001), pp. 1–25
23. A. Meir, L. Moser, On packing of squares and cubes. *J. Comb. Theory* **5**, 126–134 (1968)
24. R.D. Meller, W. Chen, H.D. Sherali, Applying the sequence-pair representation to optimal facility layout designs. *OR Lett.* **35**(5), 651–659 (2007)
25. M. Monaci, Algorithms for packing and scheduling problems. PhD thesis, Bologna (2001)
26. H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, Vlsi module placement based on rectangle-packing by the sequence-pair. *IEEE Trans. Comput. Aided Des.* **15**(12), 1518–1524 (1996)
27. M. Padberg, Packing small boxes into a big box. *Math. Methods Oper. Res.* **52**, 1–21 (2000)
28. D. Pisinger, M. Sigurd, Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS J. Comput.* **19**(1), 36–51 (2007)
29. G. Scheithauer, LP-based bounds for the container and multi-container loading problem. *Int. Trans. Oper. Res.* **6**(2), 199–213 (1999)
30. G. Scheithauer, J. Terno, Modelling of packing problems. *Optimization* **28**, 63–84 (1993)
31. H. Simonis, B. O’Sullivan, Search strategies for rectangle packing, in *Principles and Practice of Constraint Programming*, ed. by P.J. Stuckey (Springer, New York, 2008), pp. 52–66
32. A. Steinberg, A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.* **26**(2), 401–409 (1997)

# Chapter 6

## Optimal Guillotine Cutting

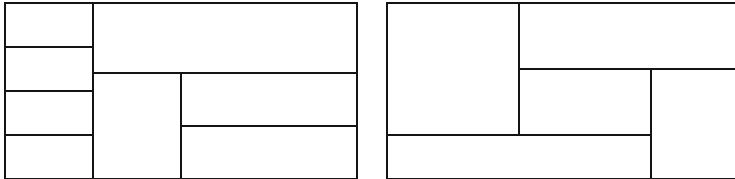
Within this chapter we address those two- and higher-dimensional cutting and packing problems where the guillotine cutting condition has to be regarded. That means, given a pattern then the desired products can be obtained by a sequence of guillotine cuts. We will consider knapsack-type problems as well as cutting stock problems.

### 6.1 Problem Statements

In many branches such as furniture and glass industry, cutting problems of the following kind are of high interest: A given (larger) rectangular material of length  $L$  and width  $W$  has to be cut into (smaller) rectangular pieces of different types. Pieces of type  $i$  have length  $\ell_i$ , width  $w_i$ , and profit coefficient (value)  $c_i$ ,  $i \in I = \{1, \dots, m\}$ . The task is to find an orthogonal (cutting) pattern with maximal total value which possesses the guillotine-property, that means, which can be realized using only guillotine cuts. Or with other words, how many items of which types can be obtained applying only guillotine cuts while maximizing the profit?

Obviously, if we choose  $c_i = \ell_i w_i$ , we look for a pattern with minimal waste. Using different profit coefficients priority rules can be defined. Without loss of generality, we assume in the following that all input data are positive integers and that rotation of items is not permitted, if not stated otherwise.

Caused by technological (cutting) restrictions two main types of patterns can be differentiated: *guillotine patterns* and *non-guillotine patterns*. For short, we will also use the notation *G-pattern*. Applying an orthogonal *guillotine cut* (G-cut) a rectangle is dissected into two smaller rectangles. The resulting rectangles can further be dissected by again applying guillotine cuts, and so on, until a desired piece is obtained, or waste results. A guillotine pattern and a non-guillotine pattern are depicted in Fig. 6.1.



**Fig. 6.1** Guillotine and non-guillotine cutting

The restriction that only guillotine cuts are allowed is of high importance, for example, for wood processing, in glass cutting, and metallurgy.

The number of turnarounds of the guillotine cuts, which is called the *stage number*, is unlimited in the (*general*) *guillotine cutting problem* (Sect. 6.2). Frequently, the *2-stage guillotine cutting problem* (Sect. 6.3) and the *3-stage guillotine cutting problem* (Sect. 6.4) are considered due to the fewer technological effort within the cutting process. Further types of G-patterns are considered in Sect. 6.5. Thereafter, we address the guillotine knapsack problem with upper bounds where the occurrence of items of the same type is additionally limited. We describe some problem-specific models and approaches. Three-dimensional guillotine cutting problems as well as the cutting stock problem are considered, too. The more general non-guillotine orthogonal cutting problem is subject of Chap. 5.

## 6.2 General Guillotine Cutting

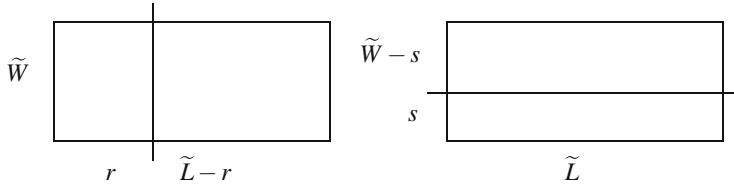
For simplicity, we assume that rotation is not permitted. This restriction is of importance for several applications due to the structure of the material (wood) or the direction of foiling (steel). The handling of rotatable pieces (rotation by  $90^\circ$ ) is subject of Exercise 6.2. The recurrence formulas below are based on those given in the fundamental work of Gilmore and Gomory [7].

We again characterize a feasible pattern by an  $m$ -dimensional nonnegative integer vector  $a = (a_1, \dots, a_m)^\top$ , i.e.  $a \in \mathbb{Z}_+^m$ , where  $a_i$  indicates how many items of type  $i$  are obtained. In order to derive a recurrence formula, we define

$$v(\tilde{L}, \tilde{W}) := \max \left\{ \sum_{i \in I} c_i a_i : a = (a_1, \dots, a_m)^\top \text{ is a feasible G-pattern for } \tilde{L} \times \cdot \tilde{W} \right\}$$

as the total profit of an optimal pattern of a rectangle with length  $\tilde{L}$  and width  $\tilde{W}$ ,  $\tilde{L} \in \{0, \dots, L\}$ ,  $\tilde{W} \in \{0, \dots, W\}$ .

Applying an (orthogonal) *guillotine cut* rectangle  $\tilde{L} \times \tilde{W}$  is either partitioned into two sub-rectangles of sizes  $r \times \tilde{W}$  and  $(\tilde{L} - r) \times \tilde{W}$ , or into two of sizes  $\tilde{L} \times s$  and  $\tilde{L} \times (\tilde{W} - s)$ , respectively, as shown in Fig. 6.2.



**Fig. 6.2** Applying a vertical or horizontal guillotine cut

According to the two possible directions to perform a G-cut, we obtain the following lower estimation of the optimal value  $v(\tilde{L}, \tilde{W})$ :

$$v(\tilde{L}, \tilde{W}) \geq \max \{g(\tilde{L}, \tilde{W}), h(\tilde{L}, \tilde{W})\}$$

where

$$g(\tilde{L}, \tilde{W}) := \max\{v(r, \tilde{W}) + v(\tilde{L} - r, \tilde{W}) : r \in \{1, \dots, \lfloor \tilde{L}/2 \rfloor\}\}$$

denotes the maximal profit obtainable performing a vertical G-cut, and

$$h(\tilde{L}, \tilde{W}) := \max\{v(\tilde{L}, s) + v(\tilde{L}, \tilde{W} - s) : s \in \{1, \dots, \lfloor \tilde{W}/2 \rfloor\}\}$$

that of a horizontal G-cut, respectively. Obviously, we have  $v(\tilde{L}, \tilde{W}) = 0$  if  $0 \leq \tilde{L} < \ell_{\min} := \min\{\ell_i : i \in I\}$  or  $0 \leq \tilde{W} < w_{\min} := \min\{w_i : i \in I\}$ . Thereby, we obtain a first recursion to compute  $v(\tilde{L}, \tilde{W})$  for all  $\tilde{L} \in \{0, \dots, L\}$  and  $\tilde{W} \in \{0, \dots, W\}$ :

$$v(\tilde{L}, \tilde{W}) := \max \{g(\tilde{L}, \tilde{W}), h(\tilde{L}, \tilde{W}), e(\tilde{L}, \tilde{W})\} \quad (6.1)$$

where term  $e(\tilde{L}, \tilde{W})$  represents some initialization:

$$e(\tilde{L}, \tilde{W}) := \max\{0, \max\{c_i : \ell_i \leq \tilde{L}, w_i \leq \tilde{W}, i \in I\}\}.$$

The examination of formula (6.1) has to be done, in principle, for every integer tuple  $(\tilde{L}, \tilde{W})$  with  $\ell_{\min} \leq \tilde{L} \leq L$  and  $w_{\min} \leq \tilde{W} \leq W$ .

As already seen for solution methods of the knapsack problem, a remarkable reduction of the computational effort can be achieved by using (reduced) sets of potential allocation points (cf. Sect. 2.7). Since all input data are integers, the function  $v : [0, L] \times [0, W] \rightarrow \mathbb{R}$  is piece-wise constant (a step-function) with characteristic jump discontinuities. To formulate a recursion based on sets of potential allocation points let

$$S(\ell, L) = \{r_j : j = 0, \dots, \alpha\} := \{r : r = \sum_{i \in I} \ell_i a_i, r \leq L, a_i \in \mathbb{Z}_+, i \in I\}$$

and

$$S(w, W) = \{s_k : k = 0, \dots, \beta\} := \{s : s = \sum_{i \in I} w_i a_i, s \leq W, a_i \in \mathbb{Z}_+, i \in I\}$$

denote respective sets in  $L$ - and  $W$ -direction. Without loss of generality, we can assume  $0 = r_0 < r_1 < \dots < r_\alpha \leq L$  and  $0 = s_0 < s_1 < \dots < s_\beta \leq W$  where  $\alpha := |S(\ell, L)| - 1$  and  $\beta := |S(w, W)| - 1$ . Using abbreviations

$$p_\ell(x) := \max \{r \in S(\ell, L) : r \leq x\}$$

and

$$p_w(x) := \max \{r \in S(w, W) : r \leq x\}$$

to indicate the maximal usable length with respect to  $S(\ell, L)$  and width with respect to  $S(w, W)$  not larger than  $x$ , respectively, we obtain the following recursion for the guillotine cutting problem:

$$\begin{aligned} v(r_j, s_k) := & \max \{e(r_j, s_k), v(r_{j-1}, s_k), v(r_j, s_{k-1}), \\ & \max_{r \in S(\ell, L)} \{v(r, s_k) + v(p_\ell(r_j - r), s_k) : r \leq r_j/2, r + p_\ell(r_j - r) = r_j\}, \\ & \max_{s \in S(w, W)} \{v(r_j, s) + v(r_j, p_w(s_k - s)) : s \leq s_k/2, s + p_w(s_k - s) = s_k\}, \\ & j = 1, \dots, \alpha, \quad k = 1, \dots, \beta. \end{aligned} \tag{6.2}$$

The explicit computation of  $e(r_j, s_k)$  for all  $j$  and  $k$  can easily be avoided by an appropriate initialization and a small modification of the recursion formula:

- Set  $v(r_j, s_k) := 0$ ,  $j = 0, \dots, \alpha$ ,  $k = 0, \dots, \beta$ .
- For  $i = 1, \dots, m$  set  $v(\ell_i, w_i) := c_i$ .

That leads to the recursion

$$\begin{aligned} v(r_j, s_k) := & \max \{v(r_{j-1}, s_k), v(r_j, s_{k-1}), \\ & \max_{r \in S(\ell, L)} \{v(r, s_k) + v(p_\ell(r_j - r), s_k) : r \leq r_j/2, r + p_\ell(r_j - r) = r_j\}, \\ & \max_{s \in S(w, W)} \{v(r_j, s) + v(r_j, p_w(s_k - s)) : s \leq s_k/2, s + p_w(s_k - s) = s_k\}, \\ & j = 1, \dots, \alpha, \quad k = 1, \dots, \beta, \end{aligned} \tag{6.3}$$

which yields the same results as recursion (6.2).

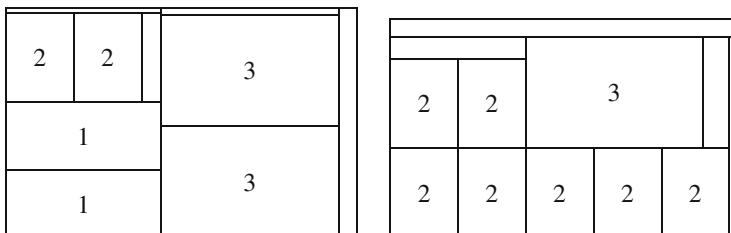
*Example 6.1* Rectangular pieces of size  $\ell_i \times w_i$  and value  $c_i$ ,  $i = 1, 2, 3$ , see Table 6.1, can be obtained by orthogonal guillotine cuts from a rectangular sheet (of raw material) of size  $L = 160$  and  $W = 105$ . Rotation of pieces is not permitted. Table 6.2 shows values  $v(r_j, s_k)$ . From row 5 downwards only the  $v$ -values in the

**Table 6.1** Input data of Example 6.1

| $i$ | $\ell_i$ | $w_i$ | $c_i$ |
|-----|----------|-------|-------|
| 1   | 71       | 31    | 8     |
| 2   | 31       | 41    | 5     |
| 3   | 81       | 51    | 16    |

**Table 6.2** Calculations according to recursion (6.3)

| $r_j \setminus s_k$ | 31 | 41 | 51 | 62 | 72 | 82 | 92 | 93 | 102 | 103 |
|---------------------|----|----|----|----|----|----|----|----|-----|-----|
| 31                  | 0  | 5  | 5  | 5  | 5  | 10 | 10 | 10 | 10  | 10  |
| 62                  | 0  | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20  | 20  |
| 71                  | 8  | 10 | 10 | 16 | 18 | 20 | 20 | 24 | 24  | 26  |
| 81                  | 8  | 10 | 16 | 16 | 18 | 24 | 26 | 26 | 32  | 32  |
| 93                  |    | 15 |    |    | 23 | 30 | 31 |    |     |     |
| 102                 |    |    |    | 21 |    |    |    | 34 |     | 36  |
| 112                 |    |    | 21 |    |    | 34 | 36 |    | 42  |     |
| 124                 |    | 20 |    |    | 28 | 40 | 41 |    |     |     |
| 133                 |    |    |    | 26 |    |    |    | 44 |     | 46  |
| 142                 | 16 |    |    | 32 | 36 |    |    | 48 |     | 52  |
| 143                 |    |    | 26 |    |    | 44 | 46 |    | 52  |     |
| 152                 |    |    |    |    |    |    |    | 50 | 56  | 58  |
| 155                 |    | 25 |    |    | 41 | 50 | 51 |    |     |     |



**Fig. 6.3** Optimal patterns for  $L \times W = 160 \times 105$  and  $L \times W = 160 \times 100$

jump discontinuities are given. The remaining values are determined by  $v(r_j, s_k) := \max \{ v(r_{j-1}, s_k), v(r_j, s_{k-1}) \}$ . We obtain the optimal value  $v(160, 105) = 58$ . Then, a corresponding pattern can be identified as follows: since  $v(160, 105) = v(152, 103)$ , we have two trimming strips of widths 8 and 2, respectively. Moreover, value  $v(152, 103)$  results as sum of  $v(71, 103)$  and  $v(81, 103)$ . When we continue to analyze how  $v(71, 103)$  and  $v(81, 103)$  are determined, we obtain the optimal guillotine cutting pattern depicted in Fig. 6.3.

Note that if another piece of raw material of size  $\tilde{L} \times \tilde{W}$  with  $\tilde{L} \leq L$  and  $\tilde{W} \leq W$  is available, then we can identify a corresponding optimal pattern based on Table 6.2 as well. For example, for  $\tilde{L} = 160$  and  $\tilde{W} = 100$  we find the optimal value  $v(160, 100) = v(155, 92) = 51$  and obtain the pattern shown in Fig. 6.3, too. ■

A straightforward implementation of recursions (6.2) or (6.3) requires  $O(\alpha \cdot \beta)$  (memory) space. Besides the values  $v(r_j, s_k)$ , corresponding index information should be saved to easily recover the origin of a value and the related cut position and cut direction.

In certain cases there is no interest to have the possibility to identify optimal patterns for smaller raw material sheets; only a solution for  $L \times W$  has to be found. In that case, reduced sets of potential allocation points are applicable. To transfer this concept (cf. Sect. 2.7) to the two-dimensional guillotine cutting problem, we define

$$S^{red}(\ell, L) = \{\tilde{r}_0, \tilde{r}_1, \dots, \tilde{r}_{\tilde{\alpha}}\} := \{p_{\ell}(L - r) : r \in S(\ell, L)\}$$

and

$$S^{red}(w, W) = \{\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{\tilde{\beta}}\} := \{p_w(W - s) : s \in S(w, W)\}.$$

Without loss of generality, we assume  $\tilde{r}_0 = 0 < \tilde{r}_1 < \dots < \tilde{r}_{\tilde{\alpha}} = r_{\alpha} \leq L$  and  $\tilde{s}_0 = 0 < \tilde{s}_1 < \dots < \tilde{s}_{\tilde{\beta}} = s_{\beta} \leq W$ . Furthermore, we define

$$p_{\ell}^{red}(x) := \max\{r \in S^{red}(\ell, L) : r \leq x\}$$

and

$$p_w^{red}(x) := \max\{s \in S^{red}(w, W) : s \leq x\}$$

to indicate the maximal usable length with respect to  $S^{red}(\ell, L)$  and width with respect to  $S^{red}(w, W)$  not larger than  $x$ , respectively. In this way, we obtain a recursion based on the reduced sets of potential allocation points  $S^{red}(\ell, L)$  and  $S^{red}(w, W)$ :

$$\begin{aligned} v(\tilde{r}_j, \tilde{s}_k) := & \max \left\{ e(\tilde{r}_j, \tilde{s}_k), \right. \\ & \max\{v(r, \tilde{s}_k) + v(p_{\ell}^{red}(\tilde{r}_j - r), \tilde{s}_k) : r \in S^{red}(\ell, L) \text{ with } r \leq \tilde{r}_j/2\}, \\ & \left. \max\{v(\tilde{r}_j, s) + v(\tilde{r}_j, p_w^{red}(\tilde{s}_k - s)) : s \in S^{red}(w, W) \text{ with } s \leq \tilde{s}_k/2\} \right\}, \\ & j = 1, \dots, \tilde{\alpha}, \quad k = 1, \dots, \tilde{\beta}. \end{aligned} \tag{6.4}$$

The explicit computation of  $e(\tilde{r}_j, \tilde{s}_k)$  for all  $j$  and  $k$  can again be replaced by

- Set  $v(\tilde{r}_j, \tilde{s}_k) := 0$ ,  $j = 0, \dots, \tilde{\alpha}$ ,  $k = 0, \dots, \tilde{\beta}$ .
- For  $i = 1, \dots, m$  set  $v(q_{\ell}(\ell_i), q_w(w_i)) := \max\{c_i, v(q_{\ell}(\ell_i), q_w(w_i))\}$ ,

where  $q_{\ell}(x) := \min\{r \in S^{red}(\ell, L) : r \geq x\}$  defines the minimal length not smaller than  $x$  in  $L$ -direction and  $q_w(x) := \min\{s \in S^{red}(w, W) : s \geq x\}$  that in  $W$ -direction.

Note that we can also use a recursion similar to (6.3).

$$\begin{aligned}
 v(\tilde{r}_j, \tilde{s}_k) := & \max \{ v(\tilde{r}_{j-1}, \tilde{s}_k), v(\tilde{r}_j, \tilde{s}_{k-1}), \\
 & \max_{r \in S^{red}(\ell, L)} \{ v(r, \tilde{s}_k) + v(p_\ell^{red}(\tilde{r}_j - r), \tilde{s}_k) : r \leq \frac{\tilde{r}_j}{2}, r + p_\ell^{red}(\tilde{r}_j - r) > \tilde{r}_{j-1} \}, \\
 & \max_{s \in S^{red}(w, W)} \{ v(\tilde{r}_j, s) + v(\tilde{r}_j, p_w^{red}(\tilde{s}_k - s)) : s \leq \frac{\tilde{s}_k}{2}, s + p_w^{red}(\tilde{s}_k - s) > \tilde{s}_{k-1} \}, \\
 & j = 1, \dots, \tilde{\alpha}, \quad k = 1, \dots, \tilde{\beta}.
 \end{aligned} \tag{6.5}$$

*Example 6.2* We consider once more the cutting problem of Example 6.1. According to the concept of reduced sets of potential allocation points, we obtain  $S^{red}(\ell, L)$ ,  $S^{red}(w, W)$ , and the related optimal values of recursion (6.4) as given in Table 6.3. The assigned indices indicate that term in recursion (6.4) which determines that value. For example, since  $v(155, 103)$  has index 2, the value 58 is obtained by a horizontal cut:  $v(155, 103) = v(71, 103) + v(81, 103)$  whereas  $v(71, 103)$  results applying a vertical cut.

In this example, the decrease of the number of potential allocation points from  $\alpha = 13$  to  $\tilde{\alpha} = 7$  and from  $\beta = 10$  to  $\tilde{\beta} = 6$  involves a reduction of the number of optimal values, which have to be computed, from  $\alpha \cdot \beta = 130$  to  $\tilde{\alpha} \cdot \tilde{\beta} = 42$ . Note that an optimal pattern for  $\tilde{L} = 160$ ,  $\tilde{W} = 100$  cannot be obtained from Table 6.3. ■

The efficiency of the proposed reduction of the number of potential allocation points is strongly dependent on the input data. Its application has to be decided from case to case. In the worst case, recursion (6.1) requires  $O(L^2W^2)$  time, recursions (6.2) and (6.3) run in  $O(\alpha^2\beta^2)$  time whereas recursion (6.4) needs  $O(\tilde{\alpha}^2\tilde{\beta}^2)$  time.

However, it is to note that the cardinality of the (reduced) sets of potential allocation points does not increase if the unit of measure is changed, for instance from cm to mm.

**Table 6.3** Calculations according to recursion (6.4)

| $\tilde{r}_j \setminus \tilde{s}_k$ | 31              | 41              | 51              | 62              | 72              | 103             |
|-------------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 31                                  | 0 <sub>1</sub>  | 5 <sub>1</sub>  | 5 <sub>1</sub>  | 5 <sub>1</sub>  | 5 <sub>1</sub>  | 10 <sub>3</sub> |
| 62                                  | 0 <sub>1</sub>  | 10 <sub>2</sub> | 10 <sub>2</sub> | 10 <sub>2</sub> | 10 <sub>2</sub> | 20 <sub>2</sub> |
| 71                                  | 8 <sub>1</sub>  | 10 <sub>2</sub> | 10 <sub>2</sub> | 16 <sub>3</sub> | 18 <sub>3</sub> | 26 <sub>3</sub> |
| 81                                  | 8 <sub>1</sub>  | 10 <sub>2</sub> | 16 <sub>1</sub> | 16 <sub>3</sub> | 18 <sub>3</sub> | 32 <sub>3</sub> |
| 93                                  | 8 <sub>1</sub>  | 15 <sub>2</sub> | 16 <sub>1</sub> | 16 <sub>3</sub> | 23 <sub>3</sub> | 32 <sub>3</sub> |
| 124                                 | 8 <sub>1</sub>  | 20 <sub>2</sub> | 20 <sub>2</sub> | 20 <sub>2</sub> | 28 <sub>3</sub> | 42 <sub>2</sub> |
| 155                                 | 16 <sub>2</sub> | 25 <sub>2</sub> | 26 <sub>2</sub> | 32 <sub>2</sub> | 41 <sub>3</sub> | 58 <sub>2</sub> |

### 6.3 Two-Stage Guillotine Cutting

Here we consider the same cutting problem as in the previous section, but with the limitation that a 2-stage guillotine cutting pattern (for short, 2-stage pattern) has to be obtained. Rotation of pieces is again not permitted. However, an extension to that, more general case is straightforward.

In case of *2-stage (guillotine) cutting* either the rectangular raw material sheet is cut into (horizontal) strips using horizontal G-cuts (*first stage*) which can further be dissected by vertical G-cuts (*second stage*) into desired products or waste, or the raw material is cut into vertical strips using vertical G-cuts (first stage) which are dissected by horizontal G-cuts (second stage) into desired products or waste. In the following, we restrict ourselves, if not stated otherwise, to the horizontal variant.

When dealing with the 2-stage guillotine cutting problem further two cases are differentiated: *non-exact case* and *exact case*, [7]. For the exact case it is demanded that the width of all items obtained from a (horizontal) strip equals the strip width. That means, subsequent trimming is not allowed (see Fig. 6.4, left).

In the sequel we assume  $w_1 \leq w_2 \leq \dots \leq w_m$  and define the *different* piece widths  $\bar{w}_j, j = 1, \dots, \bar{m}$ , with  $\bar{w}_1 = w_1 < \dots < \bar{w}_{\bar{m}} = w_m$  and corresponding index sets

$$I_j = \{i \in I : w_i = \bar{w}_j\}$$

for  $j = 1, \dots, \bar{m}$ . Hence,  $I = \cup_{j=1}^{\bar{m}} I_j$  and  $I_j \neq \emptyset$  for  $j = 1, \dots, \bar{m}$ .

In order to compute an optimal 2-stage pattern with horizontal G-cuts in the first stage, we have to find an optimal (in fact, one-dimensional) strip pattern for each different strip width  $\bar{w}_j, j = 1, \dots, \bar{m}$ . That means, we have to solve  $\bar{m}$  knapsack problems. For the non-exact case these are the knapsack problems

$$F_j(L) := \max \left\{ \sum_{k=1}^j \sum_{i \in I_k} c_i x_i : \sum_{k=1}^j \sum_{i \in I_k} \ell_i x_i \leq L, x_i \in \mathbb{Z}_+, i \in \bigcup_{k=1}^j I_k \right\}, \quad j = 1, \dots, \bar{m}, \quad (6.6)$$

and for the exact case

$$F_j(L) := \max \left\{ \sum_{i \in I_j} c_i x_i : \sum_{i \in I_j} \ell_i x_i \leq L, x_i \in \mathbb{Z}_+, i \in I_j \right\}, \quad j = 1, \dots, \bar{m}. \quad (6.7)$$

|   |   |   |   |
|---|---|---|---|
| 3 | 3 | 4 |   |
| 2 | 2 | 2 | 2 |
| 1 | 1 | 1 |   |

|   |   |  |   |   |
|---|---|--|---|---|
| 3 | 3 |  | 6 | 6 |
| 2 | 2 |  | 5 | 5 |
| 1 | 1 |  | 1 |   |

**Fig. 6.4** Exact and non-exact 2-stage guillotine cutting patterns with horizontal G-cuts in the first stage

Let  $y_j, j = 1, \dots, \bar{m}$ , denote the frequency a strip with width  $\bar{w}_j$  (and the related strip pattern) is used in the two-dimensional pattern. Clearly, an optimal combination of the horizontal strip patterns can be achieved by solving another knapsack problem:

$$G(W) := \max \left\{ \sum_{j=1}^{\bar{m}} F_j(L)y_j : \sum_{j=1}^{\bar{m}} \bar{w}_j y_j \leq W, y_j \in \mathbb{Z}_+, j \in \{1, \dots, \bar{m}\} \right\}.$$

Note that we can apply a recursion similar to the algorithm of Gilmore and Gomory (see Sect. 2.2) to calculate the  $\bar{m}$  values  $F_j(L)$  in the non-exact case. Then the computational effort is proportional to  $O(mL)$  which is the effort to solve a single knapsack problem with  $m$  variables and knapsack capacity  $L$ .

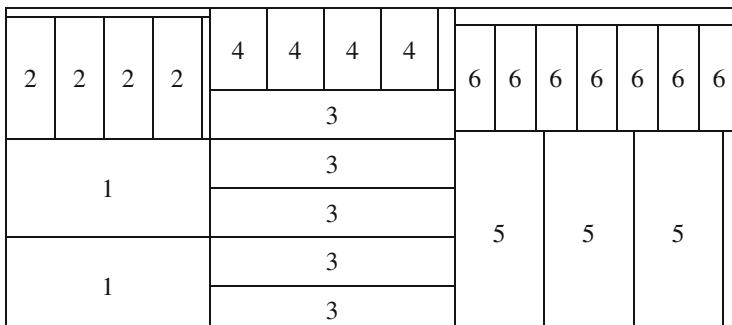
However, in the exact case the  $\bar{m}$  knapsack problems are independent of each other so that a b&b algorithm can be applied alternatively. A gain, the total computational complexity can be estimated by  $O(mL)$  if DP-based algorithms are used.

Altogether, we need to solve two knapsack problems in order to compute an optimal 2-stage pattern with horizontal G-cuts in the first stage. Consequently, four knapsack problems have to be solved to obtain an optimal 2-stage pattern (regarding both directions in the first stage).

## 6.4 Three-Stage Guillotine Cutting

In Figs. 6.5 and 6.6, 3-stage guillotine patterns with vertical G-cuts in the first stage are drawn (exact and non-exact case). The G-cuts of the second stage are horizontal followed by vertical G-cuts in the third stage. Horizontal G-cuts in the first stage as well as the exact and the non-exact case are considered, too.

We say, a rectangular part of  $L \times W$ , resulting from a G-cut of the first stage, forms a *segment*. In a 3-stage pattern with vertical G-cuts in the first stage (Fig. 6.5)



**Fig. 6.5** A 3-stage guillotine cutting pattern, exact case, vertical G-cuts in the first stage

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   |   |   |   | 4 | 4 | 4 | 4 |   | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 2 | 2 | 8 |   |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   | 1 |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   | 1 |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   | 3 |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   | 5 |   |   | 5 |   |   | 7 |

**Fig. 6.6** A 3-stage guillotine cutting pattern, non-exact case

segments of type  $r \times W$  are obtained, where  $r \in \widetilde{S}$  defines the lengths of the segment and  $\widetilde{S}$  is an appropriate set of segment-lengths, e.g. equal to  $S(\ell, L)$  or  $S^{red}(\ell, L)$ . To obtain an optimal 3-stage pattern (with vertical cuts in the first stage) we need to combine optimal 2-stage patterns (with horizontal G-cuts in the first stage) of the segments. To this end, we can choose the following approach.

Without loss of generality, we assume again that the piece types are numbered according to  $w_1 \leq w_2 \leq \dots \leq w_m$  and consider the different widths of piece types  $\overline{w}_j, j \in \overline{J} := \{1, \dots, \overline{m}\}$ , with  $w_1 = \overline{w}_1 < \dots < \overline{w}_{\overline{m}} = w_m$  together with corresponding index sets  $I_j = \{i \in I : w_i = \overline{w}_j\}$  for  $j = 1, \dots, \overline{m}$ . Again, we have  $I = \cup_{j=1}^{\overline{m}} I_j$  and  $I_j \neq \emptyset$  for  $j \in \overline{J}$ .

To compute an optimal 2-stage pattern of segment  $r \times W$  ( $r \in \widetilde{S}$ ) with horizontal G-cuts in the first stage, we solve a knapsack problem with capacity  $r$  for each  $\overline{w}_j, j = 1, \dots, \overline{m}$ , analogously to (6.6) or (6.7). In this way, we obtain optimal strip patterns of all rectangles  $r \times \overline{w}_j$ . In the non-exact case, these are the knapsack problems

$$F_j(r) := \max \left\{ \sum_{k=1}^j \sum_{i \in I_k} c_i x_i : \sum_{k=1}^j \sum_{i \in I_k} \ell_i x_i \leq r, x_i \in \mathbb{Z}_+, i \in \bigcup_{k=1}^j I_k \right\}, \quad r \in \widetilde{S}, j \in \overline{J}, \quad (6.8)$$

and in the exact case

$$F_j(r) := \max \left\{ \sum_{i \in I_j} c_i x_i : \sum_{i \in I_j} \ell_i x_i \leq r, x_i \in \mathbb{Z}_+ i \in I_j \right\}, \quad r \in \widetilde{S}, j \in \overline{J}. \quad (6.9)$$

Variable  $x_i$  indicates the frequency of type  $i$  items within the strip. For a fixed  $r$  ( $r \in \widetilde{S}$ ), we can optimally combine the strip patterns for  $r \times \overline{w}_j, j \in \overline{J}$ , by solving another knapsack problem, i.e., we obtain the value of an optimal 2-stage pattern of

segment  $r \times W$  according to

$$G(r, W) := \max \left\{ \sum_{j=1}^{\bar{m}} F_j(r)y_j : \sum_{j=1}^{\bar{m}} \bar{w}_j y_j \leq W, \quad y_j \in \mathbb{Z}_+, \quad j = 1, \dots, \bar{m} \right\} \quad (6.10)$$

where variable  $y_j$  denotes the number how often an optimal strip pattern of  $r \times \bar{w}_j$  is used. It is easy to see that the optimal value function  $G(\cdot, W) : [0, L] \rightarrow \mathbb{R}$  is piecewise constant and monotone non-decreasing. Let  $K$  denote an appropriate index set of  $\bar{S} = \{r_k : k \in K\}$ , the set of potential jump discontinuities of  $G(\cdot, W)$ . Defining  $z_k$  as the frequency the 2-stage pattern of segment  $r_k \times W$  occurs within the 3-stage pattern, we obtain the value of an optimal one by solving the knapsack problem

$$H(L) := \max \left\{ \sum_{k \in K} G(r_k, W) z_k : \sum_{k \in K} r_k z_k \leq L, \quad z_k \in \mathbb{Z}_+, \quad k \in K \right\}. \quad (6.11)$$

Hence,  $H(L)$  is the value of an optimal 3-stage pattern with vertical G-cuts in the first stage. A corresponding pattern can be identified in a similar way as for 2-stage patterns.

To compute all values  $F_j(r)$  for all  $r$  a recursion in analogy to the Gilmore/Gomory algorithm or the Longest Path Method (see Sects. 2.2 and 2.3) can be applied. Because of  $r \leq L$ , the related computational effort can be estimated by  $O(mL)$ .

The application of the concept of potential allocation points can again lead to a remarkable decrease of the total computational expense. Therefore, we identify a subset of the set of potential allocation points  $S(\ell, L) = \{r_0, \dots, r_\alpha\}$  with  $r_0 = 0 < r_1 < \dots < r_\alpha \leq L$  as follows: let

$$J_p := \{j \in \{1, \dots, \bar{m}\} : F_j(r_p) > \max\{F_j(r_{p-1}), F_{j-1}(r_p)\}\}, \quad p = 1, \dots, \alpha,$$

where  $F_0(r_p) := 0$  for all  $p$ . If  $|J_p| = 0$  for some  $p$ , then there does not exist any one-dimensional strip pattern for rectangle  $r_p \times \bar{w}_j$  ( $j \in \{1, \dots, \bar{m}\}$ ) with value larger than  $F_j(r_{p-1})$ . Thus,  $G(r_p, W) = G(r_{p-1}, W)$  holds. Consequently, there is no need to solve the knapsack problem for  $G(r_p, W)$ . If  $F_j(r_p) = F_{j-1}(r_p)$ , then, because of dominance considerations, we can set  $y_j := 0$ .

Otherwise, if  $|J_p| > 0$ , then  $G(r_p, W)$  has to be computed. Hereby, due to (6.10), the already obtained values  $G(r_{p-1}, s)$  can be used. Let

$$G_p(s) := \max \left\{ \sum_{j \in J_p} F_j(r_p) y_j : \sum_{j \in J_p} \bar{w}_j y_j \leq s, \quad y_j \in \mathbb{Z}_+, \quad j \in J_p \right\}, \quad s \leq W,$$

denote the value of an optimal 2-stage pattern with horizontal G-cuts in the first stage for rectangle  $r_p \times s$  which only contains one-dimensional strip patterns of

**Table 6.4** Input data of Example 6.3

| $i$      | 1  | 2  | 3  | 4  |
|----------|----|----|----|----|
| $\ell_i$ | 30 | 12 | 40 | 14 |
| $w_i$    | 20 | 25 | 30 | 50 |
| $c_i$    | 6  | 3  | 12 | 7  |

length  $r_p$ , then we have

$$G(r_p, W) = \max_{s \leq W} \{G(r_{p-1}, s) + G_p(W - s)\}.$$

According to that maximization at most all potential allocation points in  $S(\bar{w}, W) = S(w, W)$  have to be considered. However, function  $G_p$  has fewer jump discontinuities, in general, for instance in case of  $|J_p| = 1$ .

The computation of an optimal 3-stage pattern with horizontal G-cuts in the first stage can be performed in a similar way. The total expense for that can be estimated by  $O(nLW)$ . Since, in general, the total effort for computing an optimal 3-stage pattern is considerably larger than that for optimal 2-stage patterns, frequently only a restricted subset of 3-stage patterns is considered in applications. That could be the restriction on the exact case within the one-dimensional strip patterns, i.e., defining  $F_j(r)$  according to (6.9). Another simplification results if only homogeneous strip patterns are used (as in Fig. 6.5) according to

$$F_j(r) := \max \left\{ \max_{i \in I_j} c_i \lfloor r / \ell_i \rfloor \right\}, \quad r \in \widetilde{S}, j = 1, \dots, \bar{m}.$$

Limitations on the position and number of G-cuts of the first stage are also applied as, for example, in 3-block patterns considered in the subsequent section. Sometimes,  $\widetilde{S} := \{\ell_i : i \in I\}$  is used together with demanding that at least an item of length  $\ell_i$  is obtained if a segment of length  $\ell_i$  is cut. Mostly, the necessity of such limitations results from technological circumstances.

*Example 6.3* Let a rectangle with length  $L = 100$  and width  $W = 80$  as well as four piece types be given, see Table 6.4. The task is to compute an optimal 3-stage pattern with vertical G-cuts in the first stage for the non-exact case. For comparison, we consider  $\widetilde{S} = S(\ell, L)$  as well as  $\widetilde{S} = S^{red}(\ell, L)$ .

First of all, we compute the values  $F_j(r)$  for all  $r \in S(\ell, L)$  and all  $j \in \{1, \dots, 4\}$  according to the definition in formula (6.8). These values are shown in Table 6.5. In the last row of the table the values  $G(r, W)$ , which are obtained according to formula (6.10), are also listed. Only the enlarged values are given.

Finally, we obtain the optimal value  $H(L) = 77$  by formula (6.11). Corresponding optimal patterns are drawn in Fig. 6.7. Note that the allocation of an additional piece of type 2 in the left pattern would lead to a 4-stage pattern and not to a 3-stage one.

**Table 6.5** Recursions (6.8) and (6.10) with  $\tilde{S} = S(\ell, L)$ 

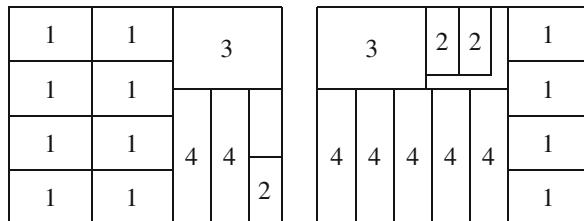
| $j \setminus r$ | 0 | 12 | 14 | 24 | 26 | 28 | 30 | 36 | 38 | 40 | 42 | 44 | 48 | 50 | 52 | 54 | 56 | 58 | 60 |
|-----------------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $F_1(r)$        | 0 |    |    |    |    | 6  |    |    |    |    |    |    |    |    |    |    |    |    | 12 |
| $F_2(r)$        |   | 3  |    | 6  |    |    | 9  |    |    |    |    |    | 12 |    |    |    |    |    | 15 |
| $F_3(r)$        |   |    |    |    |    |    |    |    | 12 |    |    |    |    |    |    | 15 |    |    |    |
| $F_4(r)$        |   |    | 7  |    | 10 | 14 |    |    |    | 17 | 21 |    |    |    |    | 24 | 28 |    |    |
| $G(r, W)$       | 0 | 9  | 10 | 18 |    | 20 | 24 | 27 |    | 29 | 33 |    | 36 |    |    | 39 | 43 |    | 48 |

| $j \setminus r$ | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| $F_1(r)$        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 18 |    |    |    |     |
| $F_2(r)$        |    |    |    |    |    | 18 |    |    |    |    |    | 21 |    |    |    |    |    | 24 |    |     |
| $F_3(r)$        |    | 18 |    |    |    |    | 21 |    | 24 |    |    |    |    |    |    | 27 |    |    |    |     |
| $F_4(r)$        |    |    |    | 31 | 35 |    |    |    |    |    | 38 | 42 |    |    |    |    | 45 | 49 |    |     |
| $G(r, W)$       |    |    |    | 49 | 53 | 54 |    | 57 |    | 60 | 62 | 66 |    |    | 72 |    | 75 | 76 |    |     |

**Table 6.6** Recursions (6.8) and (6.10) with  $\tilde{S} = S^{red}(\ell, L)$ 

| $j \setminus r$ | 0 | 12 | 14 | 24 | 26 | 28 | 30 | 36 | 38 | 40 | 42 | 44 | 48 | 50 | 52 | 56 | 58 | 60 | 62 | 64 | 70 | 72 | 74 | 76 | 86 | 88 | 100 |
|-----------------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| $F_1(r)$        | 0 |    |    |    |    | 6  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 18 |     |
| $F_2(r)$        |   | 3  |    | 6  |    |    | 9  |    |    |    | 12 |    |    |    |    | 15 |    |    |    |    | 18 |    |    | 21 | 24 |    |     |
| $F_3(r)$        |   |    |    |    |    |    |    | 12 |    |    |    |    |    | 15 |    |    |    |    |    | 18 |    |    | 21 | 24 | 27 |    |     |
| $F_4(r)$        |   |    | 7  |    | 10 | 14 |    |    |    | 17 | 21 |    |    |    |    | 28 |    |    |    |    | 35 |    |    | 42 | 49 |    |     |
| $G(r, W)$       | 0 | 9  | 10 | 18 |    | 20 | 24 | 27 |    | 29 | 33 |    | 36 |    |    | 43 | 48 |    |    |    | 53 | 54 |    | 57 | 66 |    | 76  |

**Fig. 6.7** Optimal 3-stage patterns of Example 6.3

In this example it becomes evident that the computation of several  $G$ -values can be avoided. Moreover, a limitation of cut-positions of the first stage, e.g. to  $\{\ell_i : i \in I\}$  or to  $r \in S(\ell, L)$  with  $r \leq \max_{i \in I} \ell_i$ , leads to a remarkable saving of computational effort. Since  $S^{red}(\ell, L) \subseteq S(\ell, L)$  holds, setting  $\tilde{S} = S^{red}(\ell, L)$  can save computational effort as Table 6.6 demonstrates. ■

As the example also shows, when restricting  $\tilde{S}$  to  $\tilde{S} = \{\ell_i : i \in I\}$ , we loose an optimal 3-stage pattern (Fig. 6.7, right), but another (Fig. 6.7, left) remains so that the optimal value is not decreased (in the example).

## 6.5 Further Pattern Types

In various applications specific pattern types are used caused by technological restrictions of the cutting machines. For instance, when cutting (approximately rectangular) granite sheets into rectangular pieces the concept of *1-group cutting* often is applied (see Fig. 6.8). In a single *group* all G-cuts are continuous from one side to the opposite side. Note that up to two trimming G-cuts are possibly permitted. So far, there are not known any applicable recurrence formula to compute an optimal 1-group cutting pattern. Instead, b&b algorithms have to be performed.

In the furniture industry where large rectangular sheets (e.g. of size 6m×4m) are cut in smaller rectangular pieces, frequently so-called *2-block* and *3-block patterns* are considered which are special 2- and 3-group patterns. Each *block pattern* is a homogeneous one. Figure 6.9 shows two example patterns. Recurrence formulas are known for these schemata.

For simplicity, we consider non-rotatable pieces. Let  $v(r, s)$  denote the maximal value of homogeneous patterns for rectangle  $r \times s$ . Then, obviously,

$$v(r, s) = \max_{i \in I} \{c_i \lfloor r/\ell_i \rfloor \lfloor s/w_i \rfloor\}$$

holds. Hence, the maximal value  $v_{2bv}(L, W)$  for 2-block patterns with vertical dissection of  $L \times W$  into  $r \times W$  and  $(L - r) \times W$  can be obtained according to

$$v_{2bv}(L, W) = \max_{0 \leq r \leq L/2} \{v(r, W) + v(L - r, W)\}.$$

Moreover, for the optimal value  $v_{3bv}(L, W)$  of 3-block cutting with initial vertical dissection of  $L \times W$  and subsequent horizontal dissection of  $(L - r) \times W$ , we have

$$v_{3bv}(L, W) = \max_{0 \leq r \leq L} \left\{ v(r, W) + \max_{0 \leq s \leq W/2} \{v(L - r, s) + v(L - r, W - s)\} \right\}.$$

|   |   |  |   |   |
|---|---|--|---|---|
| 3 | 4 |  | 5 | 5 |
| 1 | 1 |  | 2 | 2 |
| 1 | 1 |  | 2 | 2 |

Fig. 6.8 1-group cutting pattern

|   |   |   |  |   |  |
|---|---|---|--|---|--|
| 1 | 1 | 2 |  | 2 |  |
| 1 | 1 |   |  |   |  |
| 1 | 1 |   |  |   |  |
| 1 | 1 |   |  |   |  |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 |
| 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 |
| 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |

Fig. 6.9 2- and 3-block patterns

If rotation of pieces is permitted, then the optimal value  $v(r, s)$  of homogeneous patterns for rectangle  $r \times s$  is defined by

$$v(r, s) = \max_{i \in I} \{c_i \cdot \max\{\lfloor r/\ell_i \rfloor \lfloor s/w_i \rfloor, \lfloor r/w_i \rfloor \lfloor s/\ell_i \rfloor\}\}.$$

Clearly, the concept of (reduced) sets of potential allocation points can be used in all of the above mentioned cases by an appropriate definition of the respective sets.

## 6.6 Bounded Guillotine Cutting

A *bounded guillotine cutting problem* is a guillotine cutting problem where additionally the frequency, items of type  $i$  are contained within the pattern, is limited to be at most  $u_i$  for some or all piece types  $i \in I$ .

In what follows, we describe a constructive principle to obtain (near-)optimal patterns for bounded, general and  $k$ -stage, guillotine cutting, as well as two ILP models for the 2-stage case.

### 6.6.1 The Algorithm of Wang

No efficient recurrence formulas are available for the bounded cutting problem since the number of *states* (with respect to dynamic programming) to be considered becomes much larger. In difference to the non-bounded case, for each rectangle  $r \times s$  at least all maximal patterns, together with their value and some information how it is obtained, have to be identified and saved.

A constructive principle, similar to the Longest Path Method (see Sect. 2.3), is proposed by Wang [12] for the non-stage bounded guillotine cutting problem. Appropriate reductions of the number of states (partial solutions) by means of appropriate criteria enable to derive some heuristics. The basic principle of the approach consists in evaluating partial solutions and to combine them to get new ones for larger rectangles. We represent a solution for rectangle  $r \times s$  by a 4-tuple  $(r, s, v, a)$  where  $a = (a_1, \dots, a_m)^\top \in \mathbb{Z}_+^m$ . As usual, coefficient  $a_i$  indicates how many items of type  $i$  will be obtained when  $r \times s$  is cut according to  $a$ . Moreover,  $v = c^\top a$  is the value of pattern  $a$ . To construct guillotine patterns which do not contain more than  $u_i$  items of type  $i \in I$ , we consider the vertical and horizontal combination of partial solutions. Let  $(r, s, v, a)$  and  $(r', s', v', a')$  represent two (feasible) partial solutions. Then, a *horizontal combination* of them yields the tuple  $(r + r', \max\{s, s'\}, v + v', a + a')$ , whereas a *vertical combination* leads to  $(\max\{r, r'\}, s + s', v + v', a + a')$ . The resulting 4-tuple represents a feasible pattern of the enlarged rectangle if its length and width do not exceed  $L$  and  $W$ , respectively, and if  $a + a' \leq u$  is satisfied.

**Algorithm of Wang**

Input:  $E = (m, L, W, \ell, w, c, u)$

Output: feasible pattern

- (1)  $k := 1$ ,  $\Omega_k := \emptyset$ . For  $i \in I$  set  $\Omega_k := \Omega_k \cup \{(\ell_i, w_i, c_i, e^i)\}$ .
- (2) Set  $\Omega_{k+1} := \emptyset$ . Combine every partial solution  $(r, s, v, a) \in \Omega_k$  horizontally with each solution  $(r', s', v', a') \in \bigcup_{j=1}^k \Omega_j$  and add the new 4-tuple, if feasible, to  $\Omega_{k+1}$ .
- (3) Combine every partial solution  $(r, s, v, a) \in \Omega_k$  vertically with each solution  $(r', s', v', a') \in \bigcup_{j=1}^k \Omega_j$  and add the new 4-tuple, if feasible, to  $\Omega_{k+1}$ .
- (4) Termination criterion: if  $\Omega_{k+1} = \emptyset$ , then identify a best pattern in  $\Omega_k$  – stop.
- (5) Remove feasible solutions from  $\Omega_{k+1}$  according to some predefined rule. Set  $k := k + 1$  and go to (2).

**Fig. 6.10** A schema of the Algorithm of Wang

To formulate the algorithm, let  $\Omega_k$  denote the set of feasible 4-tuples generated in the  $k$ th iteration,  $k = 1, 2, \dots$ . As initialization we use patterns which contain only a single piece, i.e., we have  $e^i \in \Omega_1$  for all  $i \in I$  where  $e^i$  is the  $i$ th unit vector of  $\mathbb{R}^m$ .

A reduction in step (5) of the Algorithm of Wang (Fig. 6.10) can be oriented, e.g., on the cardinality of  $\Omega_{k+1}$  for which an upper limit can be predefined. Only those partial solutions, generated in the last iteration, are maintained which have (relative) best values. An example for such rule is considered in Exercise 6.5.

It is quite obvious that the algorithm of Wang allows some modifications. For instance, one can also combine two partial solutions  $a^1$  and  $a^2$  with  $a^1 + a^2 \not\leq u$ . Setting  $a_i := \min\{u_i, [a^1]_i + [a^2]_i\}$  leads to a feasible solution if the size restrictions are met. In this way, a larger amount of combinations can be inspected.

### 6.6.2 ILP Models for Bounded 2-Stage Cutting

A straightforward application of ILP models of the 2OPP seems to be difficult because of the guillotine condition and the a priori not known maximal stage-number in an optimal pattern. Therefore, the following considerations are restricted to bounded 2-stage cutting. Without loss of generality, we only consider the case of horizontal G-cuts in the first stage.

Let  $E = (m, L, W, \ell, w, c, u)$  be an instance of the bounded 2-stage guillotine cutting problem. For simplicity of description, rotation of piece types is not permitted. We again assume

$$w_1 \geq w_2 \geq \dots \geq w_m \quad \text{and} \quad \ell_i \neq \ell_j \text{ if } w_i = w_j.$$

Since at most  $u_i$  items of type  $i \in I$  can be obtained, altogether  $n := \sum_{i \in I} u_i$  items belong to the instance. To represent the  $n$  items, let

$$j_0 := 0, \quad j_i := j_{i-1} + u_i, \quad i = 1, \dots, m, \quad J := \{1, \dots, n\}.$$

Hence, all items  $k \in \{j_{i-1} + 1, \dots, j_i\}$  are of the same type. Moreover, for  $k = 1, \dots, n$ , let

$$i(k) := \min\{i \in I : k \leq j_i\}$$

indicate the type of item  $k$ . Thus, we have  $w_{i(k)} \geq w_{i(k+1)}$  for  $k = 1, \dots, n-1$ .

According to [8], for each of the  $n = j_m$  desired items a pattern  $a^k = (a_{1k}, \dots, a_{nk})^\top \in \mathbb{B}^n$  is defined for strip  $L \times w_{i(k)}$  as follows:

$$a_{jk} = \begin{cases} 1, & \text{if item } j \in J \text{ is cut from strip } k, \\ 0, & \text{otherwise,} \end{cases} \quad j \in J, k \in J.$$

Since the item widths are monotone non-increasing and to avoid a lot of symmetric solutions, we set

$$a_{jk} := 0 \quad \text{for all } j < k$$

and demand  $a_{kk} = 1$  if strip  $k$  is cut. The latter condition ensures that the width of strip  $k$  is determined by item  $k \in J$  and equal to  $w_{i(k)}$ . Now, we are ready to formulate a first model.

### Model 1 for bounded 2-stage cutting

$$z^{b2s} = \max \sum_{j \in J} c_{i(j)} \sum_{k=1}^j a_{jk} \quad \text{s.t.} \quad (6.12a)$$

$$\sum_{k=1}^j a_{jk} \leq 1, \quad j \in J, \quad (6.12b)$$

$$\sum_{j=k+1}^n \ell_{i(j)} a_{jk} \leq (L - \ell_{i(k)}) a_{kk}, \quad k = 1, \dots, n-1, \quad (6.12c)$$

$$\sum_{k \in J} w_{i(k)} a_{kk} \leq W, \quad (6.12d)$$

$$a_{jk} \in \{0, 1\}, \quad j = k, \dots, n, k \in J.$$

Clearly, the total number  $n$  of items, and hence the number  $n(n+1)/2$  of binary variables, can rise rapidly if the bounds  $u_i$  are larger, for instance, near to  $\lfloor L/\ell_i \rfloor \cdot \lfloor W/w_i \rfloor$ .

To further reduce the solution space by discarding symmetric solutions, one can use additional inequalities as, for example,

$$a_{jj} \geq a_{j+1,j+1} \quad \text{for } j = j_{i-1} + 1, \dots, j_i - 1, \quad i \in I,$$

which guarantee that the ‘first’ strip belonging to type  $i \in I$  is used if any. Moreover, we can further exclude solutions without changing the optimal value by demanding

$$\sum_{s=t+1}^{j_i} a_{st} \geq \sum_{s=t+2}^{j_i} a_{s,t+1}, \quad t = j_{i-1} + 1, \dots, j_i - 1, \quad i \in I,$$

which means that in a preceding strip at least as many items of type  $i$  are used as in the subsequent one if both strips belong to the same type. Not all symmetric solutions are removed in this way since a complete lexicographic sorting is still not achieved.

In a second ILP model a more compact formulation is reached by using  $m$ -dimensional nonnegative integer vectors for the strip patterns and binary variables which model whether a strip is contained in the solution. For  $i \in I$  and  $k \in J$ , let

$$a_{ik} := \begin{cases} \text{number of items of type } i \text{ cut from strip } k & \text{if } i \neq i(k), \\ \text{number of \textbf{additional} items of type } i \text{ cut from strip } k \text{ if } i = i(k). \end{cases}$$

The term ‘additional’ indicates that the item of type  $i$ , initializing strip  $k$ , is separately considered (if the strip corresponds to this type of items). We define binary variables  $y_k$  as follows:

$$y_k := \begin{cases} 1, \text{ if strip } k \text{ occurs in the pattern,} \\ 0, \text{ otherwise,} \end{cases} \quad k \in J.$$

Then a second model can be stated:

### Model 2 for bounded 2-stage cutting

$$z^{b2s} = \max \sum_{i \in I} c_i \left( \sum_{k=1}^{j_i} a_{ik} + \sum_{k=j_{i-1}+1}^{j_i} y_k \right) \quad \text{s.t.} \quad (6.13a)$$

$$\sum_{k=1}^{j_i} a_{ik} + \sum_{k=j_{i-1}+1}^{j_i} y_k \leq u_i, \quad i \in I, \quad (6.13b)$$

$$\sum_{i=i(k)}^m \ell_i a_{ik} \leq (L - \ell_{i(k)}) y_k, \quad k \in J, \quad (6.13c)$$

$$\sum_{k \in J} w_{i(k)} y_k \leq W, \quad (6.13d)$$

$$\sum_{s=k}^{j_i} a_{is} \leq u_i - (k - j_{i-1}), \quad k = j_{i-1} + 1, \dots, j_i - 1, \quad i \in I, \quad (6.13e)$$

$$a_{ik} \in \mathbb{Z}_+, \quad i \in I, \quad k = 1, \dots, j_i, \quad y_k \in \mathbb{B}, \quad k \in J.$$

Note that the bounding value in (6.13e) regards the definition of  $a_{is}$  as number of additional items of type  $i$  available for strip  $s \in \{k, \dots, j_i\}$  and  $k \in \{j_{i-1} + 1, \dots, j_i - 1\}$ .

To reduce the solution space of Model 2 by discarding symmetric solutions additional inequalities similar to Model 1 can be used. In [8] the equivalence of both models with respect to their LP relaxations is shown, and the efficiency of symmetry-reducing inequalities is numerically proved. The rotatable case is also considered there.

## 6.7 Guillotine Cutting Stock Problems

Two-dimensional *Guillotine Cutting Stock Problems* (2CSP) occur, for instance, in furniture industry where larger numbers of rectangular items of relatively few different types have to be cut from large raw material sheets. When cutting foamed plastic a three-dimensional version of the CSP can be meaningful.

### 6.7.1 Column Generation Approach

In all the cases where guillotine cuts have to be applied and for which an efficient solution method is known to compute an optimal pattern with respect to given profit coefficients, the procedure to approximately solve the 1CSP can be transferred to the higher-dimensional case. In the column generation approach (see Sect. 4.3.3) a sequence of slave problems has to be solved. The effort to solve a slave problem, i.e., to compute an optimal pattern with respect to the current simplex multipliers, essentially determines the total efficiency of the whole solution process.

In particular, since optimal 2- or 3-stage patterns can be obtained by pseudo-polynomial expense, the related LP relaxation of the 2CSP can be solved efficiently. Appropriate rounding procedures are similar to those mentioned for the one-dimensional case.

There are also known ILP models for the 2- and 3-stage two-dimensional guillotine bin packing problem (Chap. 8) which are directly applicable to cutting stock problems.

### 6.7.2 One-Cut ILP Model of the 2- and 3-Stage 2CSP

The following approach is proposed by Silva, Alvelos, and de Carvalho [11].

Let an instance  $E = (m, L, W, \ell, w, d)$  of the 2CSP be given, and let  $I = \{1, \dots, m\}$ . We aim to minimize the number of rectangles  $L \times W$  needed to cut  $d_i$  pieces of type  $\ell_i \times w_i$ ,  $i \in I$ . To this end, 2- or 3-stage guillotine cutting can be applied (exact or non-exact case) regarding the following limitations:

1. The G-cuts of the first stage have to be horizontal (producing *strips*).
2. Also in the 3-stage case, the height of a strip is determined by a piece height.

Similar to the one-cut model of Dyckhoff [4] for the one-dimensional case, all intermediate rectangles, called *residual rectangles*, resulting by a horizontal and/or vertical G-cut, are considered for further cutting, if meaningful.

In case of 2-stage cutting, residual rectangles of size  $L \times w$  (resulting by horizontal G-cuts) with  $w = W - \sum_{i \in I} w_i a_i$  for some  $a_i \in \mathbb{Z}_+$ , and of form  $l \times w_i$  (when dissecting a strip of widths  $w_i$ ) with  $l = L - \sum_{i \in I} \ell_i a_i$  and  $a_i \in \mathbb{Z}_+$  can be obtained.

In case of 3-stage cutting, further residual rectangles can be produced, namely of form  $\ell_i \times w'$  with  $w' = w_j - \sum_{k \in I} w_k a_k$  and  $a_k \in \mathbb{Z}_+$ .

Let  $J = \{0, 1, \dots, n\}$  denote an index set of all (further usable) residual rectangles  $R_j$  where  $R_0 = L \times W$ . Then, variables  $x_{ij} \in \mathbb{Z}_+$ ,  $i \in I$ ,  $j \in J$ , are defined where  $x_{ij} > 0$  indicates how many rectangles  $R_j$  are cut into a piece  $i$  and up to two (smaller) residual rectangles, or waste (cf. Fig. 6.11). In particular, a resulting rectangle of size  $\tilde{l} \times \tilde{w}$  is waste if  $\tilde{l} < \min\{\ell_i : i \in I, w_i \leq \tilde{w}\}$  or  $\tilde{w} < \min\{w_i : i \in I, \ell_i \leq \tilde{l}\}$ . Some more can be identified to be waste in case of 2-stage cutting or the exact case demand.

For the sake of an easier description, we introduce coefficients  $a_{ijk} \in \mathbb{B}$  where  $a_{ijk} = 1$  if and only if residual rectangle  $R_k$  results from  $R_j$  if piece  $i$  is cut.

#### One-Cut Model of the 2- and 3-stage guillotine CSP

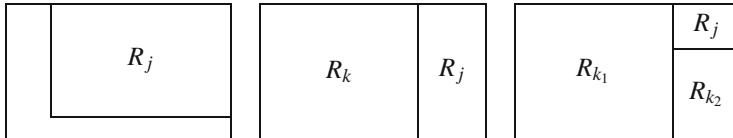
$$z^{OC2} = \min \sum_{i \in I} x_{i0} \quad \text{s.t.} \tag{6.14a}$$

$$\sum_{j \in J} x_{ij} \geq d_i, \quad i \in I, \tag{6.14b}$$

$$\sum_{j \in J} \sum_{i \in I} a_{ijk} x_{ij} \geq \sum_{i \in I} x_{ik}, \quad k \in J, \tag{6.14c}$$

$$x_{ij} \in \mathbb{Z}_+, \quad i \in I, j \in J.$$

It is easy to see, this basic model can be improved by a more sophisticated definition of variables since it contains, obviously, variables which have value 0. For instance, if  $R_j$  has size  $l' \times w'$ , then  $x_{ij} = 0$  follows for all pieces  $i$  with  $w_i > w'$  or  $\ell_i > l'$ . The



**Fig. 6.11** To the one-cut model of 2- and 3-stage guillotine cutting

application of the concept of reduced sets of potential allocation points can lead to a smaller number of variables, too. An adaptation of the model to compute an optimal 2- or 3-stage guillotine pattern is even possible without changing the number of variables.

### 6.7.3 Extensions

The 2-stage guillotine CSP with multiple types of stock rectangles are addressed by Cintra et al. [2] and Furini et al. [5] where a column generation heuristic is presented. First investigations concerning the IRUP and MIRUP are presented in [10]. In [6] lower bounds for one-, two-, and three-dimensional online bin packing algorithms are proposed.

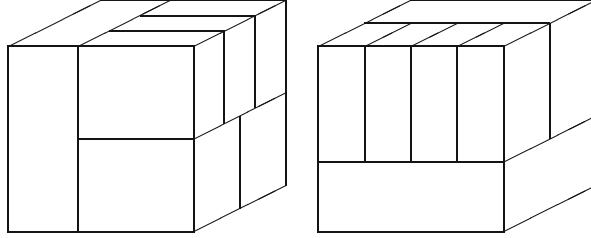
The 3-stage (guillotine) bin packing problem (3sBPP) is modeled as an ILP problem in [9], too. At first, a *restricted* 3-stage BPP is considered: in case of horizontal G-cuts in the first stage the widths  $\tilde{w}$  of each resulting sub-rectangle (residual rectangle, segment)  $L \times \tilde{w}$  are determined by an item width, i.e.,  $\tilde{w} \in \{w_1, \dots, w_n\}$ . This basic model involves  $O(n^2)$  binary variables and  $O(n)$  restrictions. Based on it, a model for the general case is developed having  $O(n^3)$  binary variables and restrictions, respectively (see Sect. 8.1.2).

Multi-stage cutting stock problems of two and more dimensions are considered in [7]. Among other topics, the three-dimensional guillotine CSP is also addressed in [3].

## 6.8 Three-Dimensional Guillotine Cutting

Let a (larger) rectangular parallelepiped of length  $L$ , width  $W$  and height  $H$ , called *container*, be given. Rectangular pieces (boxes) of  $m$  different types should be obtained by a sequence of G-cuts (now parallel to the sides of the parallelepiped). A box of type  $i$  has length  $\ell_i$ , width  $w_i$ , height  $h_i$ , and profit coefficient (value)  $c_i$ ,  $i \in I = \{1, \dots, m\}$ . The task is to find an orthogonal (cutting) pattern with maximal total value (Fig. 6.12).

Let  $v(r, s, t)$  denote the optimal value of a guillotine pattern for parallelepiped  $r \times s \times t$ ,  $0 \leq r \leq L$ ,  $0 \leq s \leq W$ ,  $0 \leq t \leq H$ .



**Fig. 6.12** Three-dimensional guillotine patterns

In order to formulate an appropriate recursion, we use an initialization similar to that of the two-dimensional case:

- Set  $v(r, s, t) := 0$ , for all  $r \in S(\ell, L)$ ,  $s \in S(w, W)$ , and  $t \in S(h, H)$ .
- For  $i = 1, \dots, m$  set  $v(\ell_i, w_i, h_i) := c_i$ .

That leads to the recursion

$$\begin{aligned} v(\tilde{L}, \tilde{W}, \tilde{H}) &:= \max \{ \\ &v(p_{\ell}(\tilde{L}-1), \tilde{W}, \tilde{H}), v(\tilde{L}, p_w(\tilde{W}-1), \tilde{H}), v(\tilde{L}, \tilde{W}, p_h(\tilde{H}-1)), \\ &\max_{r \in S(\ell, L)} \{v(r, \tilde{W}, \tilde{H}) + v(p_{\ell}(\tilde{L}-r), \tilde{W}, \tilde{H}) : r \leq \frac{\tilde{L}}{2}, r + p_{\ell}(\tilde{L}-r) = \tilde{L}\}, \\ &\max_{s \in S(w, W)} \{v(\tilde{L}, s, \tilde{H}) + v(\tilde{L}, p_w(\tilde{W}-s), \tilde{H}) : s \leq \frac{\tilde{W}}{2}, s + p_w(\tilde{W}-s) = \tilde{W}\}, \\ &\max_{t \in S(h, H)} \{v(\tilde{L}, \tilde{W}, t) + v(\tilde{L}, \tilde{W}, p_h(\tilde{H}-t)) : t \leq \frac{\tilde{H}}{2}, t + p_h(\tilde{H}-t) = \tilde{H}\} \}, \\ &\tilde{L} \in S(\ell, L), \quad \tilde{W} \in S(w, W), \quad \tilde{H} \in S(h, H). \end{aligned}$$

Applying the concept of reduced sets of potential allocation points is also possible and can lead to a remarkable saving of computational efforts with respect to time and space. Let  $\tilde{p}_{\ell}(x)$ ,  $\tilde{p}_w(x)$ ,  $\tilde{p}_h(x)$  denote the largest element in  $S^{red}(\ell, L)$ ,  $S^{red}(w, W)$ ,  $S^{red}(h, H)$ , respectively, not larger than  $x$ , then we obtain the following recursion:

$$\begin{aligned} v(\tilde{L}, \tilde{W}, \tilde{H}) &:= \max \{ \\ &v(\tilde{p}_{\ell}(\tilde{L}-1), \tilde{W}, \tilde{H}), v(\tilde{L}, \tilde{p}_w(\tilde{W}-1), \tilde{H}), v(\tilde{L}, \tilde{W}, \tilde{p}_h(\tilde{H}-1)), \\ &\max_{r \in S(\ell, L)} \{v(r, \tilde{W}, \tilde{H}) + v(\tilde{p}_{\ell}(\tilde{L}-r), \tilde{W}, \tilde{H}) : r \leq \frac{\tilde{L}}{2}, r + \tilde{p}_{\ell}(\tilde{L}-r) > \tilde{p}_{\ell}(\tilde{L}-1)\}, \\ &\max_{s \in S(w, W)} \{v(\tilde{L}, s, \tilde{H}) + v(\tilde{L}, \tilde{p}_w(\tilde{W}-s), \tilde{H}) : s \leq \frac{\tilde{W}}{2}, s + \tilde{p}_w(\tilde{W}-s) > \tilde{p}_w(\tilde{W}-1)\}, \\ &\max_{t \in S(h, H)} \{v(\tilde{L}, \tilde{W}, t) + v(\tilde{L}, \tilde{W}, \tilde{p}_h(\tilde{H}-t)) : t \leq \frac{\tilde{H}}{2}, t + \tilde{p}_h(\tilde{H}-t) > \tilde{p}_h(\tilde{H}-1)\} \}, \\ &\tilde{L} \in S^{red}(\ell, L), \quad \tilde{W} \in S^{red}(w, W), \quad \tilde{H} \in S^{red}(h, H). \end{aligned}$$

It is easy to see, that the recursion above can be restricted to obtain special pattern-types, for example, 3-stage patterns with horizontal G-cuts (parallel to the  $L \times W$ -plane) in the first stage.

More details to solution methods are given in [3] where the three-dimensional knapsack, the cutting stock, and the strip packing problem with guillotine constraint are addressed. Rotation of pieces as well as the  $k$ -stage case are considered, too. Detailed algorithms are presented based on the reduced sets of potential allocation points (called *raster points*). A *Constraint Programming* approach for three-dimensional guillotine cutting is proposed in [1].

## 6.9 Exercises

**Exercise 6.1** Consider Example 6.1 and compute an optimal pattern for  $\tilde{L} = 150$ ,  $\tilde{W} = 105$  using Table 6.2 [recursion (6.3)]. Compare it with the pattern obtained from Table 6.3 [recursion (6.4)] for  $\tilde{L} = 150$  and  $\tilde{W} = 105$ .

**Exercise 6.2** How can rotation of pieces by  $90^\circ$  be handled when computing optimal guillotine patterns?

**Exercise 6.3** Which knapsack problems have to be solved when all pieces are rotatable and a 2-stage pattern is searched? (The direction of G-cuts in the first stage is not predefined.) Compute an optimal 2-stage pattern for the following input data:

$$L = 32, W = 27, m = 3, \ell = (6, 7, 5)^\top, w = (4, 4, 5)^\top, \text{ and } c = (4, 5, 4)^\top.$$

**Exercise 6.4** Which knapsack problems have to be solved to compute an optimal 2-stage pattern for the exact case?

**Exercise 6.5** Applying the algorithm of Wang, compute an optimal guillotine pattern for the instance with input data  $L \times W = 10 \times 8$ ,  $m = 3$ ,  $\ell_1 \times w_1 = 6 \times 4$ ,  $u_1 = 2$ ,  $\ell_2 \times w_2 = 5 \times 5$ ,  $u_2 = 2$ , and  $\ell_3 \times w_3 = 2 \times 3$ ,  $u_3 = 3$ . Use  $c_i := \ell_i w_i$  for all  $i$ .

**Exercise 6.6** Let the following squares be given: two of size  $6 \times 6$ , two of size  $5 \times 5$ , three of size  $4 \times 4$ , and four of size  $3 \times 3$ . Which guillotine pattern of rectangle  $15 \times 11$  has maximal area utilization? Show that it is not possible to obtain an optimal pattern with the Wang algorithm if all feasible (partial) solutions  $(r, s, v, a)$  with  $rs - v \geq 4$  are removed.

## 6.10 Solutions

**To Exercise 6.1** The recursion based on the set of potential allocation points yields:

$$v(150, 105) = v(143, 102) = 52 \text{ with } v(143, 102) = v(62, 102) + v(81, 102) = 20 + 32, \quad v(62, 102) = v(62, 82) = 2v(62, 41) = 4v(31, 41), \quad v(81, 102) = 2v(81, 51).$$

The recursion based on the reduced set of potential allocation points yields:

$$v(150, 105) = v(124, 103) = 42 \text{ with } v(124, 103) = v(31, 103) + v(93, 103) = 10 + 32, v(93, 103) = 2v(93, 51) = 2v(81, 51), v(31, 103) = 2v(31, 41).$$

**To Exercise 6.2** Without loss of generality, let  $\ell_i \neq w_i$  for  $i = 1, \dots, m'$  with  $m' \leq m$  and  $\ell_i = w_i$  for  $i = m' + 1, \dots, m$ . Then we define for any non-quadratic piece  $i$  another piece  $m + i$  with length  $\ell_{m+i} := w_i$  and width  $w_{m+i} = \ell_i$ ,  $i = 1, \dots, m'$ . In case of upper bounds  $u_i$ , how often piece  $i$  could be obtained, the total number of items of type  $i$  and  $m + i$  has to be limited by  $u_i$ .

**To Exercise 6.3** After adding the rotated piece types and some resorting, we have the following input data:  $m = 5$ ,  $\ell = (6, 7, 5, 4, 4)^\top$ ,  $w = (4, 4, 5, 6, 7)^\top$ ,  $c = (4, 5, 4, 4, 5)^\top$ , and  $(L, W) = (32, 27)$  or  $(L, W) = (27, 32)$ . Knapsack problems to be solved are ( $z_1 = F_j(27)$ ,  $z_2 = F_j(32)$ ):

| $j$ | $\bar{w}_j$ | $F_j(L)$   | $z_1$ | $z_2$ |
|-----|-------------|--|-------|-------|
| 1   | 4           | $\max\{4a_1 + 5a_2 : 6a_1 + 7a_2 \leq L, a_i \in \mathbb{Z}_+\}$ | 19    | 22    |
| 2   | 5           | $\max\{5a_2 + 4a_3 : 7a_2 + 5a_3 \leq L, a_i \in \mathbb{Z}_+\}$ | 21    | 25    |
| 3   | 6           | $\max\{5a_2 + 4a_4 : 7a_2 + 4a_4 \leq L, a_i \in \mathbb{Z}_+\}$ | 25    | 32    |
| 4   | 7           | $\max\{5a_5 : 4a_5 \leq L, a_i \in \mathbb{Z}_+\}$               | 30    | 40    |

(Dominated variables are omitted.)

With  $G(W) := \max\{\sum_{j=1}^4 F_j(L)y_j : \sum_{j=1}^4 \bar{w}_jy_j \leq W, y_j \in \mathbb{Z}_+\}$  we obtain

$$G(27) = 3F_4(32) + F_3(32) = 152 \text{ and } G(32) = 8F_1(27) = 152.$$

Note that in this example the optimal vertical and the optimal horizontal 2-stage pattern have the same value.

**To Exercise 6.4** In the exact case of 2-stage cutting some simplification arises, namely the computation of  $F_j(L)$ :  $F_j(L) := \max\{\sum_{i \in I_j} c_i x_i : \sum_{i \in I_j} \ell_i x_i \leq L, x_i \in \mathbb{Z}_+\}$ . Similar knapsack problems result if the G-cuts of the first stage have to be vertical.

**To Exercise 6.5** The optimal value is 68. Two items of type 2 and three of type 3 are allocated.

**To Exercise 6.6** The optimal value equals 156. Each square of size 6, 5 and 3 is used twice, and the square of size 4 once. That guillotine pattern results from partial solutions of rectangles  $15 \times 6$  and  $15 \times 5$ . Since the pattern of  $15 \times 5$  results from a pattern of  $14 \times 5$ , five units of waste occur, and therefore, an optimal pattern cannot be obtained by the Wang algorithm if the assumed reduction rule is used.

## References

1. R.R. Amossen, D. Pisinger, Multi-dimensional bin packing problems with guillotine constraints. *Comput. Oper. Res.* **37**, 1999–2006 (2010)
2. G.F. Cintra, F.K. Miyazawa, Y. Wakabayashi, E.C. Xavier, Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *Eur. J. Oper. Res.* **191**, 61–85 (2008)

3. T.A. de Queiroz, F.K. Miyazawa, Y. Wakabayashi, E.C. Xavier, Algorithms for 3d guillotine cutting problems: unbounded knapsack, cutting stock and strip packing. *Comput. Oper. Res.* **39**, 200–212 (2012)
4. H. Dyckhoff, A new linear approach to the cutting stock problem. *Oper. Res.* **29**(6), 1092–1104 (1981)
5. F. Furini, E. Malaguti, R.M. Durán, A. Persiani, P. Toth, A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *Eur. J. Oper. Res.* **218**(1), 251–260 (2012)
6. G. Galambos, A. van Vliet, Lower bounds for 1-, 2- and 3-dimensional on-line bin packing algorithms. *Computing* **52**, 281–297 (1994)
7. P.C. Gilmore, R.E. Gomory, Multistage cutting stock problems of two and more dimensions. *Oper. Res.* **13**, 94–120 (1965)
8. A. Lodi, M. Monaci, Integer linear programming models for 2-staged two-dimensional knapsack problems. *Math. Program. Ser. B* **94**, 257–278 (2003)
9. J. Puchinger, G.R. Raidl, Models and algorithms for three-stage two-dimensional bin packing. *Eur. J. Oper. Res.* **183**, 1304–1327 (2007)
10. G. Scheithauer, On the MAXGAP problem for cutting stock problems. *J. Inf. Process. Cybern.* **30**, 111–117 (1994)
11. E. Silva, F. Alvelos, J.M.V. de Carvalho, An integer programming model for two- and three-stage two-dimensional cutting stock problems. *Eur. J. Oper. Res.* **205**, 699–708 (2010)
12. P.Y. Wang, Two algorithms for constrained two-dimensional cutting stock problems. *Oper. Res.* **31**, 573–586 (1983)

# Chapter 7

## Packing Rectangles into a Strip

Within this chapter we consider the following problem of minimizing material consumption in the two-dimensional case: there are given a list (set)  $\mathcal{L} = (R_1, \dots, R_m)$  of rectangles (pieces, items)  $R_i, i \in I = \{1, \dots, m\}$  and a strip of fixed width  $W$  and unlimited height. Rectangle  $R_i$  has width  $w_i$  and height  $h_i$ . The task is to pack all pieces orthogonally into the strip such that the minimal needed height is used.

Obviously, solution approaches for this *Strip Packing Problem* (SPP) can involve those of the orthogonal rectangle packing feasibility problem (OPP), but differences result not only by the objective function. Again, to simplify the description, we do not allow rotation of pieces. This restriction is important in several applications such as, for instance,

- *Resource Constraint Scheduling:* The available amount of a resource (per time unit) is limited by  $W$ . Several jobs can be processed in parallel where job  $i \in I$  requires  $w_i$  units of this resource and has to be processed  $h_i$  time units without interruption.
- *Optimal Memory Management:* Input value  $w_i$  represents the memory requirement and  $h_i$  the desired time duration of job  $i \in I$ .

The SPP belongs, like many other cutting and packing problems, to the class of NP-hard optimization problems [18]. It is polynomially equivalent to the OPP. Due to the NP-hardness, efficient heuristics are considered in the majority of related literature.

Besides an appropriate modeling of the SPP, we describe and analyze some heuristic and metaheuristic approaches. Moreover, based on lower bounds, we present a b&b algorithm which packs the pieces sequentially. The Guillotine Strip Packing Problem is considered as well.

## 7.1 Integer Linear Programming Models

Within this section we formulate two ILP models of the SPP. Let a strip of width  $W$  and unbounded height as well as  $m$  non-rotatable rectangular pieces  $R_i = w_i \times h_i$ ,  $i \in I = \{1, \dots, m\}$  be given. Without loss of generality, we assume  $0 < w_i \leq W$ ,  $0 < h_i$ ,  $i \in I$ , and that all input data are positive integers, if not stated otherwise. Find an arrangement (packing pattern)  $A(I)$  with minimal height  $H$ , i.e., we have to identify an allocation point (lower-left corner)  $(x_i, y_i)$  for each  $i \in I$  such that  $A(I) = \bigcup_{i \in I} R_i(x_i, y_i)$  is a feasible pattern and the occupied height  $H = H(A(I)) := \max_{i \in I} (y_i + h_i)$  is minimal.

### 7.1.1 A Beasley-Type Model

Although the concept of (reduced) sets of potential allocation points (Sect. 2.7) can be applied in the modeling, we do not use it here for the sake of a simpler description. First of all, we define index sets

$$J := \{0, 1, \dots, W - 1\} \quad \text{and} \quad K := \{0, 1, \dots, \bar{H} - 1\},$$

where  $\bar{H} \in \mathbb{Z}_+$  is the height of a known feasible packing pattern or another upper bound of the minimal height needed to pack all items. Moreover, let

$$J_i := \{0, \dots, W - w_i\} \quad \text{and} \quad K_i := \{0, \dots, \bar{H} - h_i\}, \quad i \in I,$$

represent feasible coordinates for piece  $i \in I$  in  $W$ - and  $H$ -direction, respectively. Similar to Sect. 5.2.1, we define a binary variable  $x_{ipq}$  for all  $i \in I$ ,  $p \in J_i$ , and  $q \in K_i$ , where  $x_{ipq} = 1$  if and only if rectangle  $R_i$  gets allocation point  $(x_i, y_i) = (p, q)$ . To describe the region of the strip covered by

$$R_i(p, q) = \{(x, y) : p \leq x < p + w_i, q \leq y < q + h_i\}$$

we introduce again 0/1-coefficients  $a_{ipqrs}$  according to

$$a_{ipqrs} := \begin{cases} 1, & \text{if } x_{ipq} = 1, (r, s) \in R_i(p, q), \\ 0, & \text{otherwise,} \end{cases}$$

where  $r \in J$ ,  $s \in K$ ,  $p \in J_i$ ,  $q \in K_i$ , and  $i \in I$ . Herewith we obtain a model with binary variables only:

### Beasley-type model of the SPP

$$z = \min H \quad \text{s.t.} \quad (7.1a)$$

$$h_i + \sum_{q \in K_i} q \sum_{p \in J_i} x_{ipq} \leq H, \quad i \in I, \quad (7.1b)$$

$$\sum_{i \in I} \sum_{p \in J_i} \sum_{q \in K_i} a_{ipqrs} x_{ipq} \leq 1, \quad r \in J, s \in K, \quad (7.1c)$$

$$\sum_{p \in J_i} \sum_{q \in K_i} x_{ipq} = 1, \quad i \in I, \quad (7.1d)$$

$$x_{ipq} \in \{0, 1\}, \quad i \in I, p \in J_i, q \in K_i.$$

Restrictions (7.1b) ensure that all packed pieces are completely contained within the strip of height  $H$ . Non-overlapping of items is guaranteed by constraints (7.1c). Finally, due to (7.1d), all pieces are packed exactly once.

Obviously, almost  $mHW$  binary variables are included in model (7.1). For that reason, only these variables being necessary according to the concept of potential allocation points should be defined. In  $W$ -direction we can use the reduced set, that means,  $J_i$  can be replaced by

$$J_i \cap \{p_w(W - r) : r \in S(w, W)\}$$

where

$$S(w, W) := \{r = w^\top a : r \leq W, a \in \mathbb{B}^m\} \quad \text{and} \quad p_w(x) := \max\{r \in S(w, W) : r \leq x\}.$$

Since the strip height is not known in advance. a reduced set cannot be used, only

$$K_i \cap S(h) \quad \text{with} \quad S(h) := \{r : r = h^\top a, a \in \mathbb{B}^m\}.$$

### 7.1.2 A Linear Mixed-Integer Model

Since all pieces have to be packed, we do not need decision variables to indicate whether a piece is packed or not. To characterize the mutual position of pieces  $R_i$  and  $R_j$ , we define binary variables  $u_{ij}$  and  $v_{ij}$ ,  $i \neq j$ ,  $i, j \in I$ . Then, non-overlapping of  $R_i$  and  $R_j$  can be modeled by

$$u_{ij} = 1 \Rightarrow x_j \geq x_i + w_i \quad \text{and} \quad v_{ij} = 1 \Rightarrow y_j \geq y_i + h_i.$$

In this way, we obtain the following model:

### Linear mixed-integer model of the SPP

$$z = \min H \quad \text{s.t.} \quad (7.2a)$$

$$0 \leq x_i \leq W - w_i, \quad 0 \leq y_i \leq H - h_i, \quad i \in I, \quad (7.2b)$$

$$x_i + w_i \leq x_j + W(1 - u_{ij}), \quad y_i + h_i \leq y_j + \bar{H}(1 - v_{ij}), \quad i \neq j, \quad i, j \in I, \quad (7.2c)$$

$$u_{ij} + u_{ji} + v_{ij} + v_{ji} = 1, \quad i < j, \quad i, j \in I, \quad (7.2d)$$

$$u_{ij}, v_{ij} \in \{0, 1\}, \quad i \neq j, \quad i, j \in I,$$

where  $\bar{H} \in \mathbb{Z}_+$  is the height of a feasible pattern. Because of conditions (7.2b) piece  $R_i(x_i, y_i)$  is completely placed within a strip of width  $W$  and height  $H$ . According to (7.2a) the strip height is minimized. Restrictions (7.2d) ensure that for each pair  $i \neq j$  exactly one of the four  $u$ - and  $v$ -variables gets value 1. Therefore, conditions (7.2c) imply non-overlapping of the two pieces. Due to use of such binary variables, the model can be seen to be a Padberg-type model.

Note that the concept of sequence-pairs, as mentioned in Sect. 5.2.2 for the OPP, can be used to reduce the number of binary variables and to improve the performance of solution methods because of a tighter LP relaxation.

Certainly, the number of binary variables can become too large for SPP instances of practical interest so that the solution by means of standard software is too expensive or even impossible. Nevertheless, the models provide starting points to compute meaningful lower bounds for the optimal height or to obtain approximate solutions.

## 7.2 Lower Bounds

To better evaluate the quality of heuristic solutions and for bounding purposes within b&b approaches the knowledge of strong lower bounds can be very useful. Besides some natural lower bounds, a wide variety of further lower bounds exists for the SPP. In the following we describe some of them including combinatorial and LP based ones and give related performance results. Let  $E = (m, W, w, h)$  represent any SPP instance where  $w = (w_1, \dots, w_m)^\top$  and  $h = (h_1, \dots, h_m)^\top$ . Or, alternatively, the instance can be given as a list  $L$  of rectangles, and let  $I := \{1, \dots, m\}$ . Moreover, let  $H^* = H^*(E)$  denote the optimal value of SPP instance  $E$ .

### 7.2.1 Natural Lower Bounds

The simple *maximum height bound*

$$lb_h = lb_h(E) := \max\{h_i : i \in I\}$$

and the *area or material bound*

$$lb_A = lb_A(E) := \left\lceil \frac{1}{W} \sum_{i \in I} w_i h_i \right\rceil$$

can be seen as trivial lower bounds for the SPP. Both of them can be arbitrarily worse.

**Proposition 7.1** *For  $lb_h$  and  $lb_A$  we have*

$$\sup_E \frac{H^*(E)}{lb_h(E)} = \infty \quad \text{and} \quad \sup_E \frac{H^*(E)}{lb_A(E)} = \infty.$$

*Proof* See Exercise 7.1. ■

However, the combination

$$lb_0 = lb_0(E) := \max\{lb_h, lb_A\}$$

of  $lb_h$  and  $lb_A$  has a bounded performance ratio.

**Theorem 7.2 (Martello et al. [23])** *The worst-case performance ratio of  $lb_0$  is equal to 2, i.e.,*

$$\sup_E \frac{H^*(E)}{lb_0(E)} = 2.$$

*Proof* Let  $lb_0 = lb_0(E)$ . In order to obtain  $H^*(E) \leq 2lb_0$ , we apply Steinberg's theorem 5.6 to the rectangle  $W \times 2lb_0$ . Since  $2 \sum_{i \in I} w_i h_i \leq 2Wlb_0$  and  $\max_{i \in I} h_i \leq lb_0$  all assumptions of Steinberg's theorem are satisfied and the desired inequality follows.

Considering the sequence of instances  $E_k$ ,  $k = 1, 2, \dots$ , with  $m = 2k$ ,  $W = 2k$ ,  $w_i = k + 1$ , and  $h_i = 1$  for  $i = 1, \dots, m$  it is easy to see that the performance ratio 2 cannot be improved. ■

Another simple lower bound with performance ratio 2 is also proposed in [23].

**Theorem 7.3 (Martello et al. [23])** *Let  $h_i \geq h_{i+1}$  for all  $i \in I$  and let*

$$k := \max \left\{ j \in I : \sum_{i=1}^j w_i \leq W \right\}.$$

*For item  $l > k$  with  $w_l + \sum_{i=1}^k w_i > W$  let  $i(l)$  denote the smallest index such that  $w_l + \sum_{i=1}^{i(l)} w_i > W$  holds. Then*

$$lb_{MMV1} := \max \left\{ lb_0, \max \left\{ h_l + h_{i(l)} : l > k \text{ and } w_l + \sum_{i=1}^k w_i > W \right\} \right\} \quad (7.3)$$

defines a valid lower bound for the SPP which, obviously, dominates  $lb_0$ , but its worst-case performance ratio is also 2.

*Proof* The validity of  $lb_{MMV1}$  to be a lower bound follows immediately from the fact that the items  $1, \dots, i(l)$  and  $l$  cannot be placed side by side within the strip. The proof of the performance ratio is similar to the proof of Theorem 7.2 using the same sequence of instances. ■

A lower bound oriented on the total width of the items is considered by Boschetti and Montaletti [12]. Similar to  $lb_h$  and  $lb_{MMV1}$ , only height values of pieces are used. To this end, the minimum number of bins (in [12], the notation *layer* is used instead of bin)

$$\lambda := \left\lceil \frac{1}{W} \sum_{i \in I} w_i \right\rceil$$

needed for an 1BPP instance with item length  $w_i$  and bin capacity  $W$  is defined. Clearly, the  $\lambda$  smallest  $h_i$ -values define a valid lower bound, named  $lb_h^1$ , for the 2SPP. Without loss of generality, we assume  $h_i \leq h_{i+1}$  for all  $i$  in the following. Then  $lb_h^1$  is given by

$$lb_h^1 := \sum_{k=1}^{\lambda} h_k.$$

Based on considerations done in [12], we define

$$w_{top} := \sum_{i \in I} w_i - (\lambda - 1)W$$

as the remaining total width if all previous  $(\lambda - 1)$  bins (layers) are filled completely. For  $k = 1, \dots, \lambda$ , let

$$\delta_k := \min\{\delta : \sum_{i=k}^{k+\delta} w_i \geq w_{top}\}$$

indicate that item which leads to a total width larger than or equal to  $w_{top}$  when assigning items  $k, \dots, k + \delta(k)$  to a bin, and let

$$lb_\lambda(k) := \sum_{i=1}^{k-1} h_i + \sum_{i=k+\delta_k}^{\lambda+\delta_k} h_i.$$

Then, because of the assumed sorting of the items,  $lb_\lambda(k)$  is a valid lower bound for the SPP for each  $k = 1, \dots, \lambda$ , too. None of them dominates the other, but all of them dominates  $lb_h^1$ . Therefore,

$$lb_\lambda := \max_{k=1, \dots, \lambda} lb_\lambda(k)$$

defines a lower bound for the SPP. Altogether, none of the bounds  $lb_0$ ,  $lb_{MMV1}$ , and  $lb_\lambda$  dominates the other two so that the maximum of them should be taken.

Interestingly, the bounds  $lb_\lambda(k)$  depend on an additional sorting of pieces with respect to the  $w_i$ -values if several items have the same height, and therefore  $lb_\lambda$ , too. A maximal  $lb_\lambda(k)$  can be obtained if the following properties (maintaining  $h_i \leq h_{i+1}$  for all  $i$ ) hold:

- (1) for  $1 \leq i < k$ :  $h_i = h_k \Rightarrow w_i \geq \max\{w_j : j \geq k \text{ with } h_j = h_k\}$ ,
- (2) for  $k \leq i < m$ :  $h_i = h_{i+1} \Rightarrow w_i \leq w_{i+1}$ .

### 7.2.2 Combinatorial Bounds

The following lower bound, also proposed by Martello et al. [23], can be seen as a generalization of a lower bound for the 1BPP introduced by Martello and Toth [22].

Let  $\alpha \in [1, W/2] \cap \mathbb{N}$ . Moreover, let

$$\begin{aligned} J_1 &:= \{j \in I : W - \alpha < w_j \leq W\}, \\ J_2 &:= \{j \in I : W/2 < w_j \leq W - \alpha\}, \quad \text{and} \\ J_3 &:= \{j \in I : \alpha \leq w_j \leq W/2\} \end{aligned}$$

represent some subsets of items which depend on the parameter  $\alpha$ . Using

$$lb(\alpha) := \sum_{j \in J_1 \cup J_2} h_j + \max \left\{ 0, \left\lceil \frac{1}{W} \left( \sum_{j \in J_3} w_j h_j - \left( \sum_{j \in J_2} (W - w_j) h_j \right) \right) \right\rceil \right\},$$

the lower bound defined by Martello et al. [23], is as follows:

$$lb_{MMV2} := \max_{1 \leq \alpha \leq W/2} \{lb(\alpha)\}$$

Note that lower bound  $lb_{MMV2}$  dominates the area bound  $lb_A$ , but not the bound  $lb_{MMV1}$  as the following example shows.

*Example 7.1* For the SPP instance  $E_1$  with  $W = 10$ ,  $m = 3$ , and  $w_i = h_i = 6$  for  $i \in \{1, 2, 3\}$ , we obtain  $lb_{MMV1} = 12$  and  $lb_{MMV2} = 18$  (for  $\alpha \in [1, 5] \cap \mathbb{N}$ ). Contrarily, for the instance  $E_2$  with  $W = 10$ ,  $m = 6$ ,  $\bar{w} = (3, 3, 2, 3, 2, 5)$ , and  $\bar{h} = (9, 8, 6, 5, 4, 4)$ , we obtain  $lb_{MMV1} = 12$  and  $lb_{MMV2} = 11$  (for  $\alpha \in \{1, 2\}$ ). ■

A slightly improved lower bound, based on the same index sets, is proposed by Bekrar et al. [6]. Let  $nps$  denote the maximal number of items of  $J_3$  which can be packed beside pieces of  $J_2$ . Then, assuming  $w_i h_i \leq w_{i+1} h_{i+1}$  for all  $i \in J_3$ ,

$$lb_{BKC} := \max_{1 \leq \alpha \leq W/2} \{lb_{BKC}(\alpha)\}$$

is a lower bound for the SPP where

$$lb_{BKC}(\alpha) := \sum_{j \in J_1 \cup J_2} h_j + \max \left( \left\lceil \frac{1}{W} \sum_{j=1}^{|J_3|-nps} w_j h_j \right\rceil, h_{[|J_3|-nps]} \right)$$

and  $h_{[j]}$  is the height of item  $j$  if the items of  $J_3$  are sorted according to non-decreasing height. The number  $nps$  is hard to compute, but, because of the assumption  $w_i h_i \leq w_{i+1} h_{i+1}$  for all  $i \in J_3$ , we have  $nps \leq \max\{k \in J_3 : \sum_{i=1}^k w_i h_i \leq \sum_{j \in J_2} (W - w_j) h_j\}$ .

Another lower bound is proposed by Alvarez-Valdés, Parreño, and Tamarit in [3]. Using

$$lb_{APT}(\alpha) := \sum_{j \in J_1} h_j + \max \left\{ \left\lceil \frac{1}{\lfloor \frac{W}{\alpha} \rfloor} \sum_{j \in J_2} h_j \right\rceil, \max\{h_j : j \in J_2\} \right\},$$

the bound is as follows:

$$lb_{APT} := \max_{1 \leq \alpha \leq W/2} \{b_3(\alpha)\}.$$

Some extensions of this bound, several other bounds, and dominance investigations are also addressed in [3].

### 7.2.3 LP Based Bounds

Next, we consider some ILP and LP relaxations of the SPP. To formulate a Kantorovich-type relaxation, we define binary variables  $a_{is}$  for all  $i \in I$  and all  $s \in K := \{1, \dots, \bar{H}\}$  where  $a_{is} = 1$  if and only if  $R_i(x_i, y_i) \cap \{(x, y) : 0 \leq x \leq W, s-1 < y < s\} \neq \emptyset$ .

#### Horizontal relaxation of Kantorovich-type

$$z^{Kan} = \min H \quad \text{s.t.} \tag{7.4a}$$

$$s \cdot a_{is} \leq H, \quad i \in I, s \in K, \tag{7.4b}$$

$$\sum_{i \in I} w_i a_{is} \leq W, \quad s \in K, \tag{7.4c}$$

$$\sum_{s \in K} a_{is} = h_i, \quad i \in I, \tag{7.4d}$$

$$a_{is} \in \{0, 1\}, \quad i \in I, s \in K.$$

Each vector  $a_s = (a_{1s}, \dots, a_{ms})^T$  represents a feasible (one-dimensional) pattern in horizontal direction due to restrictions (7.4c). The number of such patterns is bounded by  $H$  in (7.4b) and  $H$  is minimized in (7.4a). Conditions (7.4d) ensure that exactly  $h_i$  (one-dimensional) patterns have  $a_{is} = 1$  implying that the whole area of piece  $i$  is packed for all  $i \in I$ . In this relaxation the  $x$ -coordinate of the allocation point of  $R_i$  is not determined and, moreover, it is not guaranteed that each item occurs in consecutive patterns. Note that the LP relaxation of model (7.4) is weak since its optimal value  $z_{LP}^{Kan}$  is at most  $lb_0 = \max\{lb_h, lb_A\}$ , in general (see Exercise 7.2).

In order to strengthen the relaxation we consider only horizontal (in fact one-dimensional) strip patterns with integer coefficients maintained in the LP relaxation. To this end, we define variables  $t_k$  indicating how often pattern  $a^k \in \mathbb{B}^m$  is used. Let  $\bar{K}$  represent an index set of all feasible binary patterns, i.e., of  $\{a \in \mathbb{B}^m : w^\top a \leq W\}$ . We obtain the

### Horizontal binary bar relaxation

$$z^{b-hor} := \min \sum_{k \in \bar{K}} t_k \quad \text{s.t.} \quad (7.5a)$$

$$\sum_{k \in \bar{K}} a_{ik} t_k = h_i, \quad i \in I, \quad (7.5b)$$

$$t_k \in \mathbb{Z}_+, \quad k \in \bar{K}.$$

It is easy to see, bound  $z^{b-hor}$  dominates the combined area and height bound  $lb_0$ . Nonetheless, we have

**Theorem 7.4** *The performance ratio of the horizontal binary bar relaxation is 2, i.e.,*

$$\sup_E \frac{H^*(E)}{z^{b-hor}(E)} = 2.$$

*Proof* Due to Theorem 7.2, we have

$$H^* \leq 2lb_0 \leq 2z^{b-hor} \quad \text{and therefore,} \quad \sup_E \frac{H^*(E)}{z^{b-hor}(E)} \leq 2.$$

This bound cannot be improved since we have  $H^* = 2k$  and  $z^{b-hor} = k + 1$  for the instance  $E_k = (m = k + 1, W = k, w_i = 1, h_i = k)$ . Thus,

$$\frac{H^*(E_k)}{z^{b-hor}(E_k)} = \frac{2k}{k + 1} \xrightarrow{k \rightarrow \infty} 2.$$

■

Model (7.5) is a model of a particular *Cutting Stock Problem*; all patterns have only 0/1-coefficients. Some drawback of this model is the exponential number of variables. However its LP relaxation can be solved efficiently using the column generation technique. Let  $z_{LP}^{b-hor}$  denote the lower bound of the SPP obtained by the LP relaxation of (7.5).

If we replace  $a_{ik} \in \{0, 1\}$  for all  $i$  and  $k$  by conditions  $a_{ik} \in \mathbb{Z}_+$ , the two-dimensional SPP is relaxed as a one-dimensional CSP. Let  $\tilde{K}$  denote an index set of  $\{a \in \mathbb{Z}_+^m : w^\top a \leq W\}$ .

### Horizontal bar relaxation

$$z^{hor} := \min \sum_{k \in \tilde{K}} t_k \quad \text{s.t.} \quad (7.6a)$$

$$\sum_{k \in \tilde{K}} a_{ik} t_k = h_i, \quad i \in I, \quad (7.6b)$$

$$t_k \in \mathbb{Z}_+, \quad k \in \tilde{K}.$$

Obviously, we have  $\bar{K} \subseteq \tilde{K}$  and therefore

$$z^{b-hor} \geq z^{hor} \quad \text{and} \quad z_{LP}^{b-hor} \geq z_{LP}^{hor}.$$

Note that

$$\sup_E \frac{H^*(E)}{z^{hor}(E)} = \infty$$

holds caused by the instance  $E_1(W) = (m = 1, W, w_1 = 1, h_1 = W)$ .

There is another interpretation of the horizontal bar relaxation. Let  $\bar{m}$  denote the number of different widths  $\bar{w}_i$  ( $i \in \bar{I} = \{1, \dots, \bar{m}\}$ ) of all pieces. It is bounded by  $W$ . Let  $K_h$  be an index set of all (one-dimensional) patterns  $a^k \in \mathbb{Z}_+^{\bar{m}}$  satisfying  $\sum_{i \in \bar{I}} \bar{w}_i a_{ik} \leq W$  for  $k \in K_h$ . Furthermore, let  $d_i := \sum_{j \in I: w_j = \bar{w}_i} h_j$  for  $i \in \bar{I}$  represent the total height of all items having the same width. We denote by  $t_k$  the frequency pattern  $a^k \in \mathbb{Z}^{\bar{m}}$  is used in the solution. Then the horizontal bar relaxation is also given by

$$z^{hor} = \min \sum_{k \in K_h} t_k \quad \text{s.t.}$$

$$\sum_{k \in K_h} a_{ik} t_k = d_i, \quad i \in \bar{I},$$

$$t_k \in \mathbb{Z}_+, \quad k \in K_h.$$

For a fixed  $W$  and  $h_{max} := \max\{h_i : i \in I\}$ , i.e., both values are bounded, it is known that the bar relaxation bound is asymptotically tight.

**Theorem 7.5 (Kenyon and Rémy [20])** *For every SPP instance E, we have*

$$H^* \leq z^{hor} + \bar{m} \cdot h_{max}.$$

In a similar way, we can define a bar relaxation with vertical (one-dimensional) patterns. In this case, we have to regard that the strip height  $H$  itself is a variable which has to be taken into account during the column generation process.

As already mentioned above, relaxations (7.5) and (7.6) do not ensure that all items  $w_i \times 1$  of the same type are contained in consecutive patterns. For that reason, we consider a *contiguous bar relaxation* of the SPP, too.

To formulate an MILP model, let  $\underline{H}$  denote a lower bound of  $H^*$  with  $\underline{H} \geq lb_0$ . We define binary variables  $t_{iy}$  for each possible position  $y = h_i, \dots, \bar{H}$  and each piece  $i \in I$  as follows:

$$t_{iy} = 1 \Leftrightarrow \text{the upper edge of piece } i \text{ has coordinate } y,$$

that means, the projection of piece  $i$  onto the  $H$ -axis is  $(y - h_i, y]$  if and only if  $t_{iy} = 1$ . In order to register the actual strip height required to pack all pieces, we need to define some more binary variables  $\sigma_y$  for  $y = \underline{H} + 1, \dots, \bar{H}$  with

$$\sigma_y \geq t_{iy}, \quad i \in I, \quad y = \underline{H} + 1, \dots, \bar{H} \quad \text{and} \quad \sigma_y \geq \sigma_{y+1}, \quad y = \underline{H} + 1, \dots, \bar{H} - 1.$$

Hence, variable  $\sigma_y$  gets value 1 if at least one piece intersects with bar  $y$ . The second condition ensures a dense packing from below. Obviously, we aim to minimize the number of additional bars.

### Horizontal contiguous bar relaxation

$$z^{c-hor} := \underline{H} + \min \sum_{y=\underline{H}+1}^{\bar{H}} \sigma_y \quad \text{s.t.} \tag{7.7a}$$

$$\sigma_y \geq \sigma_{y+1}, \quad y = \underline{H} + 1, \dots, \bar{H} - 1, \tag{7.7b}$$

$$\sigma_y \geq t_{iy}, \quad i \in I, \quad y = \underline{H} + 1, \dots, \bar{H}, \tag{7.7c}$$

$$\sum_{y=h_i}^{\bar{H}} t_{iy} = 1, \quad i \in I, \tag{7.7d}$$

$$\sum_{i \in I} w_i \sum_{s=\max\{h_i, y\}}^{\min\{\bar{H}, y+h_i-1\}} t_{is} \leq W, \quad y = 1, \dots, \bar{H}, \tag{7.7e}$$

$$t_{1y} = 0, \quad y = h_1, \dots, (\underline{H} + h_1)/2 - 1, \tag{7.7f}$$

$$t_{iy} \in \{0, 1\}, \quad y = h_i, \dots, \bar{H}, \quad i \in I, \quad \sigma_y \in \{0, 1\}, \quad y = \underline{H} + 1, \dots, \bar{H}.$$

According to the objective function the strip height is minimized. Restrictions (7.7b) ensure that no empty bar can occur except on top of the strip. Due to (7.7d) each piece is placed once. The contiguity is derived by definition of the  $t_{iy}$ -variables and regarded in restrictions (7.7e). Constraints (7.7f) are not necessary, but they avoid some symmetric solutions obtainable by reflection on a horizontal line.

Obviously,  $z^{c-hor}$  dominates  $z^{hor}$  and  $z^{b-hor}$ . Thus, we have

$$lb_0 \leq z^{hor} \leq z^{b-hor} \leq z^{c-hor} \leq H^* \leq 2lb_0.$$

Exercise 7.12 shows that there exist instances of the 2SPP such that  $z^{b-hor} < z^{c-hor} < H^*$  holds.

Moreover, the following results are known.

**Proposition 7.6 (Martello et al. [23])** *The optimal value  $z^{c-hor}$  of the (horizontal binary) contiguous relaxation dominates  $lb_{MMV1}$ ,  $lb_{MMV2}$ , and  $lb_0$ .*

**Theorem 7.7 (Rykov [26])** *It holds*

$$\sup_E \frac{H^*(E)}{z^{c-hor}(E)} \geq 1.25 \quad \text{and} \quad \limsup_{E: \frac{z^{c-hor}}{h_{\max}} \rightarrow \infty} \frac{H^*(E)}{z^{c-hor}(E)} \leq 1.25.$$

In Fig. 7.1 an instance is depicted which proves the first part of the theorem. The second item of size  $1 \times 2$  can be placed within a pattern of height 4 in case of the contiguous problem. However, this is not possible when considering the SPP.

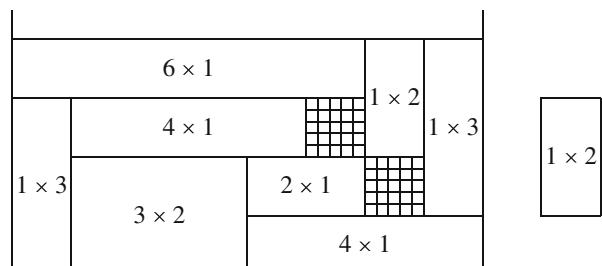
As a consequence, the performance ratio of the contiguous bound is limited by 1.25 and 2:

$$1.25 \leq \sup_E \frac{H^*(E)}{z^{c-hor}(E)} \leq 2.$$

Moreover, as shown in [10], the worst-case performance ratio of  $z^{c-hor}$  and  $z^{b-hor}$  is 2, too, i.e.,

$$\sup_E \frac{z^{c-hor}}{z^{b-hor}} = 2.$$

**Fig. 7.1** Item  $1 \times 2$  fits in a pattern for the contiguous problem of height 4, but not for the SPP



Since the contiguous bar relaxation is hard to solve because of the integer demand on the variables, we are interested in statements concerning its LP relaxation, [7]. Let  $z^{ver}$  denote the optimal value of the vertical bar relaxation and  $z_{LP}^{ver}$  that of the corresponding LP relaxation.

**Theorem 7.8** *The LP bound  $z_{LP}^{c-hor}$  of the horizontal contiguous bar relaxation is equal to the LP bound  $z_{LP}^{ver}$  of the vertical bar relaxation:*

$$z_{LP}^{c-hor} = z_{LP}^{ver}.$$

*Proof* Using a solution of the LP relaxation of the vertical bar relaxation, we can construct a solution of the LP relaxation of the horizontal contiguous bar relaxation. Such a construction in opposite direction is also possible. In each horizontal bar of a solution of the LP relaxation, we dissect each contained item into sufficiently small parts. These parts can be arranged such that parts belonging to the same piece get the same horizontal position. In this way vertical (one-dimensional) patterns are constructed. ■

Theorem 7.8 and inequality  $z_{LP}^{ver} \leq z_{LP}^{b-vert}$  yield the following dominance result concerning lower bounds of the SPP:

**Corollary 7.9** *The LP bound  $z_{LP}^{c-hor}$  of the horizontal contiguous bar relaxation is dominated by the LP bound  $z_{LP}^{b-vert}$  of the vertical binary bar relaxation:*

$$z_{LP}^{c-hor} \leq z_{LP}^{b-vert}.$$

Consequently, the usage of the horizontal contiguous bar relaxation to get a lower bound for the optimal value of the SPP is only meaningful if the integer problem is solved. Otherwise, the maximum of the LP bounds of the horizontal and the vertical binary bar relaxation provides a tighter bound.

## 7.3 Heuristics for the Strip Packing Problem

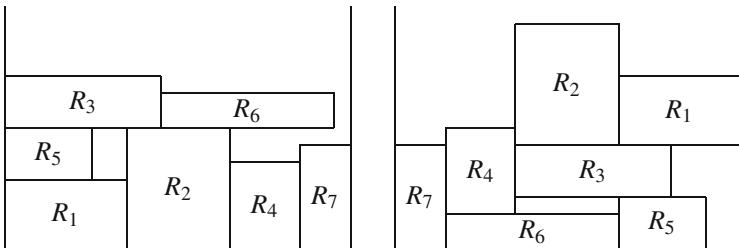
Since the SPP belongs to the class of NP-hard (optimization) problems, appropriate heuristics are of high interest. In the following we address some important of them while differentiating between offline and online scenarios.

### 7.3.1 Heuristics for the Offline Strip Packing Problem

An optimization problem is of type *offline* if all input data of the problem are known when the optimization process has to start. Otherwise, if not all input data are known, we say the problem is of type *online*. Respectively, an algorithm for solving offline-type problems is named *offline algorithm*, otherwise *online algorithm*.

**Table 7.1** Input data of Example 7.2

| $i$   | 1 | 2 | 3 | 4 | 5 | 6  | 7 |
|-------|---|---|---|---|---|----|---|
| $w_i$ | 7 | 6 | 9 | 4 | 5 | 10 | 3 |
| $h_i$ | 4 | 7 | 3 | 5 | 3 | 2  | 6 |



**Fig. 7.2** BL-packing for piece sequences  $\pi = (1, 2, 3, 4, 5, 6, 7)$  and  $\pi = (7, 6, 5, 4, 3, 2, 1)$

#### BL heuristic (Bottom-up Left-justified)

All pieces are packed according to the sequence defined by  $L$  at the lowest possible position (minimal  $y$ -coordinate) and, if not uniquely determined, at the left-most feasible place (minimal  $x$ -position).

Note that the definition of the BL heuristic includes the inspection of all holes occurring during the packing process. Note that a version of the BL heuristic which runs in  $O(m^2)$  time is proposed in [14].

*Example 7.2* Let a strip of width  $W = 20$  and a list  $L = (R_1, \dots, R_7)$  of pieces, defined in Table 7.1, be given. The packing pattern generated by the BL algorithm has height  $BL(L) = BL(R_1, \dots, R_7) = 10$ . For the reverse sequence  $R_7, \dots, R_1$  we obtain  $BL(R_7, \dots, R_1) = 13$ . Figure 7.2 shows the resulting patterns. ■

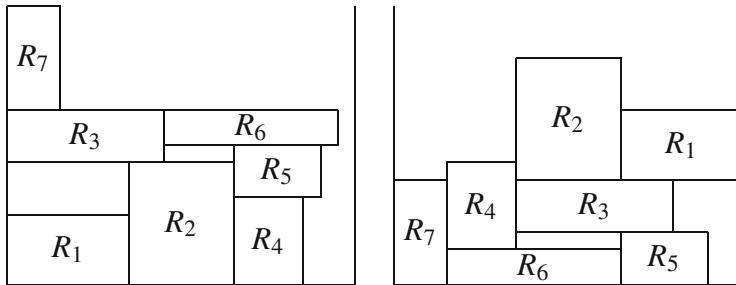
A simplification of the BL heuristic results if the holes are not inspected. In principle, each piece is placed at a sufficiently large height at an  $x$ -position from which a lowest  $y$ -position can be reached by shifting the piece downwards. (Among all mutual  $x$ -positions leading to a minimal  $y$ -position the smallest is taken.) This modification (no hole inspections) is called *BL<sub>0</sub> heuristic*. Its time consumption is proportional to  $m \log m$ .

**Definition 7.1** Let  $A_p = \{R_i(x_i, y_i) : i = 1, \dots, p\}$  be a feasible packing of pieces  $R_i, i = 1, \dots, p$ , with allocation points  $(x_i, y_i)$ . Furthermore, let

$$f_p(x) := \max\{0, \max\{y : (x, y) \in \bigcup_{i=1}^p R_i(x_i, y_i)\}\}, \quad x \in [0, W],$$

$$U(A_p) := \{(x, y) : 0 \leq x \leq W, 0 \leq y \leq f_p(x)\},$$

$$K(A_p) := \text{fr}(\text{cl}(\{(x, y) : 0 \leq x \leq W, 0 \leq y\} \setminus U(A_p)))$$



**Fig. 7.3**  $BL_0$ -patterns for sequences  $\pi = (1, 2, 3, 4, 5, 6, 7)$  and  $\pi = (7, 6, 5, 4, 3, 2, 1)$

where  $cl(\cdot)$  and  $fr(\cdot)$  denote the closure and frontier of a set, respectively. Function  $f_p$  describes the (maximal) height occupied by the packed pieces and  $U(A_p)$  the area below  $f_p$ . Then, point set  $K(A_p)$  is called  $BL_0$ -contour or skyline of packing  $A_p$ .

Note that, sometimes,  $f_p$  is said to be the skyline. As convention, let  $f_0(x) = 0$  for  $x \in [0, W]$ . Then, the  $BL_0$  heuristic can shortly be formulated as follows:

#### BL<sub>0</sub> heuristic

For  $p = 1, \dots, m$ : Compute the allocation point  $(x_p, y_p) \in K(A_{p-1})$  with minimal  $y_p$  and  $x_p$  so that  $y_p \geq f_{p-1}(x)$  holds for all  $x \in (x_p, x_p + w_p)$ .

*Example 7.3* Considering again Example 7.2, we obtain  $BL_0(L) = BL_0(R_1, \dots, R_7) = 16$  and  $BL_0(R_7, \dots, R_1) = 13$ . Figure 7.3 shows both resulting patterns. ■

Another fast algorithm for approximately solving the SPP is the well-known NFDH heuristic (*Next Fit Decreasing Height* [5]). It runs in  $O(m \log m)$  time. The main effort results by the initial sorting of pieces according to decreasing (in fact, non-increasing) heights which needs  $O(m \log m)$  time. The packing process itself requires  $O(m)$ , i.e., linear time.

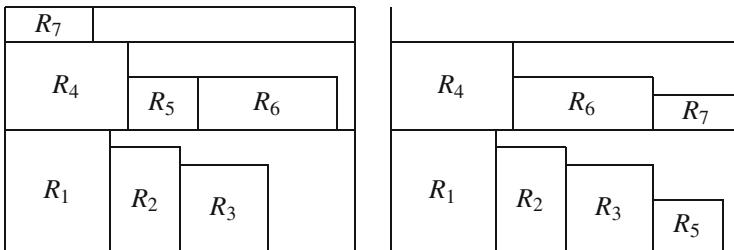
#### NFDH heuristic (Next Fit Decreasing Height)

At first sort all pieces according to non-increasing height which yields list  $L'$ . Then the pieces are placed successively in *layers* in the sequence determined by  $L'$ . The first (highest) piece is packed into the first layer (which is now *opened*) and it defines the height of the layer. Subsequent pieces are placed side by side in the open layer as long as the occupied width does not exceed strip width  $W$ . If the next piece, say  $i$ , does not fit, the layer is *closed* and a new one is opened whose height is  $h_i$ , and so on, until all pieces are packed.

A more expensive algorithm for approximately solving (offline) SPP is the FFDH heuristic (*First Fit Decreasing Height*). More precisely, it requires  $O(m \log m)$  time for the packing process. As performance assertions (Sect. 7.3.2) prove, the higher

**Table 7.2** Input data of Example 7.4

| $i$   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------|---|---|---|---|---|---|---|
| $w_i$ | 6 | 4 | 5 | 7 | 4 | 8 | 5 |
| $h_i$ | 7 | 6 | 5 | 5 | 3 | 3 | 2 |



**Fig. 7.4** Patterns obtained by the NFDH and the FFDH heuristic

computational effort in comparison to the NFDH heuristic is compensated by better results.

#### FFDH heuristic (First Fit Decreasing Height)

The FFDH heuristic works similar to the NFDH heuristic. As main difference, several layers can be open at the same time. In this case the lowest layer is chosen which can include the item to be packed next.

*Example 7.4* Let a strip of width  $W = 20$  and a list  $L = (R_1, \dots, R_7)$  of rectangles, defined in Table 7.2, be given. We obtain  $\text{NFDH}(L)=14$  and  $\text{FFDH}(L)=12$ . The resulting patterns are drawn in Fig. 7.4. ■

#### BFDH heuristic (Best Fit Decreasing Height)

The BFDH heuristic works similar to the FFDH heuristic. The difference consists in the choice of a layer: among all open layers that layer is chosen in which the next item to be packed fits best, i.e., the remaining unused width within the layer is minimal.

There are further heuristics known for the SPP, for instance, the *Reverse Fit* heuristic or the Algorithm of Steinberg (cf. Sect. 5.5.1). We refer the interested reader to [28] and [30], respectively, for a detailed description.

### 7.3.2 Performance Results

Within the relevant literature various performance results are known for heuristic algorithms of the SPP. We present here only some of them in order to demonstrate

some issues in this respect. For a given list  $L$  of rectangles let  $OPT(L)$  denote the minimal height needed to pack all items into the strip.

Initially, we show that a particular sorting of pieces can lead to worse results.

**Theorem 7.9 (Baker et al. [5])**

- (a) For every  $M > 0$  there exists a list  $L$  of rectangles sorted according to non-decreasing width with  $BL(L) > M \cdot OPT(L)$ .
- (b) For every  $M > 0$  there exists a list  $L$  of rectangles sorted according to non-increasing height with  $BL(L) > M \cdot OPT(L)$ .
- (c) For every  $\delta > 0$  there exists a list  $L$  of rectangles sorted according to non-increasing width with  $BL(L) > (3 - \delta) \cdot OPT(L)$ .

If all elements are squares, then there exists a list  $L$  with  $BL(L) > (2 - \delta) \cdot OPT(L)$ .

*Proof* To (a): We construct a sequence of instances which proves the statement. Let  $k \in \mathbb{N}$  with  $k \geq 2$ . We define

$$r_i := \max\{r \in \mathbb{Z} : i \equiv 0 \pmod{k^r}\}, \quad i = 1, 2, \dots$$

For example, for  $k = 4$  we have  $r_i = 0$  if 4 is not a divisor of  $i$ ,  $r_i = 1$  if 4 is a divisor of  $i$  but 16 not, and  $r_i = 2$  if 16 is a divisor of  $i$  but 64 not, etc.

Let  $W := k^k$  and  $0 < \varepsilon \ll 1$ . Then the following rectangles  $R_i = w_i \times h_i$  define list  $L$ :

| row      | $w_i :=$  | $h_i :=$                                   | $i =$                   |
|----------|-----------|--|-------------------------|
| 1        | 1         | $1 - r_1 \varepsilon$                      | $1, \dots, m_1 := k^k,$ |
| 2        | 1         | $1, m_1 + 1, \dots, m_2 := m_1 + k^{k-1},$ |                         |
| 3        | $k$       | $1, m_2 + 1, \dots, m_3 := m_2 + k^{k-2},$ |                         |
| 4        | $k^2$     | $1, m_3 + 1, \dots, m_4 := m_3 + k^{k-3},$ |                         |
| $\vdots$ | $\vdots$  | $\vdots$                                   | $\vdots$                |
| $(k+1)$  | $k^{k-1}$ | 1  | $m_k + 1.$              |

All rectangles are placed in the defined order. Then the BL heuristic yields a solution with  $BL(L) = k + 1$ . Figure 7.5 shows the resulting packing for  $k = 3$ .

On the other hand, since  $1 \cdot k^{k-1} + k \cdot k^{k-2} + \dots + k^{k-1} \cdot 1 = k^k$ , all rectangles  $R_i$  with  $i > m_1$  can be placed within a row of height 1. Hence,  $OPT(L) = 2$ .

A proof of (b) can be found in [5].

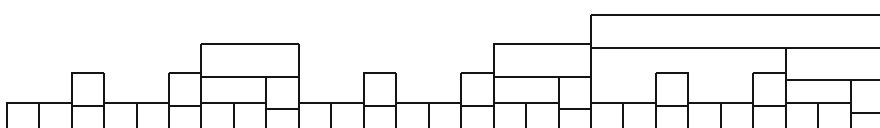


Fig. 7.5 To the proof of Theorem 7.9 (a), pattern obtained for  $k = 3$

To (c): The assertion is proved by the following example. Let  $k \in \mathbb{N}$  with  $k \geq 2$ , even, and let  $0 < \varepsilon < 8/(k^4 + 2k^2)$ . Moreover, pieces (squares)  $R_i$  with

$$\begin{aligned} w_i = h_i &= 2 - i\varepsilon, \quad i = 1, \dots, k^2/2, \\ w_i = h_i &= 1, \quad \quad \quad i = k^2/2 + 1, \dots, k^2/2 + k^2(k-2), \end{aligned}$$

form the list  $\mathcal{L} = (R_1, \dots, R_m)$  with  $m = k^2/2 + k^2(k-2)$ . The strip width is  $W = k^2$ . It is easy to see, that the first  $k^2/2$  pieces can be placed into a strip of height 2. Because of

$$\sum_{i=1}^{k^2/2} (2 - i\varepsilon) = k^2 - \frac{k^2}{8}(k^2 + 2)\varepsilon > k^2 - 1$$

it is not possible to additionally pack a unit square into that strip. Since the  $k^2(k-2)$  unit squares can be placed into  $k-2$  strips of height 1, we get the estimation  $\text{OPT}(\mathcal{L}) \leq k$ . Applying the BL heuristic, we obtain a packing of the form depicted in Fig. 7.6 for  $k = 4$ .

Within the second row  $k^2/2+1$  unit squares are placed; within the third  $k^2/2+2$ , and so on. Let  $r$  denote the number of rows needed to pack all  $k^2(k-2)$  unit squares. Then we have  $r > 2k - 8$  since

$$\sum_{i=1}^{2k-8} \left( \frac{k^2}{2} + i \right) = \frac{k^2}{2}(2k-8) + \frac{2k-8}{2}(2k-7) < k^3 - 2k^2$$

for  $k \geq 2$ . Hence,  $\text{BL}(\mathcal{L}) \geq 2k - 6$  and therefore

$$\frac{\text{BL}(\mathcal{L})}{\text{OPT}(\mathcal{L})} \geq \frac{2k-6}{k} = 2 - \frac{6}{k}.$$

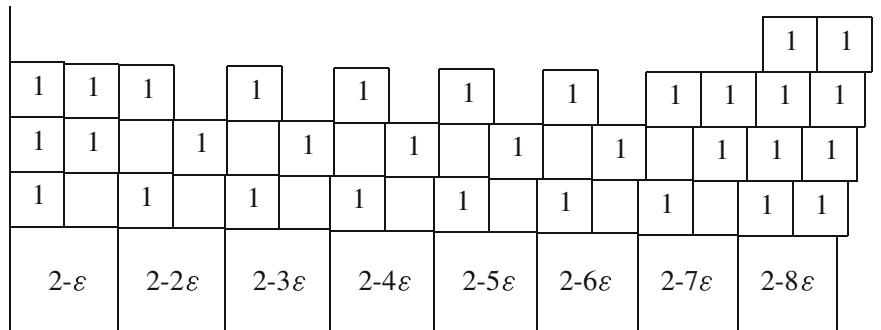


Fig. 7.6 To the proof of Theorem 7.9 (c), pattern obtained for  $k = 4$

For sufficiently large  $k$ , more precisely, for  $k > 6/\delta$ , the statement holds.

With  $h_{m+1} := k + 1$  and  $w_{m+1} := k^2 - \sum_{i=1}^{k^2/2} (2 - i\varepsilon) < 1$  we define a further piece  $R_{m+1}$ . Then piece  $R_{m+1}$  has a sufficiently small width and  $\text{OPT}(\mathcal{L}') \leq k + 1$  follows. On the other hand, this rectangle is placed in the topmost row by the BL heuristic so that  $\text{BL}(\mathcal{L}') \geq 3k - 6$  results. ■

Besides these negative results, we also know assertions which give *upper bounds* on the ratio between the height provided by the heuristic solution and the minimal one. In this sense, these are *positive* results.

### Theorem 7.11 (Baker et al. [5])

- (a) Let list  $\mathcal{L}$  of rectangles be sorted according to non-increasing height, then we have  $\text{BL}(\mathcal{L}) \leq 3 \cdot \text{OPT}(\mathcal{L})$ .

If all elements of  $\mathcal{L}$  are quadratic, then we have  $\text{BL}(\mathcal{L}) \leq 2 \cdot \text{OPT}(\mathcal{L})$ .

- (b) There exist lists  $\mathcal{L} = (R_1, \dots, R_m)$  of squares with

$$\min_{\pi \in \Pi} \frac{\text{BL}_\pi(\mathcal{L})}{\text{OPT}(\mathcal{L})} > \frac{12}{11 + \varepsilon} \quad \text{for } \varepsilon > 0,$$

where  $\Pi = \Pi(1, \dots, m)$  denotes the set of permutations and  $\text{BL}_\pi(\mathcal{L})$  the height resulting from  $\pi$ .

Here we present only the main idea of the proof.

**Sketch of proof** To (a): The region  $W \times \text{BL}(\mathcal{L})$  occupied by the BL heuristic, is partitioned into two parts. To this end, let  $h_0 = h_i$  denote the height of that piece  $R_i(x_i, y_i)$  having largest height for which  $y_i + h_i = \text{BL}(\mathcal{L})$  holds, i.e.,  $h_0 := \max\{h_i : y_i + h_i = \text{BL}(\mathcal{L})\}$ . Then, the used region is split into  $A := \{(x, y) : 0 \leq x \leq W, 0 \leq y \leq h^*\}$  with  $h^* = \text{BL}(\mathcal{L}) - h_0$  and  $\bar{A} := \{(x, y) : 0 \leq x \leq W, h^* \leq y \leq \text{BL}(\mathcal{L})\}$ .

It can be shown that in  $A$  at least half of its area is covered by placed pieces. Thereby, the assumed sorting of pieces is essentially exploited. In this case, we can conclude  $\text{OPT}(\mathcal{L}) \geq \max\{h_0, h^*/2\}$ .

If  $h_0 > \frac{h^*}{2}$ , then  $\frac{\text{BL}(\mathcal{L})}{\text{OPT}(\mathcal{L})} \leq \frac{h_0 + h^*}{h_0} < \frac{h_0 + 2h_0}{h_0} = 3$ .

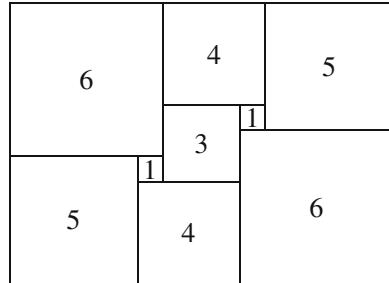
Otherwise, if  $h_0 \leq \frac{h^*}{2}$ , then we have  $\frac{\text{BL}(\mathcal{L})}{\text{OPT}(\mathcal{L})} \leq \frac{h^*/2 + h^*}{h^*/2} = 3$ .

If all pieces are quadratic, then region  $A$  can be split into  $\underline{A} := \{(x, y) : 0 \leq x \leq W, 0 \leq y \leq h_0\}$  and  $A' := \{(x, y) : 0 \leq x \leq W, h_0 \leq y \leq h^*\}$ . Since  $A' \subseteq A$ , at least half of the area of  $A'$  is covered with squares. Since the squares are sorted according to non-increasing size, in both strips,  $\underline{A}$  and  $\bar{A}$ , a total area of at least  $h_0 W$  is covered by squares. Thus, the second assertion follows.

To (b): The pattern drawn in Fig. 7.7 is optimal for a strip of width  $W = 15$  and  $H = 11$ . If we change the size of the  $3 \times 3$  square to  $(3 + \varepsilon_0) \times (3 + \varepsilon_0)$  and that of  $W$  to  $W = 15 + \varepsilon_0$ , then we obtain an example which proves the statement for  $\varepsilon > \varepsilon_0 > 0$ . ■

Theorem 7.11 states, in particular, that there exist instances of the SPP for which the BL heuristic cannot find an optimal packing. Concerning arbitrary rectangular pieces, in [13] even the lower bound  $5/4$  is shown.

**Fig. 7.7** To the proof of Theorem 7.11 (b)



Without loss of generality, we use a standardization of the input data in the following. Instead of  $w_i$ , we will consider  $w_i/W$  for all  $i$  and the normalized strip width  $W = 1$ . Moreover, we assume  $\max_{i \in I} h_i = 1$ .

**Theorem 7.12 (Coffman et al. [15])** *For every list  $L$  of rectangles the following assertions hold:*

- (a)  $NFDH(L) \leq 3 \cdot OPT(L)$ .
- (b)  $NFDH(L) \leq 2 \cdot OPT(L) + 1$ .
- (c) *The factors 3 and 2 are best possible, respectively.*

*Proof* Because of  $\max_{i \in I} h_i = 1$  we have  $OPT(L) \geq 1$ , so that assertion (a) immediately follows from (b).

To (b) and (c): The NFDH heuristic yields a sequence  $B_1, \dots, B_t$  of *blocks* (or *layers*) with heights  $H_1, \dots, H_t$  (and width  $W$ ) where  $H_1 \geq \dots \geq H_t$  holds.

Let  $y_i$  and  $x_i$  denote the covered width and the width of the first piece in block  $B_i$ , respectively, then we have

$$y_i + x_{i+1} > 1, \quad i = 1, \dots, t-1.$$

Moreover, let  $A_i$  denote the *used area* in  $B_i$ . Then

$$A_i + A_{i+1} \geq y_i H_{i+1} + x_{i+1} H_{i+1} > H_{i+1}, \quad i = 1, \dots, t-1.$$

Using  $A := \sum_{i=1}^t A_i = \sum_{j=1}^m w_j h_j$ , we obtain

$$NFDH(L) = \sum_{i=1}^t H_i \leq H_1 + \sum_{i=1}^{t-1} A_i + \sum_{i=2}^t A_i \leq H_1 + 2A \leq 1 + 2OPT(L).$$

Factor 2 is determined by the following example: let  $4m$  rectangles  $R_i$  with  $h_i = 1$  and  $w_i = 1/2$  for  $i = 1, \dots, 2m$  and  $w_i = \varepsilon = 1/(2m)$  for  $i = 2m+1, \dots, 4m$  be given. Then we have  $OPT(L) = m+1$ . On the other hand,  $NFDH(L) = 2m$  for list  $L = (R_1, R_{2m+1}, R_2, R_{2m+2}, \dots)$ . ■

As easily can be seen in the proof, if the normalization  $\max_{i \in I} h_i = 1$  is not used, then the statement in (b) results to  $NFDH(L) \leq 2 \cdot OPT(L) + \max_{i \in I} h_i$ .

We present the following performance results without proofs since they are extensively space consuming, in general.

**Theorem 7.13 (Coffman et al. [15])** *The following statements hold for every list  $L$  of rectangles:*

- (a)  $FFDH(L) \leq 1.7 \cdot OPT(L) + 1$ .
- (b) *Let  $w_i \leq \frac{1}{p}$  for  $i = 1, \dots, m, p \in \mathbb{N}$ , then  $FFDH(L) \leq (1 + \frac{1}{p}) \cdot OPT(L) + 1$ .*
- (c) *The factors are best possible.*
- (d) *If  $L$  contains only squares, then  $FFDH(L) \leq 1.5 \cdot OPT(L) + 1$ .*

A related result will be addressed in Theorem 7.20 (page 216). Similar statements hold for the BFDH heuristic Worst-case performance results are known as well, see [19].

### 7.3.3 Shelf Algorithms for Online Problems

In case of an *online* problem a sorting prior to the optimization process is not possible. Each rectangular piece has to be placed according to the given sequence. Frequently it is formulated as *each piece has to be packed immediately when it arrives*. We do not address here modifications (although easy to derive) of the subsequent approaches which result if buffering of  $k$  pieces ( $k \ll m$ ) is possible from which a most suitable item is chosen to be placed next.

When concerning online problems a large (or even huge) number of rectangles has to be packed, or this scenario can be seen as a continuous process in which the pieces become available successively.

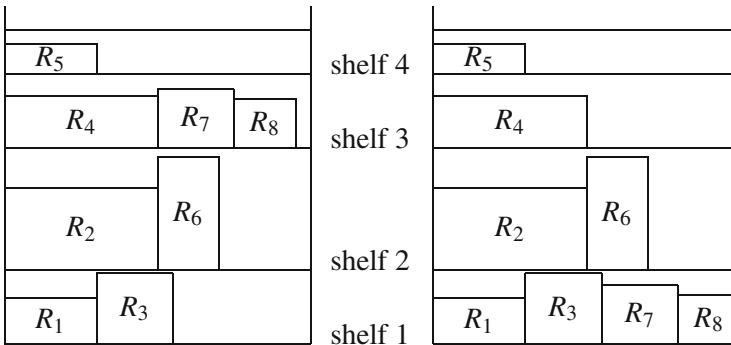
The following example shows that the *Next Fit* (NF) heuristic can produce arbitrarily bad patterns. The NF heuristic is simply the NFDH heuristic without initial sorting.

*Example 7.5* Let  $W = 1, k \in \mathbb{N}$  with  $k \geq 3, \varepsilon := 1/k$  be given. List  $L_k = (R_1, S_1, \dots, R_k, S_k)$  consists of  $2k$  rectangles where  $R_i = \varepsilon \times 1$  and  $S_i = 1 \times \varepsilon$ ,  $i = 1, \dots, k$ . As to show in Exercise 7.1,  $NF(L_k) \geq \frac{k+1}{2} \cdot OPT(L_k)$  holds for this example. ■

In order to get algorithms which pack all pieces in a sufficiently good manner without sorting them, so-called *shelf algorithms* are applied, [4]. This notation is based on the filling of a bookshelf. Within a shelf algorithm only shelves of predefined heights are used to pack the pieces. That means, piece  $i$  is put into a shelf with smallest height not smaller than  $h_i$ , if it fits, or a new shelf of that height is opened. In order to be able to estimate the resulting unused area, we introduce a parameter  $r \in (0, 1)$ . If  $h_{max}$  is the maximal piece height which can occur (or an upper bound of it), then the shelf heights are defined by  $r^k \cdot h_{max}, k = 0, 1, \dots$ . For that reason, the approach is also known as *harmonic* (shelf) heuristics.

**Table 7.3** Input data of Example 7.6

| $i$   | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
|-------|----|----|----|----|----|----|----|----|
| $w_i$ | 30 | 50 | 25 | 50 | 30 | 20 | 25 | 20 |
| $h_i$ | 15 | 27 | 23 | 17 | 10 | 37 | 19 | 16 |
| $k$   | 2  | 1  | 2  | 2  | 3  | 1  | 2  | 2  |



**Fig. 7.8** NFS and FFS patterns

Hence, a rectangle of height  $h_i$  has to be placed in a shelf with height  $r^k h_{\max}$  such that  $r^{k+1} < h_i/h_{\max} \leq r^k$  holds provided that a sufficiently large width is still available. If not, then the shelf is closed (in the NF heuristic) and a new one has to be opened having the same height  $r^k h_{\max}$ .

Similar to the *offline*-case, we can consider the *Next Fit*, the *First Fit*, and the *Best Fit* strategy. At first, we illustrate the approaches by means of an example.

*Example 7.6* Let  $W = 100$  and  $h_{\max} = 40$ . If we choose  $r = 0.6$ , we obtain the shelf heights 40, 24, 14.4, 8.64, ... According to the supposed *online* scenario the pieces defined in Table 7.3 have to be packed in the given sequence. Figure 7.8 shows the resulting patterns of the NFS (*Next Fit Shelf*) and the FFS (*First Fit Shelf*) heuristic. ■

The Next Fit Shelf heuristic [4], also called *harmonic NF* heuristic, can be formulated as follows:

#### NFS heuristic (Next Fit Shelf)

Choose parameter  $r$  with  $0 < r < 1$  and define shelf heights  $r^k h_{\max}$ ,  $k = 0, 1, 2, \dots$ . For  $i = 1, 2, \dots$ , pack rectangle  $R_i$  (having height  $h_i$ ) left-justified into the open shelf of height  $r^k h_{\max}$ , where  $r^{k+1} < h_i/h_{\max} \leq r^k$  holds, if there is sufficiently large space in horizontal direction. Otherwise, close this shelf, open a new one with height  $r^k h_{\max}$ , and pack  $R_i$  into the new shelf.

The First Fit Shelf heuristic, also called *harmonic FF* heuristic, works similar to the NFS heuristic, but with only open shelves.

**FFS heuristic (First Fit Shelf)**

Choose parameter  $r$  with  $0 < r < 1$  and define shelf heights  $r^k h_{\max}$ ,  $k = 0, 1, 2, \dots$ . For  $i = 1, 2, \dots$ , pack rectangle  $R_i$  (having height  $h_i$ ) left-justified into the bottom-most open shelf of height  $r^k h_{\max}$ , where  $r^{k+1} < h_i/h_{\max} \leq r^k$  holds, in which it fits. Otherwise, open a new shelf with height  $r^k h_{\max}$  and pack  $R_i$  into the new shelf.

Obviously, a *Best Fit Shelf* (BFS) heuristic can be formulated very similar to the FFS heuristic. Among others, the following performance results are known.

**Theorem 7.14 (Baker and Schwarz [4])** *For  $r$  with  $0 < r < 1$  and every list  $L$  of rectangles whose heights are not larger than  $h_{\max}$ , we have*

$$NFS_r(L) < \frac{2}{r} OPT(L) + \frac{h_{\max}}{r(1-r)}.$$

*The asymptotic performance ratio  $2/r$  is tight.*

It is remarkable that positive statements are obtained for shelf algorithms without any initial sorting of pieces. To simplify the proofs, let  $H_o$  denote the total height of *open* and  $H_c$  that of the *closed* shelves, respectively.

*Proof* First of all, we show that  $H_o < \frac{h_{\max}}{r(1-r)}$  holds. Let  $h$  be the height of an open shelf having largest height. Then the height  $h_i$  of each piece placed within this shelf is larger than  $rh$ . Hence,  $h_{\max} > rh$ . Moreover,

$$H_o < h \sum_{k=0}^{\infty} r^k = \frac{h}{1-r} < \frac{h_{\max}}{r(1-r)}$$

due to the sum of a geometric series.

Next, we show  $H_c < \frac{2}{r} OPT(L)$ . For each closed shelf  $S$  with height  $h_S$  there exists another open shelf  $S'$  with the same height. Due to the algorithm, all pieces placed within shelves  $S$  and  $S'$  have height larger than  $rh_S$ . Moreover, the width of the first piece in  $S'$  is larger than the remaining unused width in  $S$ . Consequently, the total area of rectangles placed into shelves  $S$  and  $S'$  is greater than  $rh_S W$ . If we now form pairs of shelves having the same height, including the open shelf if the number of closed ones is odd, we obtain that the total area of all placed pieces is larger than  $\frac{1}{2}rH_c W$ . Therefore,  $\frac{1}{2}rH_c < OPT(L)$  follows.

It remains to show that factor  $r/2$  is best-possible. Without loss of generality, we assume  $W = 1$  and consider the list  $L = (R_1, S_1, R_2, S_2, \dots, R_n, S_n)$  of rectangles with  $n > 0$  where each rectangle  $R_i$  has width  $\frac{1}{2} - \Delta$  with some  $\Delta$  in  $(0, \frac{1}{3n})$ . Every rectangle  $S_i$  has width  $3\Delta$ , and all  $2n$  rectangles have height  $r + \varepsilon$  with  $0 < \varepsilon < 1 - r$ . The NFS heuristic packs the rectangles pairwise into shelves of height 1. Therefore,  $NFS(L) = n - 1 + r + \varepsilon$  results.

However, an optimal packing requires a height not larger than that obtained using the sequence  $R_1, R_2, \dots, R_n, S_1, \dots, S_n$  which is  $(\lceil \frac{n}{2} \rceil + 1)(r + \varepsilon)$ .

For arbitrary positive constants  $\alpha$  and  $\beta$  with  $\alpha < \frac{2}{r}$  there exists a sufficiently large  $n$  with  $n - 1 + r + \varepsilon > \alpha(\lceil \frac{n}{2} \rceil + 1)(r + \varepsilon) + \beta(r + \varepsilon)$ . ■

**Corollary 7.15** *For  $r$  with  $0 < r < 1$  and every list  $L$  of rectangles, the inequality*

$$NFS_r(L) < \left( \frac{2}{r} + \frac{1}{r(1-r)} \right) OPT(L)$$

*is satisfied. Moreover, the asymptotic factor is tight.*

The proof of the corollary is subject of Exercise 7.6.

It is remarkable that the asymptotic performance of the  $NFS_r$  heuristic for  $r \rightarrow 1$  becomes conform with that of the NFDH heuristic for the *offline* case although no sorting is required. However, the absolute performance ratio is worsening for increasing  $r$ . An optimal value  $r^*$  has to be determined in Exercise 7.7. It is already given in the following corollary.

**Corollary 7.16** *The best absolute performance ratio of the  $NFS_r$  heuristic is obtained for  $r^* = (3 - \sqrt{3})/2 \approx 0.634$ . Then, we have  $NFS_r(L) \leq 7.464 \cdot OPT(L)$  for any list  $L$  of rectangles.*

Concerning the FFS heuristic, we have a similar statement with asymptotic performance factor  $1.7/r$ :

**Theorem 7.15 (Baker and Schwarz [4])** *For  $r$  with  $0 < r < 1$  and every list  $L$  of rectangles with heights not exceeding  $h_{max}$ ,*

$$FFS_r(L) < \frac{1.7}{r} OPT(L) + \frac{h_{max}}{r(1-r)}$$

*holds. Moreover, the asymptotic factor  $1.7/r$  is tight.*

A proof of Theorem 7.15 can be found in [4]. A related statement is addressed in Theorem 7.20. Corollaries similar to 7.15 and 7.16 can be drawn for the FFS<sub>r</sub> heuristic, too.

Besides the consideration of the worst-case behavior, i.e., of performance statements guaranteeing results obtainable with certainty, also statements of kind *what is best possible* are of interest. A lower bound for *online* algorithms is presented in [21] for the one-dimensional case (i.e.  $h_i = 1 \forall i$ ).

**Theorem 7.16 (Liang [21])** *For each online algorithm  $A$ , we have*

$$\lim_{|L| \rightarrow \infty} \sup \frac{A(L)}{OPT(L)} \geq 1.53\dots$$

The improved lower bound 1.540 is given in [31] (see also in [16]).

## 7.4 Local Search and Metaheuristics

In order to obtain good (near-optimal) solutions for NP-hard optimization problems and for problems with a large number of local extreme points, heuristic approaches have to be applied which cover a large portion of the solution (or search) space. In the following, let  $x$  denote a feasible solution with objective value  $f(x)$  of a certain integer minimization problem  $P$ .

In case of a very large set of feasible solutions and the absence of tight *performance bounds* for the chosen heuristic, it is hard to evaluate the quality of the solution obtained by the heuristic, in general.

To overcome this unsatisfactory situation, frequently methods of *Local Search* and *Metaheuristics* are applied which aim to find a better solution by repeatedly using one or several (fast) heuristics.

### 7.4.1 Local Search

We briefly describe some of the methods known in literature (see for example [2]) by means of the SPP. Let  $\pi = (\pi_1, \dots, \pi_m) \in \Pi(1, \dots, m)$  be an arbitrarily chosen permutation.  $BL(\pi)$  denotes the height obtained using the BL heuristic for the list  $R_{\pi_1}, \dots, R_{\pi_m}$  of rectangles. Furthermore, let  $U(\pi)$  represent a set of *neighboring* permutations, that means, a *neighborhood* of  $\pi$ :

$$U(\pi) := \{\tilde{\pi} \in \Pi(1, \dots, m) : d(\tilde{\pi}, \pi) \leq \varrho\}$$

where  $d(., .)$  represents a distance (function) and  $\varrho$  a predefined parameter.

As an example of a neighborhood and a distance function within the set of all permutations, we use the minimal number of interchanges of two elements needed to get another permutation  $\tilde{\pi}$  from  $\pi$ . Obviously, in case of  $\varrho = 1$ , we have  $|U(\pi)| = m(m - 1)/2$ .

To obtain a *local solution* with respect to a particular neighborhood, the following search has to be performed:

#### Algorithm Local Search (LS)—general principle

- (1) Compute a starting point (feasible solution)  $x$ .
- (2) If a feasible solution  $\tilde{x} \in U(x)$  with  $f(\tilde{x}) < f(x)$  is found, then replace  $x := \tilde{x}$  and repeat (2). Otherwise, the current solution  $x$  is a local solution (with respect to the neighborhood): stop.

In case of the SPP using the BL heuristic the general principle leads to the following concretization:

#### **Algorithm Local Search for the SPP**

- (1) Choose a (random) permutation  $\pi$  and compute  $BL(\pi)$ .
- (2) If a permutation  $\tilde{\pi} \in U(\pi)$  with  $BL(\tilde{\pi}) < BL(\pi)$  exists, replace  $\pi := \tilde{\pi}$  and repeat (2). Otherwise,  $\pi$  yields a local solution with respect to the BL heuristic and the considered neighborhood.

### **7.4.2 Metaheuristics**

Since a local solution is not a global one, in general, strategies to jump from one local to another local solution are of interest. To this end, so-called *metaheuristics* are applied. Basic investigations can be found, for instance, in [1, 2], and [25]. For conceptional design of metaheuristics the following aspects have to be regarded:

- Starting solutions  $x$ : How to compute? How many?
- Neighborhood  $U(x)$ : Variable or fixed radius? Which distance function?
- New solution  $\tilde{x}$ : Are somewhat worse solutions accepted to overcome local solutions? Deterministic or random search?
- Search space: Are non-feasible solutions accepted temporarily?
- Assessment criterion: Different to the objective function?

In the following, we give some examples of metaheuristics.

#### **Algorithm Multi-Start Local Search (MLS)**

- (1) Compute a starting solution  $x^*$ .
- (2) Generate randomly another feasible solution  $x$  and improve it by LS.
- (3) If  $f(x) < f(x^*)$ , then replace  $x^* := x$ .
- (4) If some stopping criterion is met, then stop. Otherwise, continue with (2).

#### **Algorithm Iterated Local Search (ILS)**

- (1) Compute a starting solution  $x^*$ .
- (2) Generate another feasible solution  $x$  derived from  $x^*$  by random disturbance and improve it by LS.
- (3) If  $f(x) < f(x^*)$ , then replace  $x^* := x$ .
- (4) If some stopping criterion is met, then stop. Otherwise, continue with (2).

*Genetic algorithms (GA)* transfer the evolution process taking place in nature into a solution strategy. Within a genetic algorithm repeatedly feasible solutions are used to generate new ones by operations *crossover* and/or *mutation*.

### Genetic Algorithm (GA)—general principle

- (1) Compute a set  $P$  (*population*) of feasible solutions. Let  $x^*$  be a best one in  $P$ .
- (2) Repeat steps (2.1) and (2.2) until a set  $Q$  of new feasible solutions is obtained where  $|Q|$  has to be predefined, in general.
  - (2.1) Choose two or more elements of  $P$  and construct, based on them, one or several new solutions using the crossover operation.
  - (2.2) Choose an element of  $P$  and construct a new one from it by mutation.
- (3) If a feasible solution  $x \in Q$  with  $f(x) < f(x^*)$  is obtained, replace  $x^* := x$ .
- (4) Choose a subset  $\tilde{P}$  of  $P \cup Q$  and set  $P := \tilde{P}$  ( $|\tilde{P}|$  has to be predefined).
- (5) If a stopping criterion is reached, then stop. Otherwise, continue with (2).

The following rule defines a crossover operation usable for the SPP:

*Example 7.7* Let  $\pi, \sigma \in P$ . Choose randomly an index  $i \in \{1, \dots, m - 1\}$ . Then, the first  $i$  components of the new permutation  $\tilde{\pi}$  are defined by  $\pi_1, \dots, \pi_i$  in the sequence as they occur in  $\sigma$ . The remaining  $m - i$  elements are determined by  $\pi_{i+1}, \dots, \pi_m$  again according to their occurrence in  $\sigma$ . For instance:

$$\pi = (1, 2, 3, 4), \sigma = (4, 3, 2, 1), i = 2 \Rightarrow \tilde{\pi} = (2, 1, 4, 3). \quad \blacksquare$$

The interchange of two indices is an example of a mutation operator. An efficient realization of a genetic algorithm for the SPP can be found in [11].

*Simulated Annealing (SA)* copies the physical process of annealing and of reaching a stable state. Thereby, a particular problem-specific (temperature) function  $t$  serves for controlling the process.

### Algorithm Simulated Annealing (SA)

- (1) Determine a starting solution  $x^* := x$  and compute its temperature  $t$ .
- (2) Compute randomly a feasible solution  $\tilde{x}$  and define  $\Delta := f(\tilde{x}) - f(x)$ .
- (3) If  $\Delta < 0$ , then replace  $x := \tilde{x}$ , otherwise, set  $x := \tilde{x}$  with probability  $e^{-\Delta/t}$ .
- (4) If  $f(x) < f(x^*)$ , then replace  $x^* := x$ .
- (5) If a stopping criterion is reached, then stop.  
Otherwise, update  $t$  and continue with (2).

Considering the SPP, we can use, for example,  $f(x) := \text{BL}(x)$  with  $x \in \Pi(1, \dots, m)$  and  $t = t(x) := f(x)W/\sum_{i \in I} w_i h_i - 1$ .

*Tabu Search (TS)* is another method to overcome staying in a local solution. To this end, a set  $T$  (so-called *tabu-list*) of feasible solutions is of importance.

### Algorithm Tabu Search (TS)

- (1) Compute a starting solution  $x^* := x$  and set  $T := \emptyset$ .
- (2) Compute a best solution  $\tilde{x} \in U(x) \setminus (\{x\} \cup T)$  and replace  $x := \tilde{x}$ .
- (3) If  $f(x) < f(x^*)$ , then replace  $x^* := x$ .
- (4) If a stopping criterion is reached, then stop.  
Otherwise, update  $T$  and continue with (2).

Statements concerning the *asymptotic convergence* of metaheuristics can be found, for instance, in [2].

The *Sequential Value Correction* (SVC) method, [8, 24], is a further general principle to approximately solve NP-hard problems. Using a fast heuristic, for instance the  $BL_0$  heuristic for the SPP, (pseudo-profit) values (or the sequence) of the items of a first feasible solution are determined. From that, changed values (or a changed sequence) of the items are computed. Based on the new values the heuristic is applied again yielding, in general, another feasible solution. This process can be repeated until a stopping criterion is met. In difference to the simple *Monte Carlo Method*, where the next sequence (permutation) is chosen randomly, the SVC method regards in some sense the structure of an already known solution. An application of the SVC method to the SPP can be found in [9].

### Sequential Value Correction Method (SVC)

- (1) Compute a starting solution  $x^*$ .
- (2) Choose a sequence  $\pi$  of pieces to be packed.
- (3) Compute a feasible solution  $x = x(\pi)$  based on  $\pi$ .
- (4) If  $f(x) < f(x^*)$ , then replace  $x^* := x$ .
- (5) If a stopping criterion is reached, then stop. Otherwise, compute a new sequence  $\pi$  based on  $x$  and continue with (3).

Obviously, a number of variants can be derived. For instance, which (fast) heuristic should be applied, or several simultaneously? Another source of modifications belongs to the manner the new sequence is constructed. In case of 'convergence' (no better solution is found during a number of consecutive iterations) a restart can be meaningful.

The applicability of metaheuristics is strongly dependent on the considered problem, and the same holds true for the choice of relevant parameters. But, metaheuristics are in particular useful to approximately solve two-dimensional cutting or packing problems with non-regular pieces as, for instance, occurring in the textile industry.

## 7.5 A Branch-and-Bound Algorithm

Since the SPP belongs to the NP-hard problems, exact solution approaches mostly apply branch-and-bound techniques. A comparative study of exact methods can be found, for instance, in [6]. In this section, we describe a branch-and-bound algorithm for solving the SPP. Thereby, we emphasize the differences to the b&b algorithm for the 2OPP (Sect. 5.6). To simplify the description, we do not use the concept of potential allocation points whose application mutually can save computational effort.

As usual, let

$$R_i(x, y) := \{(r, s) \in \mathbb{R}^2 : x \leq r < x + w_i, y \leq s < y + h_i\}$$

represent the region covered by rectangle  $R_i$  in case  $R_i$  has allocation point  $(x, y)$ . Corresponding to the allocation points  $(x_i, y_i)$ ,  $i \in I$ , we define

$$A(\tilde{I}) := \{R_i(x_i, y_i) : i \in \tilde{I}\} \quad \text{for } \tilde{I} \subseteq I$$

to be the related (packing) pattern of a subset  $\tilde{I}$  of pieces. Pattern  $A(\tilde{I})$  is called *normalized* or *bottom-left justified* if every packed item is with its bottom and left edge in contact with another packed item or the bottom or left edge of the strip. It is easy to see that among all optimal patterns there always exists a normalized one. Therefore, a b&b algorithm to compute an optimal pattern can be designed to search only within the finite set of normalized patterns.

As in Sect. 5.6, we define

$$U(A(\tilde{I})) := \{(x, y) \in \mathbb{R}_+^2 : \exists i \in \tilde{I} \text{ with } x \leq x_i + w_i, y \leq y_i + h_i\}$$

as the *contour region* covered by  $A(\tilde{I})$  and

$$K(A(\tilde{I})) := \text{fr}(\text{cl}(\{(x, y) : 0 \leq x \leq W, 0 \leq y\} \setminus U(A(\tilde{I}))))$$

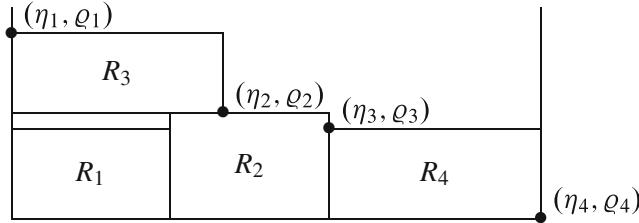
as corresponding *contour* (or *skyline*). As a convention, let  $A(\emptyset) = \emptyset$  and  $U(A(\emptyset)) = \emptyset$ . A contour region  $U(A)$  can be described in a unique manner using so-called *contour points*

$$\{(\eta_i, \varrho_i) : i = 1, \dots, n + 1\} \quad \text{with} \quad n = n(A) \geq 1$$

where  $0 = \eta_1 < \eta_2 < \dots < \eta_n < \eta_{n+1} = W$  and  $\varrho_1 > \varrho_2 > \dots > \varrho_n \geq \varrho_{n+1} = 0$ . Hence, we have  $U(A) = \bigcup_{i=1}^n \{(x, y) \in \mathbb{R}_+^2 : x \leq \eta_{i+1}, y \leq \varrho_i\}$ .

Figure 7.9 illustrates that the contour points are potential allocation points for further pieces to be placed.

The *contour approach* (similar to Sect. 5.6.1) constructs sequences of normalized monotone patterns allocating successively more pieces at one of the contour points.



**Fig. 7.9** Normalized monotone packing pattern used in the contour approach

#### Branch-and-Bound algorithm for the SPP

(1) *Initialization*

Initialize contour  $K_0$  of the empty strip and the branching tree  $I_0 := \emptyset$ .

Let  $k := 1$ .

(2) *Branching with respect to the allocation points*

(After packing  $k - 1$  pieces an allocation point for the  $k$ -th piece is chosen.)

If all allocation points of contour  $K_{k-1}$  have been considered, then *back track* :  $k := k - 1$ , if  $k = 0$ , then stop, otherwise go to (3).

Otherwise, choose an allocation point  $(\eta_k, Q_k)$  of  $K_{k-1}$  not considered so far for the  $k$ -th piece.

(3) *Branching with respect to pieces*

(After fixing the next allocation point, a suitable piece has to be selected.)

If all pieces of  $I \setminus I_{k-1}$  are taken as the  $k$ -th packed piece for allocation point  $(\eta_k, Q_k)$ , then *back-track* : go to (2).

Otherwise, choose a piece  $i_k \in I \setminus I_{k-1}$  which can feasibly be packed and has not been considered so far and allocate it at  $(\eta_k, Q_k)$ : set  $I_k := I_{k-1} \cup \{i_k\}$ ,  $A_k(I_k) := A_{k-1}(I_{k-1}) \cup \{R_{i_k}(\eta_k, Q_k)\}$ .

If a better solution is found, save it.

If  $k = m$ , then go to (3).

Compute the new contour  $K_k$  and set  $k := k + 1$ .

(4) *Bounding, equivalence and dominance tests*

If the current subproblem is fathomed by a bounding, equivalence, or dominance test (that means, no better solution can be obtained in this subproblem than that already known), then set  $k := k - 1$  and go to (3). Otherwise, continue with (2).

**Fig. 7.10** Branch-and-bound algorithm for the SPP

Within the b&b algorithm the definition of subproblems is oriented on the choice of a contour point and the piece to be allocated there. Appropriate bounding techniques and dominance considerations are addressed below.

For the sake of a simpler description of the algorithm (see Fig. 7.10), we design again the *Last In First Out* (LIFO) strategy (depth first search). A transfer of the algorithm to apply the *Best Bound Search* strategy is straightforward. Let  $k$  denote the *branching depth (stage)* which is equal to the number of packed pieces. The index set  $I_k$  represents the pieces already packed.

Within the b&b algorithm we can apply different lower bounds for the needed strip height  $H$ . Their performance quality is, in general, proportional to their computational expense.

Let again  $A_k$  represent a pattern of rectangles  $R_i, i \in I_k = \{i_1, \dots, i_k\}$ . As usual, we may assume that all input data are integer so that all allocation points are integer as well. Furthermore, we denote the area of  $U(A_k)$  by  $\alpha_k = \sum_{j=1}^n (\eta_{j+1} - \eta_j) \varrho_j$  and the least area consumption of pieces  $i \in I \setminus I_k$  not yet packed by  $\beta_k := \sum_{i \notin I_k} w_i h_i$ . A simple lower bound is the *material* or *area bound*  $lb_A$  obtained as follows:

$$lb_A(A_k) := \max \{\varrho_1, \varrho_n + \max\{h_i : i \in I \setminus I_k\}, \lceil(\alpha_k + \beta_k)/W\rceil\},$$

where  $\lceil . \rceil$  denotes rounding up.

Considering the concept of the reduced set of potential allocation points in  $W$ -direction, we can possibly assign enlarged  $\eta$ -values to pattern  $A_k$  according to

$$\tilde{\eta}_j := W - p_W^{red}(W - \eta_j)$$

and therefore, an enlarged area  $\tilde{\alpha}_k$ . Obviously, the same principle can be exploited for the definition of allocation points  $(x_i, y_i)$ . Because of the possibly increased area  $\tilde{\alpha}_k$  of  $A_k$  and a more sophisticated consideration of piece heights, we obtain the lower bound  $\tilde{lb}_A$  which dominates  $lb_A$ :

$$\tilde{lb}_A(A_k) := \max \left\{ \max_{j=1, \dots, n} \{\varrho_j + \max_{i \in I \setminus I_k} \{h_i : w_i > W - \eta_{j+1}\}\}, \lceil(\tilde{\alpha}_k + \beta_k)/W\rceil \right\}.$$

Here we use the convention that  $\max\{h_i : i \in \tilde{I}\} = 0$  holds if  $\tilde{I}$  is empty.

*Example 7.8* Six squares of edge length  $w_1 = w_2 = 6$ ,  $w_3 = w_4 = 5$ , and  $w_5 = w_6 = 4$  have to be packed into a strip of width  $W = 15$  and minimal height  $H$ . Thus, the reduced set of potential allocation points in  $W$ -direction results to  $\tilde{S}(w, 15) = \{0, 4, 5, 6, 9, 10, 11\}$ . If we consider the arrangement  $A_2 = \{R_1(0, 0), R_2(6, 0)\}$ , then we obtain  $lb_A(A_2) = \lceil(2(36 + 25 + 16)/15\rceil = 11$  and  $\tilde{lb}_A(A_2) = \lceil(154 + 18)/15\rceil = 12$ . ■

A more expensive, but in general tighter bound can be obtained if we relax the SPP according to the horizontal bar relaxation. Considering the SPP instance  $E = (m, W, w, h)$  the resulting instance  $E' = (m, L, \ell, b)$  of the one-dimensional cutting stock problem (1CSP) has the input data

$$L := W, \ell_i := w_i, b_i := h_i, i = 1, \dots, m.$$

The optimal value of the 1CSP constitutes the lower bound  $lb_{CSP}$  of the SPP. Consequently, the optimal value  $lb_{LP}$  of the continuous relaxation of the 1CSP is a lower bound of the SPP as well. Obviously, bound  $lb_{LP}$  can be obtained with less effort in comparison to  $lb_{CSP}$ , but, concerning the MIRUP conjecture, it yields a similarly tight bounding value.

The bar relaxation approach can also be applied to obtain a lower bound for subproblems. Let  $(\eta_r, \varrho_r)$ ,  $r = 1, \dots, n + 1$ , denote the contour-points of packing  $A_k$ . We define (one-dimensional) patterns  $a^{jr} \in \mathbb{Z}_+^m$  according to  $\sum_{i \in I} \ell_i a_i^{jr} \leq L - \eta_r$  and related variables  $x_{jr}$ ,  $j \in J_r$ ,  $r = 1, \dots, n$ . Note that patterns with  $r \geq 2$  pack pieces within the current strip height  $\varrho_1$  and therefore, do not effect the objective value, whereas patterns with  $r = 1$  lead to an increase of the used strip height. Consequently, the optimal value  $lb_{LP}(A_k)$  of the LP relaxation of the multi cutting stock problem provides a lower bound for the subproblem determined by packing  $A_k$ :

$$\begin{aligned} z = \min \quad & \varrho_1 + \sum_{j \in J_1} x_{j1} \quad \text{s.t.} \\ & \sum_{r=1}^n \sum_{j \in J_r} a_i^{jr} x_{jr} = \begin{cases} b_i, & i \in I \setminus I_k, \\ 0, & i \in I_k, \end{cases} \quad x_{jr} \geq 0, \quad j \in J_r, r \in \{1, \dots, n + 1\}. \end{aligned}$$

In order to save computational effort within a branch-and-bound approach it is favorable to apply equivalence and dominance considerations similar to Sect. 5.6.4.

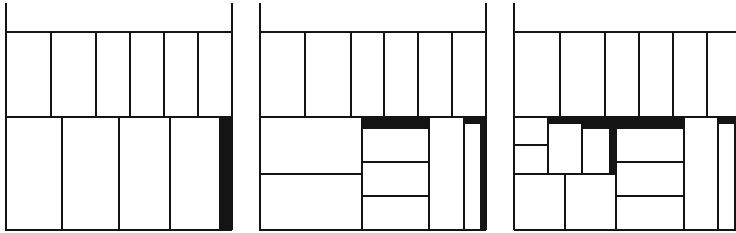
Besides the previously mentioned heuristics for the SPP, also a branch-and-bound algorithm can easily be modified to obtain approximate solutions. For instance, in each iteration step exactly one (most promising) allocation point is selected and, similarly, exactly one piece type (according to a predefined rule or randomly). In this way, after  $m$  iteration steps a feasible solution is constructed. Note that using a strategy like that allows to regard additional constraints which possibly cannot be handled in an ILP model.

Another modification of a b&b algorithm results if a time-dependent termination criterion is applied. That means, subproblem  $P_k$  is only considered if  $\bar{z} - lb(P_k) > \varepsilon(t)$  is satisfied where  $\bar{z}$  denotes the value of the current best solution,  $lb(P_k)$  is a lower bound of subproblem  $P_k$ , and  $\varepsilon(t)$  is a nonnegative time-dependent function which (exponentially) increases.

As experience shows, it seems to be advantageous, in general, to apply several heuristics (even several times if dependent on some random decisions) in order to obtain a near-optimal solution.

## 7.6 Guillotine Strip Packing

Besides the general case of orthogonal strip packing considered so far, we address here the scenario where only guillotine cuts (G-cuts) can be used. Problems of this kind arise, for instance, in the paper industry and in metallurgy. In Fig. 7.11  $k$ -stage patterns for  $k = 2, 3$ , and  $5$  are depicted. In difference to Chap. 6, we consider here only the *exact* case. Hence, trimming cuts are considered as cuts of the next stage. That means, the *non-exact* 2-stage guillotine cutting problem, considered in Sect. 6.3, is a particular 3-stage guillotine cutting problem.



**Fig. 7.11** 2-, 3-, and 5-stage guillotine cutting patterns

Before we turn to appropriate ILP models, we first analyze the asymptotic behavior in dependence of the stage-number  $k$ . Note that the heuristics NFDH, FFDH, BFDH, NFS, FFS, and BFS, described in Sect. 7.3.1, produce G-patterns.

### 7.6.1 Quality of Guillotine Patterns

To formulate an appropriate statement, we suppose that rectangular pieces  $R_i$ ,  $i \in I = \{1, \dots, m\}$ , with width  $w_i \in (0, 1]$  and height  $h_i \in (0, 1]$  have to be packed into a strip of width  $W = 1$ . We denote by  $G_k(E)$  the minimal strip height needed when applying *exact k-stage guillotine cutting* and by  $\text{OPT}(E)$  the optimal height of instance  $E$ .

For the case of 2-stage guillotine patterns the following negative result is known:

**Theorem 7.19 (Seiden and Woeginger [29])** *For every  $r \geq 1$  and every positive constant  $c$  there exists an instance  $E$  such that*

$$G_2(I) > r \cdot \text{OPT}(I) + c.$$

*Proof* We prove the theorem by constructing an appropriate instance  $E$ . Let  $t \in \mathbb{N}$  and let  $E$  be an instance with  $m = t^2$  rectangular pieces of width  $w_i = 1/t$  and height  $h_i \in [1 - \frac{1}{t}, 1]$  for all  $i \in I$  and  $h_i \neq h_j$  for  $i \neq j$ ,  $i, j \in I$ .

Obviously,  $\text{OPT}(E) \leq t$  since, in any case,  $t$  rectangles can be placed side by side within a shelf (horizontal strip) of height 1. On the other hand, because of the *exact case*, exactly one piece is assigned to a shelf of height  $h_i$  when applying 2-stage cutting (piece  $i$  itself) since all other have another height. Therefore,

$$G_2(E) = \sum_{i \in I} h_i \geq t^2 \cdot \left(1 - \frac{1}{t}\right) = t(t-1) \geq (t-1)\text{OPT}(E).$$

■

In order to formulate a statement concerning exact 3-stage cutting, we need to introduce the following sequence of numbers  $t_1, t_2, \dots$  (Salzer [27]):

$$t_1 := 2, \quad t_{i+1} := t_i(t_i - 1) + 1, \quad i = 1, 2, \dots$$

Using the Salzer-sequence, we define the number  $h_\infty$  as follows:

$$h_\infty := \sum_{i=1}^{\infty} \frac{1}{t_i - 1} = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \frac{1}{1806} + \dots \approx 1.69103.$$

The following theorem establishes a relationship to the Theorems 7.15 and 7.13.

**Theorem 7.20 (Seiden and Woeginger [29])** *For each  $\varepsilon > 0$  there exists a constant  $c(\varepsilon)$  such that for each instance  $E$*

$$G_3(E) \leq (h_\infty + \varepsilon) \cdot OPT(E) + c(\varepsilon)$$

holds. A corresponding 3-stage guillotine pattern can be computed in polynomial time. Factor  $h_\infty$  is best possible.

To prove the performance factor, we construct a 3-stage pattern using the *harmonic shelf algorithm*. Similar to the *online* algorithms in Sect. 7.3.3, the set of rectangles is partitioned according to their heights. Depending on the input parameter  $r \in (0, 1)$  the subsets are defined by

$$I_s := \{i \in I : r^{s+1} < h_i \leq r^s\}, \quad s = 0, 1, \dots$$

All pieces belonging to set  $I_s$  are placed side by side into shelves of height  $r^s$ . Since the width of a shelf is limited by  $W = 1$ , we obtain a one-dimensional *Bin Packing-Problem* (Chap. 3) for each  $s$ .

To approximately solve the 1BPPs the *Harmonic Bin Packing algorithm*, [17], is used. To this end, the interval  $W := [0, 1]$  is dissected into sub-intervals  $W_1, \dots, W_\kappa$  ( $\kappa \in \mathbb{N}$ , sufficiently large) with

$$W_j := \left( \frac{1}{j+1}, \frac{1}{j} \right], \quad j = 1, \dots, \kappa - 1, \quad W_\kappa := \left[ 0, \frac{1}{\kappa} \right].$$

The Harmonic Bin Packing algorithm defines a shelf (bin) for each interval  $W_j$ . Then, only such pieces with  $w_i \in W_j$  are packed into the shelf for  $W_j$  as long as the capacity 1 of the shelf is not exceeded. Otherwise, a new shelf for  $W_j$  is opened.

Due to construction, a 3-stage pattern is obviously obtained for each subset  $I_s$ , and therefore, for the whole instance, too. It is remarkable, that the patterns constructed in this way can be seen as *non-exact 2-stage guillotine patterns* according to the terminology of Gilmore and Gomory (Chap. 6).

We refer to [29] for a proof that a 3-stage pattern obtained by the Harmonic Shelf algorithm fulfills the proposed performance statement.

To verify that the value  $h_\infty$ , which is determined by the Harmonic Shelf algorithm, is best possible, we construct again an instance  $E$  for the worst case, i.e., a *worst-case example*. Similar to the proof of Theorem 7.19, we suppose that all piece widths  $w_i$  are different. For arbitrary but fixed  $\varepsilon \in (0, 1)$  and  $\alpha \in \mathbb{N}$  with  $\varepsilon\alpha > 7$ , we define the number  $d$  as the smallest index with

$$h_\infty - \frac{\varepsilon}{2} \leq \sum_{i=1}^d \frac{1}{t_i - 1}.$$

Furthermore, let  $\delta > 0$  be defined by

$$\delta := \frac{1}{d} \left(1 - \sum_{i=1}^d \frac{1}{t_i}\right).$$

As will be shown in Exercise 7.11, we have

$$\delta = \frac{1}{t_d(t_d - 1)d}. \quad (7.8)$$

Then, instance  $E$  contains  $d$  lists  $I_1, \dots, I_d$  which are defined as follows: let  $N_0 := 1$ . For  $p = 1, \dots, d$ , list  $I_p$  contains exactly  $n_p := \alpha(t_p - 1)N_{p-1}$  rectangular pieces of height  $h_i = H_p := \alpha/n_p$  and width  $w_i \in (1/t_p, 1/t_p + \delta)$  with  $w_i \neq w_j$  for  $i \neq j$ ,  $i, j \in I_p$ , where  $N_p := n_1 + \dots + n_p$  is used.

Now, we show that  $\text{OPT}(E) \leq \alpha$  holds. Because of  $w_i < \frac{1}{2} + \delta$  and  $h_i = 1$  for all  $i \in I_1$ , all these  $n_1 = \alpha$  pieces can be packed within a rectangle of height  $\alpha$  and width  $\delta + 1/t_1$  on top of each other. Due to the definition of  $H_p$ , we can pack  $(t_p - 1)N_{p-1}$  pieces of  $I_p$  on top of each other into a rectangle of height 1 and width  $\delta + 1/t_p$ . In this way, all  $n_p$  pieces of  $I_p$  are placed in a rectangle of height  $\alpha$  and width  $\delta + 1/t_p$ . Since

$$\sum_{p=1}^d \left(\frac{1}{t_p} + \delta\right) = d\delta + \sum_{p=1}^d \frac{1}{t_p} = 1$$

the total width needed is equal to 1. Thus,  $\text{OPT}(E) = \alpha$ . It is easy to see that the constructed pattern is a 4-stage one.

It remains to show which strip height a 3-stage pattern at least has to have. We show that  $G_3(E) \geq (\alpha - 1) \sum_{p=1}^d 1/t_p$  holds for  $E$ .

In any 3-stage pattern the strip is partitioned into shelves by horizontal G-cuts of the first stage. Within a shelf rectangular blocks are obtained by vertical G-cuts of the second stage. We assume that such a block cannot be further dissected by vertical G-cuts. Within a block several pieces (placed on top of each other) could appear if they have the same width. Then, these pieces can be obtained by horizontal G-cuts of the third stage. However, different piece widths would require vertical G-cuts of

the fourth stage. Due to construction of instance  $E$  (all widths are different) each block of a 3-stage pattern contains at most one piece.

A shelf is of type  $p$  if it contains at least one piece of  $I_p$ , but none of  $I_1, \dots, I_{p-1}$ . Let  $x_p$  denote the number of shelves of type  $p$ ,  $p = 1, \dots, d$ . Furthermore, let  $X_p := x_1 + \dots + x_p$ . Then we have  $x_p \leq n_p$  and  $X_p \leq N_p$  for all  $p$ . Moreover, it holds that all pieces of list  $I_p$  have to be placed into shelves of type  $1, \dots, p$  and each shelf can contain at most  $t_p - 1$  of such pieces. Therefore, we have

$$n_p \leq (x_p + X_{p-1}) \cdot (t_p - 1) \leq (x_p + N_{p-1}) \cdot (t_p - 1).$$

Using  $n_p := \alpha(t_p - 1)N_{p-1}$ , we obtain

$$x_p \geq (\alpha - 1) \cdot N_{p-1}.$$

Since each shelf of type  $p$  has height  $H_p = \frac{1}{(t_p - 1)N_{p-1}}$ , finally

$$\sum_{p=1}^d x_p H_p \geq \sum_{p=1}^d (\alpha - 1) N_{p-1} \frac{1}{(t_p - 1)N_{p-1}} = (\alpha - 1) \sum_{p=1}^d \frac{1}{(t_p - 1)}$$

follows. Altogether, supposing  $\alpha > 7/\varepsilon$ , we obtain

$$\frac{G_3(E)}{\text{OPT}(E)} > \frac{\alpha - 1}{\alpha + 1} \sum_{p=1}^d \frac{1}{(t_p - 1)} \geq \frac{\alpha - 1}{\alpha + 1} \left( h_\infty - \frac{\varepsilon}{2} \right) > h_\infty - \varepsilon.$$

This completes the proof. ■

**Theorem 7.21 (Seiden and Woeginger [29])** *For any  $\varepsilon > 0$  there exists a constant  $c(\varepsilon)$  such that for each instance  $E$*

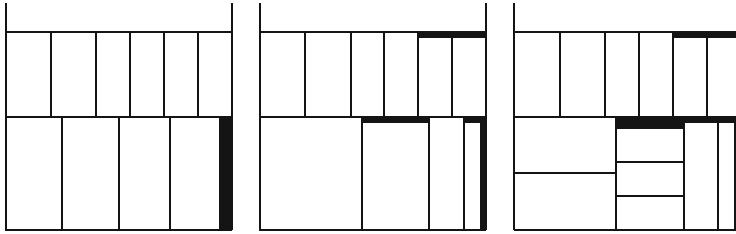
$$G_4(E) \leq (1 + \varepsilon) \cdot \text{OPT}(E) + c(\varepsilon).$$

holds. A corresponding 4-stage pattern can be obtained in polynomial time (polynomial in  $m$  and  $1/\varepsilon$ ).

As a consequence of Theorem 7.21, the computation of an optimal  $k$ -stage pattern for  $k > 4$  is not worthwhile for SPP instances with a very large number of pieces since (asymptotically) no better solutions can be obtained. This statement is not true if  $m$  is rather small.

### 7.6.2 ILP Models for k-Stage Guillotine Strip Packing

In the following, we model the *exact case* of the  $k$ -stage guillotine strip packing problem (GSPP) where  $b_i$  non-rotatable rectangular pieces of type  $R_i$ , having width



**Fig. 7.12** Exact 2-stage, non-exact 2-stage, and general 3-stage G-patterns

width  $w_i$  and height  $h_i$ , have to be placed into a strip of width  $W$  and minimal height for each  $i \in I = \{1, \dots, m\}$ . Let  $H_j$ ,  $j = 1, \dots, \bar{m}$ , denote the different heights of pieces and

$$I_j := \{i \in I : h_i = H_j\}, \quad j = 1, \dots, \bar{m},$$

corresponding index sets. Hence,  $I_j \neq \emptyset$  and  $\cup_{j \in J} I_j = I$  with  $J = \{1, \dots, \bar{m}\}$ .

Due to the definition of an exact 2-stage G-pattern as depicted in Fig. 7.12, only pieces of the same height are allowed to be placed within the same shelf. Consequently, the 2-stage GSPP decomposes into  $\bar{m}$  independent *one-dimensional bin packing-problems* (Chap. 3).

Without loss of generality, let all rectangular pieces of height  $H_j$  be indexed such that  $I_j = \{1, \dots, m_j\}$  for  $j \in \{1, \dots, \bar{m}\}$  where  $m_j := |I_j|$ . Moreover, let  $a^{jk} = (a_{1jk}, \dots, a_{m_jjk}) \in \mathbb{Z}_+^{m_j}$ ,  $k \in K_j$ , represent all possible shelf patterns of height  $H_j$ , that means, they are characterized by

$$\sum_{i \in I_j} w_i a_{ijk} \leq W \quad \text{and} \quad 0 \leq a_{ijk} \leq b_i, \quad i \in I_j.$$

Using variables  $x_{jk}$  to indicate the frequency pattern  $a^{jk}$  is used, we obtain the following ILP model for a single piece height  $H_j$  ( $j \in J$ ):

#### Model of the 2-sage GSPP, exact case

$$\begin{aligned} z_j &= \min \sum_{k \in K_j} x_{jk} \quad \text{s.t.} \\ &\sum_{k \in K_j} a_{ijk} x_{jk} = b_i \quad i \in I_j, \\ &x_{jk} \in \mathbb{Z}_+, \quad k \in K_j. \end{aligned}$$

Since the ILP model represents one of the 1BPP and the 1CSP, approaches to (approximately) solve it can be applied as described in Chaps. 3 and 4. Obviously, the optimal value  $z^{2-st}$  of the 2-stage GSPP is equal to the total sum of all  $z_j$ ,  $j \in J$ .

In the following, we consider a particular 3-stage strip packing problem which is named *non-exact 2-stage* according to the terminology of Gilmore and Gomory (Fig. 7.12). The horizontal G-cuts of the first stage partition the entire strip into *shelves* (segments) which are dissected by vertical G-cuts into rectangular *blocks*. Within a block at most a single piece can be contained and its width has to equal the block width while its height can be smaller than the shelf height. In this case, a further horizontal G-cut (third stage) is necessary, called *trimming cut*.

Now, without loss of generality, we assume  $h_1 \leq h_2 \leq \dots \leq h_m$  so that  $H_1 < \dots < H_{\bar{m}}$  follows. Moreover, according to the definition of  $I_j = \{i \in I : h_i = H_j\}$ , we also have that  $p \in I_j, q \in I_k, j < k$  implies  $p < q$ . Due to the non-exact case, we need to define index sets  $\bar{I}_j$  to characterize a pattern for a shelf of height  $H_j$ :

$$\bar{I}_j = \{1, \dots, \bar{m}_j\} := \bigcup_{t=1}^j I_t = \{i \in I : h_i \leq H_j\}, \quad j = 1, \dots, \bar{m}.$$

Similar to above, let  $a^{jk} = (a_{1jk}, \dots, a_{\bar{m}_j jk}) \in \mathbb{Z}_+^{\bar{m}_j}$ ,  $k \in K_j$ , represent all possible patterns belonging to shelf height  $H_j$ . They are characterized by

$$\sum_{i \in \bar{I}_j} w_i a_{ijk} \leq W \quad \text{und} \quad 0 \leq a_{ijk} \leq b_i, \quad i \in \bar{I}_j.$$

The only difference in comparison to the exact case consists in the fact that more pieces (of different heights) can be combined within a shelf. For that reason, the 3-stage SPP, and hence the non-exact 2-stage SPP, does not decompose into independent subproblems. Using variables  $x_{jk}$  indicating the frequency pattern  $a^{jk}$  is used, we obtain the following model:

### Model of the non-exact 2-stage SPP

$$z^{2-non} := \min \sum_{j \in J} H_j \sum_{k \in K_j} x_{jk} \quad \text{s.t.}$$

$$\sum_{j \in J} \sum_{k \in K_j} a_{ijk} x_{jk} = b_i, \quad i \in I,$$

$$x_{jk} \in \mathbb{Z}_+, \quad k \in K_j, \quad j \in J.$$

Considering the general 3-stage SPP, in each shelf a 2-stage pattern (with vertical G-cuts in the first stage) can be present. It is obvious, that the proposed solution strategy for the 2-stage SPP is applicable to the 3-stage SPP as well. The slave problems occurring during the column generation process can be solved as described in Chap. 6. The real difficulty for the general 3-stage SPP consists in the enlarged number of shelf heights since now, in principle, each non-negative integer combination of piece heights can determine a shelf height. This can lead to

a remarkable increase of the computational effort. Therefore, for practical purposes, a predefined subset of shelf heights should be used instead.

Clearly, if  $z^{3-st}$  denotes the optimal value of the (general) 3-stage case, then we have

$$z^{2-st} \geq z^{2-non} \geq z^{3-st}.$$

Depending on the real background when 2- or 3-stage cutting is applied, an investigation (estimation) of the gaps between the optimal values can be of interest.

## 7.7 Exercises

**Exercise 7.1** Prove Proposition 7.1:  $\sup_E \frac{H^*(E)}{\overline{lb}_h(E)} = \infty$ ,  $\sup_E \frac{H^*(E)}{\overline{lb}_A(E)} = \infty$ .

**Exercise 7.2** Let  $z_{LP}^{Kan}$  denote the optimal value of the continuous relaxation of the Kantorovich-type horizontal relaxation (7.4) of the 2SPP. Show, by means of an example, that  $z_{LP}^{Kan} < lb_0 = \max\{lb_h, lb_A\}$  is possible.

**Exercise 7.3** Compute the BL pattern (sketch) and  $BL(L)$  for the SPP instance with input data  $W = 9$  and

| $i$   | a | b | c | d | e | f | g | h |
|-------|---|---|---|---|---|---|---|---|
| $w_i$ | 6 | 2 | 3 | 4 | 2 | 1 | 1 | 3 |
| $h_i$ | 1 | 5 | 3 | 2 | 2 | 3 | 1 | 1 |

Which value is provided by the material bound? Does there exist a better pattern? If yes, which sequence of pieces (if any) allows to obtain it with the BL heuristic?

**Exercise 7.4** Compute NFDH( $L$ ), FFDH( $L$ ), and BFDH( $L$ ) for the SPP instance given in Exercise 7.3.

**Exercise 7.5** Compute the strip heights obtained by the BL, NFDH, and FFDH heuristic applied to the SPP instance with  $W = 20$  and

| $i$   | a | b | c | d | e | f | a | b | c | d | e | f |
|-------|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_i$ | 7 | 6 | 8 | 5 | 5 | 4 | 7 | 6 | 8 | 5 | 5 | 4 |
| $h_i$ | 9 | 5 | 4 | 4 | 2 | 2 | 9 | 5 | 4 | 4 | 2 | 2 |

**Exercise 7.6** Give a proof of Corollary 7.15 of Theorem 7.14 (page 206), and show, in particular, that the asymptotic bound is tight.

**Exercise 7.7** Prove Corollary 7.16 of Theorem 7.14 (page 206).

**Exercise 7.8** Formulate the algorithms Multi-Start Local Search (MLS) and Iterated Local Search (ILS) for the SPP.

**Exercise 7.9** Formulate the algorithm Tabu Search (TS) for the SPP using the  $BL_0$  heuristic.

**Exercise 7.10** Compute  $NF(L)$  and  $OPT(L)$  for the Example 7.5 on page 203.

**Exercise 7.11** Show the validity of  $\frac{1}{d}(1 - \sum_{i=1}^d \frac{1}{t_i}) = \frac{1}{t_d(t_d-1)d}$  within formula (7.8) on page 217.

**Exercise 7.12** Consider the instance  $E$  of the 2SPP with strip width  $W = 20$  and 11 rectangles as defined in the following table:

| $i$   | 1 | 2  | 3  | 4 | 5 | 6 | 7 | 8  | 9  | 10 | 11 |
|-------|---|----|----|---|---|---|---|----|----|----|----|
| $w_i$ | 2 | 11 | 12 | 6 | 7 | 8 | 9 | 5  | 10 | 10 | 10 |
| $h_i$ | 4 | 4  | 4  | 8 | 8 | 8 | 8 | 12 | 2  | 2  | 2  |

Show that  $z^{b-hor} < z^{c-hor} < H^*$  holds.

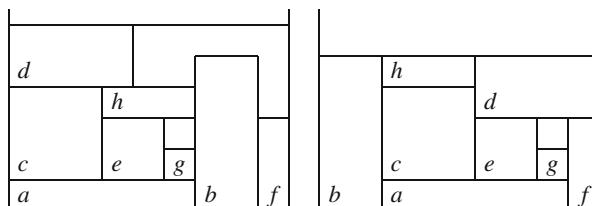
## 7.8 Solutions

**To Exercise 7.1** Instance  $E_1(W)$  with  $m = 1$ ,  $w_1 = 1$ , and  $h_1 = W$  has optimal height  $H^* = W$ , but the area bound is  $lb_A = 1$ . Hence, for each  $\vartheta > 0$  there exists an instance  $E_1(W)$ , such that  $H^* > \vartheta lb_A$  holds. Therefore, for each  $\vartheta > 0$  we have  $\sup_{E_1(W)}(H^*/lb_A) > \vartheta$  and the assertion is proved for the area bound.

For the instance  $E_2(W)$  with  $w_i = W$  and  $h_i = 1$  for all  $i \in I = \{1, \dots, m\}$  we obtain  $H^* = m$  and  $lb_h = 1$ . Hence, again we can find for each  $\vartheta > 0$  an instance  $E_2(W)$  for which  $H^* > \vartheta lb_h$  holds and therefore,  $\sup_{E_2(W)}(H^*/lb_h) > \vartheta$ .

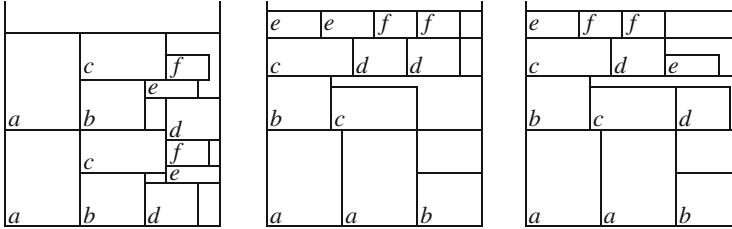
**To Exercise 7.2** We consider the following 2SPP instance with  $m = 4$  rectangles to be placed into a strip of width  $W = 16$ . The items are  $6 \times 8$ ,  $6 \times 8$ ,  $4 \times 7$ , and  $2 \times 12$ . It is easy to see,  $H^* = 15$ , and we take  $\bar{H} = 15$ . Then we have  $lb_h = 12$  and  $lb_A = 148/16 > 9$ . Let  $a_{is} = \frac{8}{9}$  for  $i = 1, \dots, 4$ ,  $s = 1, \dots, 9$ ,  $a_{4s} = \frac{8}{s}$  for  $s = 10, \dots, 15$ , and  $a_{is} = 0$  otherwise. Then we have  $\sum_{s=1}^9 a_{is} = 8 \geq h_i$  for  $i = 1, 2, 3$  and  $\sum_{s=1}^{15} a_{4s} > 12 = h_4$ . Moreover,  $\sum_{i=1}^4 w_i a_{is} \leq 16$  for all  $s = 1, \dots, 15$ . Since  $s \cdot a_{is} < 9$  for all  $i$  and  $s$ , it follows that even  $z_{LP}^{Kan} < \min\{lb_h, lb_A\} \leq lb_0$  is possible.

**To Exercise 7.3** We obtain  $BL(\{a, \dots, h\}) = 6$  and  $BL(\{b, a, f, c, e, g, d, h\}) = 5$  showing that different sequences of the items lead to different results, in general.



**To Exercise 7.4** Obviously,  $\text{NFDH}(\mathcal{L}) = 9$  and  $\text{FFDH}(\mathcal{L}) = \text{BFDH}(\mathcal{L}) = 8$  result. Compare these values with those of Exercise 7.3.

**To Exercise 7.5** We obtain  $\text{BL}(\mathcal{L}) = 18$  and  $\text{NFDH}(\mathcal{L}) = \text{FFDH}(\mathcal{L}) = 20$ .



**To Exercise 7.6** Because of  $h_{\max} \leq \text{OPT}(\mathcal{L})$ , we have

$$\text{NFS}_r(\mathcal{L}) < \frac{2}{r} \text{OPT}(\mathcal{L}) + \frac{h_{\max}}{r(1-r)} \leq \left( \frac{2}{r} + \frac{1}{r(1-r)} \right) \text{OPT}(\mathcal{L}).$$

To show that the estimation is best possible, we construct a list  $\mathcal{L}$  such that for the  $\text{NFS}_r$  pattern the total height of the open shelves is approximately  $\frac{1}{r(1-r)}$ , the area of closed shelves is nearly  $\frac{2}{r}$  of the total area of placed pieces therein, and the height of an optimal packing is approximately equal to the height of the highest piece. Let  $W = 1$ .

Let  $k$  be a positive integer and  $n = 2\lceil r^{-k} \rceil$ . We choose  $\Delta > 0$  and  $\varepsilon > 0$  such that  $3n\Delta < 1$  and  $\varepsilon < r^k - r^{k+1}$ . Then, the list  $\mathcal{L}$  is composed as follows:

$$P_1, P_2, \dots, P_k, T_1, S_1, T_2, S_2, \dots, T_n, S_n,$$

where each rectangle  $P_i$  has width  $\Delta/k$  and height  $r^i + \varepsilon$ . Every rectangle  $T_i$  has width  $\frac{1}{2} - \Delta$  and height  $r^{k+1} + \varepsilon$ , and each rectangle  $S_i$  has width  $3\Delta$  and height  $r^{k+1} + \varepsilon$ .

Within the  $\text{NFS}_r$  pattern each piece  $P_i$  defines a shelf of height  $r^{i-1}$ . Pieces  $T_i$  and  $S_i$  are placed pairwise within a shelf of height  $r^k$ . Hence,

$$\begin{aligned} \text{NFS}_r(\mathcal{L}) &\geq \sum_{i=0}^{k-1} r^i + nr^k - (r^k - r^{k+1} - \varepsilon) \\ &\geq \frac{1}{1-r} - \frac{r^k}{1-r} + \frac{2}{r^k} r^k - (r^k - r^{k+1} - \varepsilon) \geq \frac{1}{1-r} - \frac{r^k}{1-r} + 2 - (r^k - r^{k+1}). \end{aligned}$$

An optimal pattern is as follows: all  $k$  rectangles  $P_i$  are packed side by side occupying a rectangle of width  $\Delta$  and height  $r + \varepsilon$ . On its right side, two rectangles  $T_i$  and  $T_{i+1}$  (with total width  $1 - 2\Delta$ ) are respectively placed in  $n/2$  rows. The total height for that is  $\frac{n}{2}(r^{k+1} + \varepsilon) \leq (1 + r^{-k})(r^{k+1} + \varepsilon) \leq r^{k+1} + r + (1 + r^{-k})\varepsilon$ . Finally, all rectangles  $S_i$  are placed in a line of height  $r^{k+1} + \varepsilon$  above it. The total

height of the constructed pattern results to  $2r^{k+1} + r + (2 + r^{-k})\varepsilon \geq \text{OPT}(\mathbf{L})$ . For any  $\delta > 0$  there exists a sufficiently large  $k$  and a sufficiently small  $\varepsilon$  with

$$\left[ \frac{2}{r} + \frac{1}{r(1-r)} - \delta \right] \text{OPT}(\mathbf{L}) < \text{NFS}_r(\mathbf{L}).$$

**To Exercise 7.7** Function  $f(r) = \frac{2}{r} + \frac{1}{r(1-r)}$ ,  $0 < r < 1$ , has its global minimum at  $r^* = (3 - \sqrt{3})/2$ .

**To Exercise 7.8** The algorithms Multi-Start Local Search (MLS) and Iterated Local Search (ILS) for the SPP are as follows:

#### Multi-Start Local Search (MLS) for the SPP

- (1) Choose a starting permutation  $\pi^*$ .
- (2) Generate randomly another permutation  $\pi$  and improve it using LS.
- (3) If  $BL(\pi) < BL(\pi^*)$ , then set  $\pi^* := \pi$ . If a stopping criterion is fulfilled, then stop. Otherwise, go to step (2).

#### Iterated Local Search (ILS) for the SPP

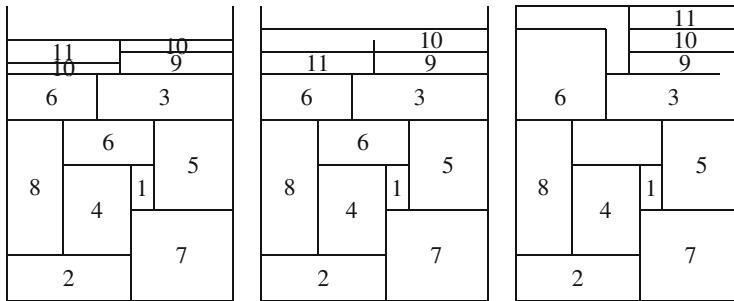
- (1) Choose a starting permutation  $\pi^*$ .
- (2) Generate a permutation  $\pi$  based on  $\pi^*$  using random disturbances and improve it by means of LS.
- (3) If  $BL(\pi) < BL(\pi^*)$ , then set  $\pi^* := \pi$ . If a stopping criterion is fulfilled, then stop. Otherwise, go to step (2).

**To Exercise 7.9** The algorithm Tabu Search (TS) for the SPP using the  $BL_0$  heuristic can be stated as

#### Algorithm Tabu Search (TS) for the SPP

- (1) Compute a pattern using the  $BL_0$  heuristic for  $\pi^* := \pi$  and set  $T := \emptyset$ .
- (2) Compute a best solution  $\bar{\pi} \in U(\pi) \setminus (\{\pi\} \cup T)$  and set  $\pi := \bar{\pi}$ .
- (3) If  $BL_0(\pi) < BL_0(\pi^*)$ , then set  $\pi^* := \pi$ . If a stopping criterion is fulfilled, then stop. Otherwise, update  $T$  and go to step (2).

**To Exercise 7.10** Obviously,  $\text{OPT}(\mathbf{L}) = 2$ . An optimal pattern can be obtained, for instance, using the BL heuristic with piece sequence  $R_1, \dots, R_k, S_1, \dots, S_k$ . Since two neighbored pieces in  $\mathbf{L}$  have a total width larger than  $W$ , we have  $\text{NF}(\mathbf{L}) = k+1$ .



**Fig. 7.13** Example with  $z^{b-hor} < z^{c-hor} < H^*$

**To Exercise 7.11** We show by induction that  $\sum_{i=1}^d \frac{1}{t_i} = 1 - \frac{1}{t_d(t_d-1)}$  holds. Obviously, due to  $t_1 = 2$  the assertion holds for  $d = 1$ .

Because of  $t_d - 1 = t_{d-1}(t_{d-1} - 1)$  and  $\frac{1}{t_{d-1}} - \frac{1}{t_d} = \frac{1}{t_d(t_d-1)}$ , we have

$$1 - \frac{1}{t_{d-1}(t_{d-1} - 1)} + \frac{1}{t_d} = 1 - \frac{1}{t_d(t_d - 1)}$$

which implies the assertion.

**To Exercise 7.12** We obtain  $z^{b-hor} = 23$ ,  $z^{c-hor} = 24$ , and  $H^* = 26$ . Corresponding patterns are shown in Fig. 7.13.

## References

1. E. Aarts, J. Korst, *Simulated Annealing and Boltzmann Machines* (Wiley, Chichester, 1989)
2. E. Aarts, J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization* (Wiley, Chichester, 1997)
3. R. Alvarez-Valdes, F. Parreno, J.M. Tamarit, A branch and bound algorithm for the strip packing problem. OR Spektrum **31**, 431–459 (2009)
4. B.S. Baker, J.S. Schwarz, Shelf algorithms for two-dimensional packing problems. SIAM J. Comput. **12**(3), 508–525 (1983)
5. B.S. Baker, E.G. Coffman, R.L. Rivest, Orthogonal packings in two dimensions. SIAM J. Comput. **9**(4), 846–855 (1980)
6. A. Bekrar, I. Kacem, C. Chu, A comparative study of exact algorithms for the two dimensional strip packing problem. J. Ind. Syst. Eng. **1**(2), 151–170 (2007)
7. G. Belov, G. Scheithauer, A new model and lower bounds for oriented non-guillotine two-dimensional strip packing, in *WSCSP2005 Workshop on Cutting Stock Problems 2005*, ed. by L. Pal (Sapientia University, Miercurea Ciuc, 2005), pp. 19–27
8. G. Belov, G. Scheithauer, Setup and open-stacks minimization in one-dimensional stock cutting. INFORMS J. Comput. **19**(1), 27–35 (2007)
9. G. Belov, G. Scheithauer, E.A. Mukhacheva, One-dimensional heuristics adapted for two-dimensional rectangular strip packing. J. Oper. Res. Soc. **59**, 823–832 (2008)

10. G. Belov, V. Kartak, H. Rohling, G. Scheithauer, One-dimensional relaxations and LP bounds for orthogonal packing. *Int. Trans. Oper. Res.* **16**, 745–766 (2009)
11. A. Bortfeldt, A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *Eur. J. Oper. Res.* **172**, 814–837 (2006)
12. M.A. Boschetti, L. Montaletti, An exact algorithm for the two-dimensional strip-packing problem. *Oper. Res.* **58**, 1774–1791 (2010)
13. D.J. Brown, An improved BL lower bound. *Inf. Process. Lett.* **11**(1), 37–39 (1980)
14. B. Chazelle, The bottom-left bin-packing heuristic: an efficient implementation. *IEEE Trans. Comput.* **32**(8), 697–707 (1983)
15. E.G. Coffman, M.R. Garey, D.S. Johnson, R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.* **9**(4), 808–826 (1980)
16. E.G. Coffman, M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: a survey, in *Approximation Algorithms for NP-Hard Problems*, ed. by D. Hochbaum (PWS Publishing, Boston, 1996), pp. 46–93
17. J. Csiszák, G.J. Woeginger, Shelf algorithms for on-line strip packing. *Inf. Process. Lett.* **63**, 171–175 (1997)
18. M.R. Garey, D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
19. D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.* **3**(4), 299–325 (1974)
20. C. Kenyon, E. Remila, A near-optimal solution to a two-dimensional cutting stock problem. Technical report, Université J. Monnet (2000)
21. F.M. Liang, A lower bound for on-line bin packing. *Inf. Process. Lett.* **10**(2), 76–79 (1980)
22. S. Martello, P. Toth, *Knapsack Problems* (Wiley, Chichester, 1990)
23. S. Martello, M. Monaci, D. Vigo, An exact approach to the strip packing problem. *INFORMS J. Comput.* **15**(3), 310–319 (2003)
24. E.A. Mukhacheva, G.N. Belov, V.M. Kartack, A.S. Mukhacheva, Linear one-dimensional cutting-packing problems: numerical experiments with the sequential value correction method (SVC) and a modified branch-and-bound method (MBB). *Pesqui. Oper.* **20**(2), 153–168 (2000)
25. V.J. Reysard-Smith, L.H. Osman, C.R. Reeves, C.D. Smith, *Modern Heuristic Search Methods* (Wiley, Chichester, 1996)
26. I.A. Rykov, Algorithms with performance guarantees for scheduling, packing and vector subset selection problems. PhD thesis, Russian Academy of Sciences, Siberian Branch, Sobolev Institute of Mathematics (2009)
27. H.E. Salzer, The approximation of numbers as sums of reciprocals. *Am. Math. Mon.* **54**, 135–142 (1947)
28. I. Schiermeyer, Reserve-fit: a 2-optimal algorithm for packing rectangles, in *Lecture Notes in Computer Science, Proc. Of the 2nd European Symposium on Algorithms* (1994)
29. S.S. Seiden, G.J. Woeginger, The two-dimensional cutting stock problem revisited. *Math. Program. Ser. A* **102**, 519–530 (2005)
30. A. Steinberg, A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.* **26**(2), 401–409 (1997)
31. A. van Vliet, Lower and upper bounds for on-line bin packing and scheduling heuristic. PhD thesis, Erasmus University, Rotterdam, Netherlands (1995)

# Chapter 8

## Two-Dimensional Bin Packing

The *two-dimensional bin packing problem* (2BPP) occurs in different variants in important real-world applications such as glass, paper, and steel cutting. A set of two-dimensional, differently sized, rectangular items is given. They have to be packed into (or cut out of) the minimum number of identical, rectangular bins. Since the one-dimensional BPP is known to be NP-hard, the 2BPP is an NP-hard optimization problem, too.

### 8.1 Problem Statement and Modeling

In the 2BPP a set of  $n$  rectangular items  $i \in I = \{1, \dots, n\}$  as well as an unlimited number of identical rectangular bins are given. Item  $i$  possesses width  $w_i$  and height  $h_i$ , and the bins have width  $W$  and height  $H$ . Without loss of generality, we assume that all input data are integer, and that  $0 < w_i \leq W$  and  $0 < h_i \leq H$  hold for all  $i \in I$ . If not stated otherwise, we further assume that rotation is not permitted.

For convenience, we denote an instance of the 2BPP by  $E = (n, w, h, W, H)$ . Moreover, a corresponding *normalized* instance  $E' = (n, w', h', 1, 1)$  results by setting  $w'_i := w_i/W$  and  $h'_i := h_i/H$  for all  $i \in I$ .

Similar to the one-dimensional bin packing problem (1BPP), the 2BPP can be modeled as an ILP problem if an appropriate characterization of feasibility of a two-dimensional pattern is available. To this end, we first formulate a pattern-oriented basic model with unknown patterns.

Let  $\bar{z} = \bar{z}(E)$  be any upper bound of the optimal value of 2BPP-instance  $E$  with  $\bar{z} \leq n$ , for example, obtained by a heuristic. That means, not more than  $\bar{z}$  bins are needed to pack all items. Let  $K := \{1, \dots, \bar{z}\}$  denote an appropriate index set.

To obtain the basic model for the 2BPP, we introduce binary variables to indicate whether bin  $k$  is used:

$$\delta_k = \begin{cases} 1, & \text{if bin } k \text{ is used in the solution,} \\ 0, & \text{otherwise,} \end{cases} \quad k \in K.$$

Binary vectors  $a^k = (a_{1k}, \dots, a_{nk})^\top \in \mathbb{B}^n$ ,  $k \in K$ , are used to represent the (unknown) patterns. As usual,  $a_{ik} = 1$  states that item  $i$  is contained in bin  $k$ . Then, the basic model of the 2BPP is the following:

### Basic model of the 2BPP

$$z^{2BPP} = z^{2BPP}(E) := \min \sum_{k \in K} \delta_k \quad \text{s.t.} \quad (8.1a)$$

$$\sum_{k \in K} a_{ik} = 1, \quad i \in I, \quad (8.1b)$$

$$(a_{1k}, \dots, a_{nk})^\top \text{ represents a feasible pattern if } \delta_k = 1, \quad k \in K, \quad (8.1c)$$

$$a_{ik} \leq \delta_k, \quad i \in I, k \in K, \quad (8.1d)$$

$$a_{ik}, \delta_k \in \mathbb{B}, \quad i \in I, k \in K. \quad (8.1e)$$

According to (8.1a), we aim to minimize the number of used bins. Restrictions (8.1b) ensure that all the items are packed into bins according to feasible patterns, (8.1c). In order to obtain an ILP model, we need to concretize the feasibility of patterns by means of linear constraints and, possibly, additional variables. Subsequently, we consider three types of patterns: non-guillotine, 2-stage and 3-stage guillotine patterns.

In order to reduce the number of unknowns and to avoid symmetric solutions, we can, without loss of generality, fix some variables and add some restrictions:

$$\delta_k \geq \delta_{k+1}, \quad k = 1, \dots, \bar{z} - 1, \quad (8.2)$$

$$a_{11} := 1, \quad a_{21} + a_{22} := 1, \dots, \quad a_{\bar{z}-1,1} + \dots + a_{\bar{z}-1,\bar{z}-1} = 1. \quad (8.3)$$

Obviously, these conditions induce some order of the patterns within a solution. Constraints (8.3) are not additional ones since they are induced by (8.1b) when setting

$$a_{ik} := 0, \quad k = i + 1, \dots, \bar{z}, \quad i = 1, \dots, \bar{z} - 1.$$

This kind of assigning the first items to the first bins is consistently applied in [20] and [23] for the 2- and 3-stage 2BPP, respectively, as well as in the subsequent subsections.

### 8.1.1 Non-Guillotine Patterns

To replace constraints (8.1c) by some linear inequalities, we introduce additional variables according to the Padberg-type model described in Sect. 5.2.2. Obviously, the pattern coefficients  $a_{ik}$  serve as decision variables indicating whether item  $i$  is packed into bin  $k$  or not,  $i \in I, k \in K$ . We denote again the allocation point (lower left corner) of item  $i$  by  $(x_i, y_i)$  and use binary variables  $u_{ij}$  and  $v_{ij}$  for  $i, j \in I, i \neq j$ , to model the non-overlapping of pieces  $i$  and  $j$ .

#### Feasibility of non-guillotine patterns

$$0 \leq x_i \leq W - w_i, \quad 0 \leq y_i \leq H - h_i, \quad i \in I, \quad (8.4a)$$

$$x_i + w_i \leq x_j + W(1 - u_{ij}), \quad i, j \in I, i \neq j, \quad (8.4b)$$

$$y_i + h_i \leq y_j + H(1 - v_{ij}), \quad i, j \in I, i \neq j, \quad (8.4c)$$

$$u_{ij} + u_{ji} + v_{ij} + v_{ji} \geq a_{ik} + a_{jk} - 1, \quad i, j \in I, i < j, k \in K, \quad (8.4d)$$

$$a_{ik}, u_{ij}, v_{ij} \in \{0, 1\}, \quad i, j \in I, i \neq j.$$

Restrictions (8.4a) ensure that piece  $i$  lays completely within one of the bins. Conditions (8.4b) and (8.4c) model the interrelation of two placed pieces. If  $a_{ik} = 1$  and  $a_{jk} = 1$ , that means, both items are packed into the same bin, then, because of (8.4d), at least one of the variables  $u_{ij}$ ,  $u_{ji}$ ,  $v_{ij}$ , and  $v_{ji}$  gets the value 1. If, for example,  $u_{ij} = 1$ , then  $x_i + w_i \leq x_j$  follows due to (8.4b). Hence, the two rectangles do not overlap. The same holds if  $v_{ij} = 1$ . Because of (8.4d), if  $a_{ik} = 0$  and  $a_{jk} = 1$  for some  $k$  then all  $u$ - and  $v$ -variables related to  $i$  and  $j$  can take value 0 so that all constraints (8.4b) and (8.4c) become redundant.

Although model (8.4) has a polynomial number of binary variables, it is hard to solve because of a weak LP relaxation, see Exercise 8.1. Therefore, additional restrictions should be considered to strengthen the LP relaxation. For example, the usage of condition

$$\frac{1}{WH} \sum_{i \in I} w_i h_i a_{ik} \leq \delta_k, \quad k \in K, \quad (8.5)$$

ensures that the rounded-up LP bound dominates the continuous lower bound  $lb_c := \lceil \frac{1}{WH} \sum_{i \in I} w_i h_i \rceil$ , see Exercise 8.2. Moreover, the concept of sequence-pairs, as described in Sect. 5.2.2 for the OPP, can be applied to reduce the number of binary variables and to improve the performance of solution methods because of a tighter LP relaxation.

To further reduce the occurrence of symmetric solutions, we can add

$$x_i \leq (W - w_i)/2, \quad y_i \leq (H - h_i)/2 \quad \text{for an individual item } i \in I.$$

Depending on the input data, possibly further constraints can be regarded to strengthen the LP relaxation. For example, if  $w_i + w_j > W$  and  $h_i + h_j > H$  for some items  $i, j \in I$ ,  $i \neq j$ , then they cannot be placed into the same bin. Hence,

$$\alpha_{ik} + \alpha_{jk} \leq 1, \quad k \in K,$$

holds in every feasible solution. If  $w_i + w_j > W$  and  $h_i + h_j \leq H$  for some items  $i \neq j \in I$ , then they cannot be placed side by side in horizontal direction. Hence,

$$u_{ij} = u_{ji} = 0$$

is satisfied in every feasible solution. Similarly, if  $w_i + w_j \leq W$  and  $h_i + h_j > H$  for some items  $i \neq j \in I$ , then

$$v_{ij} = v_{ji} = 0$$

is fulfilled in every feasible solution.

### 8.1.2 Two-Stage Guillotine Patterns

Based on the definition of a 2-stage guillotine pattern (G-pattern) in Sect. 6.3, we consider here only the *non-exact* case with horizontal guillotine cuts (G-cuts) in the first stage (cf. Fig. 6.4). These cuts produce (horizontal, rectangular) strips which are further dissected by vertical G-cuts to obtain the desired items (trimming horizontal G-cuts are permitted).

In order to characterize feasibility of a (non-exact) 2-stage G-pattern, we introduce binary variables  $\beta_{ij}$  and  $\alpha_{jk}$  to indicate that item  $i \in I$  is contained in strip  $j$  (then  $\beta_{ij} = 1$ ) and strip  $j$  is contained in bin  $k \in K$  (then  $\alpha_{jk} = 1$ ). Since the number of such strips is not known in advance, we allow  $n = |I|$  possible strips. To differentiate the strip patterns  $(\beta_{1j}, \dots, \beta_{nj})$ , we identify its index  $j$  with that of the highest item in it. Therefore, and to reduce the number of symmetric solutions, we assume in the following

$$h_1 \geq h_2 \geq \dots \geq h_n.$$

Since now the highest item in a strip is that with smallest index, we have

$$\beta_{ij} = 0 \text{ for } i = 1, \dots, j-1 \quad \text{and} \quad \beta_{jj} = 1 \text{ for } j = 1, \dots, n.$$

Thus, the height of strip  $j \in I$  is equal to  $h_j$ . Moreover, we can set

$$\beta_{ij} := 0, \quad j = i+1, \dots, n, \quad i = 1, \dots, n-1,$$

enforcing that item 1 is contained in strip 1, item 2 in strip 1 or 2, etc.

Similarly, we can set

$$\alpha_{jk} := 0, \quad k = j + 1, \dots, \bar{z}, \quad j = 1, \dots, \bar{z} - 1,$$

enforcing that strip  $j$  is contained in one of the first  $j$  bins if  $j \leq \bar{z}$ .

A combination of (horizontal) strips fits into bin  $k$  if

$$\sum_{j \in I} h_j \alpha_{jk} \leq H \delta_k$$

where  $\delta_k = 1$  indicates again that bin  $k$  is used, otherwise,  $\alpha_{jk} = 0$  is implied for each  $j$ . Obviously, the feasibility of strip pattern  $j$  ( $j \in \{1, \dots, n\}$ ) is fulfilled if

$$\sum_{i=j+1}^n w_i \beta_{ij} \leq (W - w_j) \beta_{jj}$$

holds. That means, if strip  $j$  is used, then the total width of the other items packed into strip  $j$  must not exceed  $W - w_j$ , and if  $\beta_{jj} = 0$  then no item can be contained in strip  $j$ .

### ILP model of the two-stage 2BPP

$$z^{2BPP-2s} = z^{2BPP-2s}(E) := \min \sum_{k \in K} \delta_k \quad \text{s.t.} \quad (8.6a)$$

$$\sum_{j=1}^i \beta_{ij} = 1, \quad i \in I, \quad (8.6b)$$

$$\sum_{k \in K} \alpha_{jk} = \beta_{jj}, \quad j \in I, \quad (8.6c)$$

$$\sum_{i=j+1}^n w_i \beta_{ij} \leq (W - w_j) \beta_{jj}, \quad j \in I, \quad (8.6d)$$

$$\sum_{j \in I} h_j \alpha_{jk} \leq H \delta_k, \quad k \in K, \quad (8.6e)$$

$$\alpha_{jk}, \beta_{ij}, \delta_k \in \mathbb{B}, \quad i, j \in I, k \in K. \quad (8.6f)$$

In fact, all  $a_{ik}$ -variables in the basic model are eliminated and we have

$$a_{ik} = \sum_{j \in I} \beta_{ij} \alpha_{jk} \quad (8.7)$$

for all  $i$  and  $k$ . Note that the LP lower bound  $z_{LP}^{2BPP-2s}$  dominates the continuous lower bound, see Exercise 8.3.

A similar model where the first item in a strip is directly addressed by an extra variable (as in Sect. 6.4) is proposed by Lodi, Martello, and Vigo in [20]. Note that some adaptations to regard additional restrictions, such as rotation of pieces, are addressed there, too. An application of the model to obtain a single optimal 2-stage pattern and to optimally use a given number of bins is considered in [14]. Clearly, if the *exact* 2-stage case has to be considered, then a similar model can be formulated and further variables can be fixed to be 0.

### 8.1.3 Three-Stage Guillotine Patterns

Similar to the previous subsection, we model here only the *non-exact* case with horizontal guillotine cuts (G-cuts) in the first stage. These cuts yield (horizontal) *strips* which are further dissected by vertical G-cuts producing *stacks*. These stacks can be cut again by horizontal G-cuts to obtain the desired items (trimming vertical G-cuts are permitted, see Sect. 6.4). Moreover, we only model the restricted case where each strip height is determined by an individual item.

In order to characterize feasibility of a (non-exact) 3-stage G-pattern, we introduce now three types of binary variables. Variables  $\alpha_{jk}$  are used again to indicate whether strip  $j$  is contained in bin  $k \in K$  (then  $\alpha_{jk} = 1$ ). Variables  $\beta_{sj}$  model the containment of stack  $s \in I$  within strip  $j$  (then  $\beta_{sj} = 1$ ), and  $\gamma_{is}$  indicates that item  $i \in I$  is contained in stack  $s$  (then  $\gamma_{is} = 1$ ). Since the number of such stacks and strips is not known in advance, we allow  $n = |I|$  possible stacks and strips. Note that the number of stacks can be bounded by  $n$  since at least one item has to be contained within any stack. Moreover, we define the stack widths to be the width of the first item in it. Therefore, we can identify the index of a stack with the index of the first contained item.

Without loss of generality, we assume again that

$$h_1 \geq h_2 \geq \dots \geq h_n$$

is fulfilled. Similarly to above, we define

$$\alpha_{jk} := 0, \quad k = j + 1, \dots, \bar{z}, \quad j = 1, \dots, \bar{z} - 1, \quad (8.8)$$

enforcing that strip  $j$  is contained in one of the first  $j$  bins if  $j < \bar{z}$ . Furthermore, strip  $j$  can only be used if stack  $j$  is the first stack within the strip meaning that no stack  $s$  with  $s < j$  is allowed to appear in strip  $j$ . Hence, we can define

$$\beta_{sj} := 0, \quad s = 1, \dots, j - 1, \quad j = 1, \dots, n, \quad (8.9)$$

which enforces that stack 1 has to be contained in strip 1, stack 2 in strip 1 or 2, etc., if used.

Due to our restriction on a 3-stage G-pattern, we can further fix

$$\gamma_{is} := 0, \quad s = i + 1, \dots, n, \quad i = 1, \dots, n - 1,$$

enforcing that item 1 is contained in stack 1, item 2 in stack 1 or 2, etc. Furthermore,

$$\gamma_{is} := 0 \quad \text{for all } i \in \{s + 1, \dots, n\} \text{ with } w_i > w_s, \quad s = 1, \dots, n - 1.$$

As in the 2-stage case, a combination of strips fits within bin  $k$  if

$$\sum_{j=k}^n h_j \alpha_{jk} \leq H \delta_k$$

where  $\delta_k = 1$  if bin  $k$  is used. If  $\delta_k = 0$ , then  $\alpha_{jk} = 0$  is implied for all  $j$ . The feasibility of a strip pattern is given if

$$\sum_{s=j+1}^n w_s \beta_{sj} \leq (W - w_j) \beta_{jj}$$

and all stacks belonging to strip  $j$  have height at most  $h_j$  which is enforced by

$$\sum_{i=s}^n h_i \gamma_{is} \leq h_j \beta_{sj}, \quad s = j, \dots, n, \quad j = 1, \dots, n.$$

To guarantee that stack  $s$  fulfills  $\gamma_{ss} = 1$ , we further demand

$$\gamma_{is} \leq \gamma_{ss} \quad \text{for } i = s + 1, \dots, n, \quad s = 1, \dots, n - 1.$$

Note that, alternatively,

$$\sum_{i=s}^n h_i \gamma_{is} \leq H \gamma_{ss}, \quad s \in I,$$

can be used. Summarizing, we can state the following model:

### ILP model of the three-stage 2BPP

$$z^{2BPP-3s} = z^{2BPP-3s}(E) := \min \sum_{k \in K} \delta_k \quad \text{s.t.} \tag{8.10a}$$

$$\sum_{k \in K} \alpha_{jk} = \beta_{jj}, \quad j \in I, \tag{8.10b}$$

$$\sum_{j \in I} \beta_{sj} = \gamma_{ss}, \quad s \in I, \quad (8.10c)$$

$$\sum_{s \in I} \gamma_{is} = 1, \quad i \in I, \quad (8.10d)$$

$$\sum_{j \in I} h_j \alpha_{jk} \leq H \delta_k, \quad k \in K, \quad (8.10e)$$

$$\sum_{s=j+1}^n w_s \beta_{sj} \leq (W - w_j) \beta_{jj}, \quad j \in I, \quad (8.10f)$$

$$\sum_{i=s}^n h_i \gamma_{is} \leq h_j \beta_{sj}, \quad s \in I, \quad (8.10g)$$

$$\sum_{i=s}^n h_i \gamma_{is} \leq H \gamma_{ss}, \quad s \in I, \quad (8.10h)$$

$$\alpha_{jk}, \beta_{sj}, \gamma_{is}, \delta_k \in \mathbb{B}, \quad i, j, s \in I, k \in K.$$

The necessity of the restrictions (8.10b)–(8.10h) is explained above, partly in respect to the 2-stage case.

Compared to the basic model, all  $a_{ik}$ -variables are eliminated and it holds

$$a_{ik} = \sum_{j \in I} \sum_{s \in I} \gamma_{is} \beta_{sj} \alpha_{jk} \quad (8.11)$$

for all  $i$  and  $k$ .

As shown in Exercise 8.4, the optimal value  $z_{LP}^{2BPP-3s}$  of the LP relaxation dominates the continuous lower bound.

Similar models where the first item in a strip is directly addressed by an extra variable (as in Sect. 6.4) are proposed in [20] and [23] where more general 3-stage cases are considered, too. It is obvious, if *exact* 3-stage patterns are modeled, then further variables can be fixed to be 0.

## 8.2 Basic Results

In this section, we are interested in relations between the optimal value of the 2BPP and the total area of rectangles to be packed. Let  $E = (n, w, h, W, H)$  denote an instance of the 2BPP with integer input data, and, without loss of generality, let  $0 < w_i \leq W$  and  $0 < h_i \leq H$  for all  $i \in I = \{1, \dots, n\}$ . Moreover, let  $A(E) = A(I)$

denote the total area of all rectangles of  $E$ , i.e.

$$A(E) = A(I) := \sum_{i \in I} w_i h_i,$$

and  $z^{2BPP} = z^{2BPP}(E)$  denotes the optimal value of 2BPP-instance  $E$ .

A first statement is related to a result of Martello and Vigo [21] for the two-dimensional strip packing problem.

**Theorem 8.1** *For any instance  $E = (n, w, h, W, H)$  of the 2BPP with total area  $A(E)$ , we have*

$$z^{2BPP}(E) \leq 4 \left\lceil \frac{A(E)}{WH} \right\rceil.$$

As a consequence of Theorem 8.1, at most four bins are needed to pack a set of items having total area not larger than the bin area. However, if  $A(E) \leq WH$  then at most three bins are needed to pack all items. To see this, we apply Steinberg's theorem (Sect. 5.5.1) which states that all items can be packed into a strip of width  $2W$  and height  $H$ . Let  $(x_i, y_i)$  denote the corresponding allocation point (lower left corner) of item  $i \in I$  in such a packing. Then item  $i$  covers region

$$R_i(x_i, y_i) := \{(x, y) : x_i \leq x < x_i + w_i, y_i \leq y < y_i + h_i\} \subset [0, 2W] \times [0, H].$$

We define the following partition of  $I = \{1, \dots, n\}$ :

$$I_1 := \{i \in I : x_i + w_i \leq W\}, \quad I_2 := \{i \in I : x_i \geq W\}, \quad I_3 := I \setminus (I_1 \cup I_2).$$

It is obvious, all items of  $I_1$  can be packed into a single bin, and all items of  $I_2$  can be placed into a second bin. Since  $I_3 = \{i \in I : x_i < W < x_i + w_i\}$ , the total height  $\sum_{i \in I_3} h_i$  of  $I_3$ -items does not exceed  $H$ . Hence, all items of  $I_3$  can be packed into a third bin. Summarizing, we have

**Theorem 8.2** *To pack all rectangles of a 2BPP-instance  $E = (n, w, h, W, H)$  with  $A(E) \leq WH$ , at most three bins are needed.*

On the other hand, at least three bins are necessary in following cases:

**Theorem 8.3** *Let  $q \in \mathbb{N}$ ,  $q \leq H$ . Then there exist 2BPP-instances  $E = (n, w, h, W, H)$  with  $h_i \leq H/q$  for all  $i \in I$  and  $A(E) \leq WH$  such that at least three bins are needed to pack all items.*

*Proof* The statement is a consequence of the following example. Consider  $m = 2q + 1$  items with  $w_i = \frac{1}{2}(W + \varepsilon)$  and  $h_i = \frac{1}{q+1}(H + \varepsilon)$  for some sufficiently small positive  $\varepsilon$  and  $i \in I = \{1, \dots, 2q + 1\}$ . Then we have:

$$A(I) = \sum_{i \in I} w_i h_i = (2q+1) \frac{W + \varepsilon}{2} \cdot \frac{H + \varepsilon}{q+1} = \frac{2q+1}{2q+2} (WH + \varepsilon(W + H + \varepsilon)) \leq WH.$$

Since only  $q$  items can be packed in one bin, at least three bins are necessary to pack all items. ■

Some other related results concerning 2BPP-instances with  $A(E) \leq WH$  are listed in Sect. 5.5.1.

A statement concerning the minimum number of bins needed to pack all items of a normalized 2BPP-instance is due to Bougeret et al. [3].

**Lemma 8.4** *Let  $k \geq 3$ ,  $k \in \mathbb{N}$ , and let  $E = (n, w, h, W, H)$  be a 2BPP-instance with item widths and heights not greater than 1 and total area not greater than  $k/4$ . Then there exists a packing of  $E$  into  $k$  unit bins.*

Applying this lemma for  $k = 4$  we obtain the same statement as in Theorem 8.1. Using the technique of Theorem 8.2, the Lemma of Bougeret et al. can be strengthened as follows:

**Lemma 8.5** *Let  $k \geq 3$  and let  $E = (n, w, h, W, H)$  be a 2BPP-instance with item widths and heights not greater than 1 and total area not greater than  $k/4$ . Then there exists a packing of  $E$  into  $k - 1$  unit bins if  $k$  is even, and into  $k$  unit bins if  $k$  is odd.*

*Proof* Let  $k = 2m$ ,  $m \geq 2$ , integer. Due to Steinberg's theorem all items can be packed into a strip of width  $2 \cdot k/4 = k/2 = m$  and height 1. Using again the description of a packing by allocation points  $(x_i, y_i)$ , we can define  $m$  index sets

$$I_p := \{i \in I : p - 1 \leq x_i, x_i + w_i \leq p\}, \quad p = 1, \dots, m,$$

and  $m - 1$  index sets

$$I_{m+p} := \{i \in I : x_i < p < x_i + w_i\}, \quad p = 1, \dots, m - 1.$$

Obviously, the  $2m - 1 = k - 1$  index sets  $I_p$  form a partition of  $I$ , and each item set can be packed into a single bin. ■

Now, for  $k = 4$ , the statement of Theorem 8.2 follows from Lemma 8.5: at most three bins are needed. Lemma 8.5 can further be strengthened as follows.

**Lemma 8.6** *Let  $E = (n, w, h, W, H)$  be an instance of the 2BPP. If  $w(I) := \sum_{i \in I} w_i \leq \frac{3}{2}W$ , then at most two bins are necessary to pack all items of  $E$ .*

Exercise 8.5 asks for a proof of the lemma.

**Theorem 8.7** *Let  $E = (n, w, h, W, H)$  be a 2BPP-instance. If*

$$A(E) \leq \frac{3}{4}WH,$$

*then all items of  $E$  can be packed into at most two bins.*

A proof can be found in [4]. The following result is also proposed there.

**Theorem 8.8** Let  $E = (n, w, h, W, H)$  be a 2BPP-instance. If

$$A(E) \leq \frac{k}{4}WH \quad \text{for some } k \geq 3, k \in \mathbb{N},$$

then all items of  $E$  can be packed into at most  $k - 1$  bins. Moreover, the statement is tight.

In case of somewhat smaller items, we have:

**Theorem 8.9** Let  $E = (n, w, h, W, H)$  be a 2BPP-instance with

$$w_i \leq W/2 \text{ and } h_i \leq H/3 \text{ for all } i \in I \quad \text{and} \quad A(E) \leq WH,$$

then all items of  $E$  can be packed into at most two bins. The same assertion holds if  $w_i \leq W/3$ ,  $h_i \leq H/2$  for all  $i \in I$ , and  $A(E) \leq WH$ .

A similar result is still open if  $w_i \leq W/2$  and  $h_i \leq H/2$  for all  $i \in I$  and  $A(E) \leq WH$  is assumed.

### 8.3 Lower Bounds

Let  $E = (n, w, h, W, H)$  denote an instance of the 2BPP with integer input data, and let  $0 < w_i \leq W$  and  $0 < h_i \leq H$  for all  $i \in I = \{1, \dots, n\}$ . Then the so-called *continuous bound*

$$lb_c := \left\lceil \frac{1}{WH} \sum_{i \in I} w_i h_i \right\rceil$$

represents a natural lower bound for the optimal value  $z^{2BPP}(E)$ .

In order to obtain lower bounds for the minimal number of bins  $W \times H$  needed to pack all items  $w_i \times h_i$ ,  $i \in I$ , we use approaches successfully applied for the 1BPP.

Let  $p, q \in \mathbb{N}$  with  $0 < p \leq W/2$  and  $0 < q \leq H/2$ . Similar to [2] and [5] we define five types of items:

$$\begin{aligned} I_{big} &:= \{i \in I : w_i > W - p, h_i > H - q\}, \\ I_{wide} &:= \{i \in I : w_i > W - p, q \leq h_i \leq H - q\}, \\ I_{tall} &:= \{i \in I : p \leq w_i \leq W - p, h_i > H - q\}, \\ I_{med} &:= \{i \in I : p \leq w_i \leq W - p, q \leq h_i \leq H - q\}, \\ I_{tiny} &:= \{i \in I : w_i < p \vee h_i < q\}. \end{aligned}$$

It is obvious, if an item of  $I_{big}$  is placed within a bin, then no item of  $I_{wide}$ ,  $I_{tall}$  and  $I_{med}$  can also be placed in that bin. Moreover, an item of  $I_{wide}$  cannot be contained together with an item of  $I_{tall}$  within the same bin.

Removing all items of  $I_{tiny}$ , we can estimate the minimum number of bins needed to pack the items of  $I \setminus I_{tiny}$ .

Since no item of  $I_{med}$  can be packed side by side to an item of  $I_{wide}$  or on top of an item of  $I_{tall}$ , respectively, we can enlarge the size of these items according to

$$w_i \rightarrow W, i \in I_{wide}, \quad h_i \rightarrow H, i \in I_{tall}.$$

Considering only the area of the items of  $I_{wide} \cup I_{tall} \cup I_{med}$ , we obtain an instance  $\widetilde{E}(p, q) = (\widetilde{n}, \widetilde{w}, C)$  of the one-dimensional BPP with input data  $\widetilde{n} = |I_{wide} \cup I_{tall} \cup I_{med}|$ ,

$$\begin{aligned}\widetilde{w}_i &:= W \cdot h_i, i \in I_{wide}, \\ \widetilde{w}_i &:= w_i \cdot H, i \in I_{tall}, \\ \widetilde{w}_i &:= w_i \cdot h_i, i \in I_{med}, \\ C &:= W \cdot H.\end{aligned}$$

Therefore, any lower bound  $lb_{(p,q)}^{1D} = lb^{1D}(\widetilde{E}(p, q))$  for the 1BPP-instance  $\widetilde{E}(p, q)$  provides a lower bound for the 2BPP:

$$lb_{(p,q)}^{2D} := |I_{big}| + lb_{(p,q)}^{1D}.$$

Maximizing over  $p$  and  $q$  yields the lower bound

$$lb_1^{2D} := \max\{lb_{(p,q)}^{2D} : p = 1, \dots, \lfloor W/2 \rfloor, q = 1, \dots, \lfloor H/2 \rfloor\}.$$

Another possibility results if we define three 1BPP, one for  $I_{wide}$ ,  $I_{tall}$ , and  $I_{med}$  separately. Let  $lb_{wide}^{1D}$  be any lower bound for the minimum number of bins needed to pack all items of  $I_{wide}$ . (The input parameters of the corresponding 1BPP instance are  $h_i, i \in I_{wide}$  and bin capacity  $H$ .) Similar, let  $lb_{tall}^{1D}$  and  $lb_{med}^{1D}$  denote lower bounds for the minimum number of bins needed to pack all items of  $I_{tall}$  and  $I_{med}$ , respectively. (The input parameters of the 1BPP-instance corresponding to  $I_{med}$  are  $w_i h_i, i \in I_{wide}$  and bin capacity  $WH$ .)

All three bounds depend on  $p$  and  $q$ , so we add  $(p, q)$  in the notation and obtain by maximizing over  $p$  and  $q$ :

$$lb_2^{2D} := \max_{p=1, \dots, \lfloor W/2 \rfloor} \max_{q=1, \dots, \lfloor H/2 \rfloor} \{ |I_{big}| + lb_{wide}^{1D}(p, q) + lb_{tall}^{1D}(p, q) + lb_{med}^{1D}(p, q) \}.$$

To define further lower bounds, we consider a further partition of  $I_{med}$ :

$$I_{large} := \{i \in I_{med} : W/2 < w_i \leq W - p, H/2 < h_i \leq H - q\},$$

$$I_{small} := I_{med} \setminus I_{large} = \{i \in I_{med} : p \leq w_i \leq W/2 \vee q \leq h_i \leq H/2\}.$$

The next bound is based on the maximal number that items of  $I_{small}$  can be placed together with an item of  $I_{large}$  within a bin. To this end, we define

$$\mu(\widetilde{w}) := \max\{\sum_{i \in I_{small}} \xi_i : \sum_{i \in I_{small}} w_i \xi_i \leq \widetilde{w}, \xi_i \in \mathbb{B}, i \in I_{small}\},$$

$$\nu(\widetilde{h}) := \max\{\sum_{i \in I_{small}} \xi_i : \sum_{i \in I_{small}} h_i \xi_i \leq \widetilde{h}, \xi_i \in \mathbb{B}, i \in I_{small}\},$$

representing the maximal number of items of  $I_{small}$  which can be placed side by side in horizontal or vertical direction without exceeding a given limit. These two values can be obtained easily when using an appropriate sorting of items. Then the maximal number  $n_i$  of items of  $I_{small}$  which can be placed together with item  $i \in I_{large}$  within a bin is bounded above by

$$n_i \leq \mu(W)\nu(H - h_i) + \mu(W - w_i)\nu(H) - \mu(w_i)\nu(h_i).$$

Since for each item in  $I_{big}$  and  $I_{large}$  an extra bin is needed,

$$lb_3^{2D} := \max_{\substack{p=1, \dots, \lfloor W/2 \rfloor, \\ q=1, \dots, \lfloor H/2 \rfloor}} \left\{ |I_{big} \cup I_{large}| + \max \left\{ 0, \left\lceil \frac{|I_{small}| - \sum_{i \in I_{large}} n_i}{\mu(W)\nu(H)} \right\rceil \right\} \right\}$$

is a lower bound of  $z^{2BPP}(E)$ .

Another consideration leads to a partition of  $I_{small} = I_s^h \cup I_s^w \cup I_s^s$  according to

$$I_s^h := \{i \in I_{small} : w_i > W/2\}, I_s^w := \{i \in I_{small} : h_i > H/2\}, I_s^s := I_{small} \setminus (I_s^h \cup I_s^w).$$

The main idea is to evaluate the number of small items which cannot be packed with the large items and the number of bins needed to pack them. Let

$$n_i(p, q) := \begin{cases} \left\lfloor \frac{W - w_i}{p} \right\rfloor \left\lfloor \frac{H - h_i}{q} \right\rfloor - \left\lfloor \frac{W - w_i}{p} \right\rfloor \left\lfloor \frac{H}{q} \right\rfloor - \left\lfloor \frac{W}{p} \right\rfloor \left\lfloor \frac{H - h_i}{q} \right\rfloor, & \text{if } i \in I_{large}, \\ \left\lfloor \frac{w_i}{p} \right\rfloor \left\lfloor \frac{H}{q} \right\rfloor - \left\lfloor \frac{w_i}{p} \right\rfloor \left\lfloor \frac{H - h_i}{q} \right\rfloor, & \text{if } i \in I_s^w, \\ \left\lfloor \frac{W}{p} \right\rfloor \left\lfloor \frac{h_i}{q} \right\rfloor - \left\lfloor \frac{W - w_i}{p} \right\rfloor \left\lfloor \frac{h_i}{q} \right\rfloor, & \text{if } i \in I_s^h, \\ \left\lfloor \frac{w_i}{p} \right\rfloor \left\lfloor \frac{h_i}{q} \right\rfloor, & \text{if } i \in I_s^s. \end{cases}$$

As shown in [2],

$$lb_4^{2D} := \max_{p=1, \dots, \lfloor W/2 \rfloor, q=1, \dots, \lfloor H/2 \rfloor} \left\{ |I_{big} \cup I_{large}| + \max \left\{ 0, \left\lceil \frac{\sum_{i \in I \setminus I_{big}} n_i(p, q)}{\lceil W/p \rceil \lceil H/q \rceil} \right\rceil \right\} \right\}$$

is a lower bound of  $z^{2BPP}(E)$ .

Further lower bounds for the 2BPP can be found in [5]. Moreover, the application of dual feasible functions (Sect. 3.2.3) to obtain stronger lower bounds for the 2BPP is proposed in [11].

## 8.4 Solution Approaches

Since the 2BPP also belongs to the class of NP-hard optimization problems, a high research interest is concerned with developing fast and efficient heuristics. We address some of the many proposed approaches.

A natural approach to approximately solve the 2BPP consists in reducing it to a two-dimensional strip packing problem (2SPP) and a one-dimensional bin packing problem (1BPP). More precisely, given a 2BPP-instance  $E = (n, w, h, W, H)$ , in a first phase, by means of appropriate 2SPP-heuristics, all rectangles are packed in *blocks* (or *layers*) of widths  $W$  and block-height determined by the tallest item contained in the block. In a second phase, a 1BPP-instance is defined by the obtained blocks, their heights and bin capacity  $H$ . The 1BPP is solved again by an appropriate heuristic. Chung et al. [6] proposed a *Hybrid First Fit* (HFF) heuristic in which the well-known FFDH heuristic for the 2SPP is applied within the first phase, and the FFD heuristic of the 1BPP within the second phase. They showed for the asymptotic performance ratio of the HFF heuristic that

$$2.022 \leq \rho_\infty(\text{HFF}) \leq 2.125$$

holds. Moreover, the following performance result is presented therein:

**Theorem 8.10** *For any instance  $E = (n, w, h, W, H)$  of the 2BPP, the inequality*

$$\text{HFF}(E) \leq \frac{17}{8} z^{2\text{BPP}}(E) + 5$$

*holds where  $\text{HFF}(E)$  is the number of bins needed in the HFF-solution to pack all items.*

Obviously, other algorithms can be used in the two phases, for instance *Best Fit*. Applying exact algorithms within one or both phases is not meaningful since all in all a heuristic approach remains.

Another hybrid technique is analyzed in [12], called *Hybrid Next Fit* (HNF) heuristic. The HNF heuristic works similar to the HFF heuristic since the NFDH heuristic is applied to the 2SPP (first phase) and the NFD heuristic to the subsequent 1BPP, but both phases are performed simultaneously. In [12] the following result is stated:

**Theorem 8.11** *For any instance  $E = (n, w, h, W, H)$  of the 2BPP, the inequality*

$$\text{HNF}(E) \leq 3.382 \cdot z^{2\text{BPP}}(E) + 9,$$

is fulfilled where  $HNF(E)$  is the number of bins needed in the HNF-solution to pack all items. Moreover, concerning the asymptotic performance ratio in dependence of item sizes, if  $w_i \leq W/r$  and  $h_i \leq H/s$  for all  $i \in I$  and some integers  $r$  and  $s$ , then

$$\rho_{r,s}^{\infty}(HNF) = \frac{\alpha}{\alpha - 1} \gamma_s^*$$

where

$$\gamma_s^* = \frac{s-1}{s} \sum_{k=1}^{\infty} \frac{1}{t_k(s)-1}, \quad t_1(s) = s+1, \quad t_{k+1}(s) = \prod_{j=1}^k t_j(s) + 1, \quad k \geq 1,$$

and

$$\alpha = \begin{cases} 2, & \text{if } r = 1, \\ r, & \text{otherwise.} \end{cases}$$

The two-phase approach applying the BFDH and BFD heuristic, respectively, is considered in [1] named therein as *Finite First Fit* heuristic. Note that all three, the HNF, HFF, and HBF heuristic, can be performed in  $O(n \log n)$  time.

*Floor Ceiling* (FC) heuristics, for instance considered in [15–17], are similar to the above approaches, but, in addition, items are packed, if possible, such that their top edge is touching the block *ceiling*.

For further heuristics, the application of metaheuristics as well as concepts for exact solution approaches, we refer to [13, 18, 19].

A heuristic with asymptotic performance ratio 2.25 for the 2BPP where the items are allowed to be rotated is proposed in [10]. The non-oriented case of the 2BPP is also considered in [9] where a branch-and-bound algorithm is proposed.

The exact algorithm proposed in [22] is based on a decomposition technique and a successive restriction of ranges of values. A similar approach, applying other lower bounds, is described in [8].

Greedy randomized adaptive search procedures and variable neighborhood search are used in [7] for solving the 2BPP with variable bin sizes.

It is obvious, so-called *reduction methods* can possibly lead to equivalent instances having better lower bounds, in general, and are therefore useful in exact solution approaches. Such methods are considered, for instance, in [5].

## 8.5 Exercises

**Exercise 8.1** Let  $\bar{z} \geq 4$ . Show that the LP relaxation of model (8.1) together with (8.4) has a feasible solution with objective value  $\sqrt{\bar{z}}$  implying  $z_{LP}^{2BPP} \leq \sqrt{\bar{z}}$ .

**Exercise 8.2** Show that the lower bound defined by the rounded-up optimal value of the LP relaxation of model (8.1) with (8.4) and (8.5) dominates the continuous lower bound.

**Exercise 8.3** Show that the lower bound defined by the rounded-up optimal value of the LP relaxation of model (8.6) dominates the continuous lower bound.

**Exercise 8.4** Show that the lower bound defined by the rounded-up optimal value of the LP relaxation of model (8.10) dominates the continuous lower bound.

**Exercise 8.5** Prove Lemma 8.6: Let  $E = (n, w, h, W, H)$  be an instance of the 2BPP. If  $w(I) := \sum_{i \in I} w_i \leq \frac{3}{2}W$ , then at most two bins are necessary to pack all items of  $E$ .

## 8.6 Solutions

**To Exercise 8.1** Setting  $a_{ik} := \frac{1}{\sqrt{\bar{z}}}$  for all  $i \in I$  and  $k \in K$  implies  $\delta_k = \frac{1}{\sqrt{\bar{z}}}$  since the sum of  $\delta$ -variables is minimized. Obviously, restrictions (8.1b) are fulfilled.

Because of  $\bar{z} \geq 4$ ,  $a_{ik} \leq \frac{1}{2}$  follows for all  $i$  and  $k$ . Therefore, constraints (8.4d) become redundant so that all  $u$ - and  $v$ -variables can take value 0. As a consequence, no non-overlapping constraint is relevant.

**To Exercise 8.2** Because of (8.5), we have

$$\lceil z_{LP}^{2BPP} \rceil = \left\lceil \sum_{k \in K} \delta_k \right\rceil \geq \left\lceil \frac{1}{WH} \sum_{k \in K} \sum_{i \in I} w_i h_i a_{ik} \right\rceil = \left\lceil \frac{1}{WH} \sum_{i \in I} w_i h_i \sum_{k \in K} a_{ik} \right\rceil = lb_c.$$

**To Exercise 8.3** Because of (8.6e), (8.6d), and (8.7), we have

$$\begin{aligned} \sum_{j \in I} h_j \alpha_{jk} &\leq H \delta_k \\ \sum_{j \in I} \sum_{i \in I} w_i h_j \beta_{ij} \alpha_{jk} &\leq WH \delta_k, \\ \sum_{i \in I} \sum_{k \in K} w_i h_j \sum_{j \in I} \beta_{ij} \alpha_{jk} &\leq WH \sum_{k \in K} \delta_k, \\ \sum_{i \in I} w_i h_j \sum_{k \in K} a_{ik} &\leq WH \sum_{k \in K} \delta_k, \end{aligned}$$

which implies the assertion.

**To Exercise 8.4** Because of (8.10e), (8.10f), and (8.11), we have

$$\begin{aligned} \sum_{j \in I} h_j \alpha_{jk} &\leq H \delta_k \\ \sum_{j \in I} h_j \sum_{s \in I} w_s \beta_{sj} \alpha_{jk} &\leq WH \delta_k, \\ \sum_{j \in I} \sum_{s \in I} \sum_{i \in I} w_i h_i \gamma_{is} \beta_{sj} \alpha_{jk} &\leq WH \delta_k, \\ \sum_{j \in I} \sum_{k \in K} w_i h_i \sum_{s \in I} \sum_{i \in I} \gamma_{is} \beta_{sj} \alpha_{jk} &\leq WH \sum_{k \in K} \delta_k, \\ \sum_{i \in I} \sum_{k \in K} w_i h_i a_{ik} &\leq WH \sum_{k \in K} \delta_k, \end{aligned}$$

which implies the assertion.

**To Exercise 8.5** We consider a one-dimensional BPP  $E_1 = (n, w, W)$ . Let  $B_1$  and  $B_2$  denote the sets of items packed by the FFD heuristic, applied to  $E_1$ , into the first two bins, respectively. Assume, there is an item  $j$ , which is not packed by the FFD algorithm into one of the first two bins. Then  $w_j + w(B_1) > W$  and  $w_j + w(B_2) > W$ , and therefore  $2w_j + w(B_1) + w(B_2) > 2W$ . Since  $w_j + w(B_1) + w(B_2) \leq \frac{3}{2}W$ , we obtain  $w_j > \frac{1}{2}W$ , and consequently,  $w(B_1) + w(B_2) < W$ . Therefore all items of  $B_1$  and  $B_2$  can be packed into the first bin, i.e.  $B_2 = \emptyset$  and item  $j$  fits within the second bin.

## References

1. J.O. Berkey, P.Y. Wang, Two-dimensional finite bin-packing algorithms. *J. Oper. Res. Soc.* **38**(5), 423–429 (1987)
2. M.A. Boschetti, A. Mingozzi, The two-dimensional finite bin packing problem. part I: new lower bounds for the oriented case. *4OR* **1**, 27–42 (2003)
3. M. Bougeret, P.F. Dutot, K. Jansen, C. Otte, D. Trystram, Approximation algorithm for multiple strip packing. *Approximation and Online Algorithms*, ed. by E. Bampis, K. Jansen (Springer, Heidelberg, 2009), pp. 37–48
4. T. Buchwald, K. Hoffmann, G. Scheithauer, Relations between capacity utilization, minimal bin size and bin number. *Ann. OR* **217**(1), 55–76 (2014)
5. J. Carlier, F. Clautiaux, A. Moukrim, New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Comput. Oper. Res.* **34**, 2223–2250 (2007)
6. F.R.K. Chung, M.R. Garey, D.J. Johnson, On packing two-dimensional bins. *SIAM J. Algebr. Discrete Methods* **3**(1), 66–76 (1982)
7. A.M. Chwatal, S. Pirkwieser, Solving the two-dimensional bin-packing problem with variable bin sizes by greedy randomized adaptive search procedures and variable neighborhood search, in *Computer Aided Systems Theory & EUROCAST 2011* (Springer, Heidelberg, 2012), pp. 456–463
8. F. Clautiaux, J. Carlier, A. Moukrim, A new exact method for the two-dimensional bin-packing problem with fixed orientation. *OR Lett.* **35**, 357–364 (2007)
9. M. Dell'Amico, S. Martello, D. Vigo, A lower bound for the non-oriented two-dimensional bin packing problem. *Discrete Appl. Math.* **118**, 13–24 (2002)
10. L. Epstein, R. van Stee, This side up!, in *Approximation and Online Algorithms, Second International Workshop, WAOA 2004*, Bergen, Norway, 14–16 September 2004. Revised Selected Papers (2005), pp. 48–60
11. S.P. Fekete, J. Schepers, A general framework for bounds for higher-dimensional orthogonal packing problems. *Math. Methods Oper. Res.* **60**(2), 311–329 (2004)
12. J.B.G. Frenk, G. Galambos, Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem. *Computing* **39**, 201–217 (1987)
13. R. Harren, K. Jansen, L. Prädel, U. M. Schwarz, R. van Stee, Two for one: tight approximation of 2d bin packing. *Int. J. Found. Comput. Sci.* **24**(8), 1299–1328 (2013)
14. A. Lodi, M. Monaci, Integer linear programming models for 2-staged two-dimensional knapsack problems. *Math. Program. Ser. B* **94**, 257–278 (2003)
15. A. Lodi, S. Martello, D. Vigo, Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem, in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, ed. by S. Voss, S. Martello, I. Osman, C. Roucairol (Kluwer, Boston, 1998), pp. 125–139

16. A. Lodi, S. Martello, D. Vigo, Approximation algorithms for the oriented two-dimensional bin packing problem. *Eur. J. Oper. Res.* **112**(1), 158–166 (1999)
17. A. Lodi, S. Martello, D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS J. Comput.* **11**(4), 345–357 (1999)
18. A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems: a survey. *Eur. J. Oper. Res.* **141**(2), 241–252 (2002)
19. A. Lodi, S. Martello, D. Vigo, Heuristic algorithms for the three-dimensional bin packing problem. *Eur. J. Oper. Res.* **141**(2), 410–420 (2002)
20. A. Lodi, S. Martello, D. Vigo, Models and bounds for two-dimensional level packing problems. *J. Comb. Optim.* **8**, 363–379 (2004)
21. S. Martello, D. Vigo, Exact solution of the two-dimensional finite bin-packing problem. *Manag. Sci.* **44**, 388–399 (1998)
22. D. Pisinger, M. Sigurd, Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS J. Comput.* **19**(1), 36–51 (2007)
23. J. Puchinger, G.R. Raidl, Models and algorithms for three-stage two-dimensional bin packing. *Eur. J. Oper. Res.* **183**, 1304–1327 (2007)

# Chapter 9

## Quality Restrictions

In this chapter, we investigate cutting and packing problems arising when heterogeneous material or forbidden regions are present.

In the first part, we consider one-dimensional allocation (cutting) problems where the set of feasible patterns depends on the (raw) material. Thereby, we address a generalized cutting problem in which the length of the pieces of a desired type is only limited by a minimum and a maximum length, i.e., their length can vary within a given interval. In the second part, we turn to two-dimensional cutting problems with defective or forbidden regions.

Such problems occur, for instance, when maximizing hardwood utilization, but also in many other industrial applications. Examples for that can be found, e.g., in [6] and [7] concerning the steel industry.

### 9.1 Cutting of Variable Lengths

In wood industry, logs are cut into timber of given (rectangular) cross section. Such a timber consists, in general, of non-homogeneous material so that parts of it can fulfill different quality properties. On the other hand, the desired products have to meet predefined minimum quality demands so that not every part of the material is suitable to obtain the product.

For example, different quality demands result in those cases where the occurrence of knotholes, of sound-knots, or of wane is allowed or not. To optimize the yield when cutting timber into desired pieces, piece types with higher quality demands have to have a relative higher value to be produced. Then the task is to find a (cutting) pattern such that all produced pieces fulfill the related quality demands and the total yield is maximal.

### 9.1.1 Problem Formulation

To formulate different quality demands, we use the following form of description: all pieces of the same type  $T_i$ ,  $i \in I = \{1, \dots, m\}$ , have to fulfill the same quality demands. We model this by assuming that all related quality demands are fulfilled if a piece of type  $T_i$  is obtained from one of predefined regions of the raw material. To this end, we define so-called *allocation intervals*

$$A_{ik} = [\alpha_{ik}, \omega_{ik}] \subseteq [0, L], \quad k \in K_i := \{1, \dots, k_i\},$$

where  $k_i$  is the number of such intervals and  $L$  denotes the length of the raw material. More precisely, a piece of type  $T_i$  fulfills all desired quality demands if it is obtained from (a part of) a corresponding allocation interval  $A_{ik}$ . Note that we do not assume  $\text{int}(A_{ik}) \cap A_{ij} = \emptyset$  for  $j \neq k, j, k \in K_i$ , but, naturally,  $A_{ik} \not\subset A_{ij}$  for all  $j \neq k, j, k \in K_i$ , can be presumed. Such an assumption is meaningful if, e.g., a situation as described in Exercise 9.1 has to be considered.

Additionally to the complete containment within an appropriate allocation interval, a piece of type  $T_i$  has to have a length  $\ell$  such that  $\ell \in [l_i, L_i]$  where  $l_i$  and  $L_i$  define the minimum and maximum length pieces of type  $T_i$  may have. In case of  $l_i = L_i$ , we say the piece type has *fix-length*, Otherwise, the piece type is called *variable-length*. Obviously, we can assume for all allocation intervals  $A_{ik}$  that  $\omega_{ik} - \alpha_{ik} \geq l_i$  holds for all  $k \in K_i$ . Hence, a piece of type  $T_i$  with length  $\ell$  can be obtained from segment  $[x, x + \ell]$  of the raw material (modeled by  $[0, L]$ ) if an allocation interval  $A_{ik} = [\alpha_{ik}, \omega_{ik}], k \in K_i$ , exists with

$$\alpha_{ik} \leq x, \quad x + \ell \leq \omega_{ik}, \quad \ell \in [l_i, L_i].$$

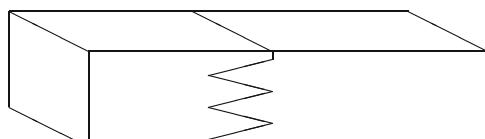
Products with varying lengths are used within wood and paper industry as intermediate goods. Applying the *finger-joining technology*, as depicted in Fig. 9.1, smaller pieces are glued together to obtain larger lengths.

In the following, we suppose that the value  $c_i(\ell)$  of a piece of type  $T_i$ ,  $i \in I$ , and length  $\ell$  is given by

$$c_i(\ell) := c_i \cdot \ell \quad \text{with } c_i > 0.$$

Since parts of the raw material that are not suitable for any desired piece type can exist, we use (for the sake of a simpler presentation) an artificial piece type defined by  $c_{m+1}(\ell) := 0$  for all  $\ell \in A_{m+1,1} := [0, L]$ ,  $l_{m+1} := 0$ ,  $L_{m+1} := L$ , in our

**Fig. 9.1** Finger-joining technology



model, which represents the waste. In this way, we can view the cutting or allocation problem as a *partitioning problem*, i.e., we search for a sequence of consecutive allocated pieces which are placed within corresponding allocation intervals and which provide the maximal total value.

For the problem under consideration, we develop an ILP model and discuss appropriate solution methods. In particular, we present a *branch-and-bound* algorithm and an approach which is based on the so-called *forward state strategy* of *dynamic programming*. Thereby the description follows that which is already used in [4]. A similar problem is addressed in [3]. To simplify the presentation of the solution methods, we disclaim to regard a positive kerf.

### 9.1.2 Modeling

The problem under consideration asks for an optimal sequence of desired pieces. Therefore, it can be formulated as an optimization problem with integer constraints. However, because of piece types with variable lengths the problem cannot be modeled as a pure integer problem. However, the set of potential allocation points to be considered within the optimization process could be reduced to a finite one due to a minimum distance between two neighbored cut positions which are adjustable. In this way, a modeling as pure integer problem becomes possible, but we will not use this opportunity because of the, in general, very large number of potential allocation points, and therefore, of variables, which would result.

To formulate a mixed-integer optimization model with binary variables, we assume in this subsection that for each piece type  $i$  exactly one allocation interval  $A_i = [\alpha_i, \omega_i]$  exists and at most one piece of this type can be allocated. This is not a loss of generality since each piece type can be defined several times.

To describe the allocation of a piece  $T_i$ ,  $i \in I$ , we use a binary variable  $a_i$  according to

$$a_i = \begin{cases} 1, & \text{if } T_i \text{ is allocated,} \\ 0, & \text{otherwise.} \end{cases}$$

We denote the allocation point of piece  $i$  and its length by  $x_i$  and  $\ell_i$ , respectively. Obviously, if piece  $i$  is allocated, i.e., if  $a_i = 1$ , then it covers the interval  $T_i(x_i, \ell_i) := [x_i, x_i + \ell_i]$ . Therefore, we can state the following general model of the considered allocation problem:

#### Allocation of variable lengths

$$v(L) = \max \sum_{i \in I} c_i(\ell_i) \quad \text{s.t.} \tag{9.1a}$$

$$l_i a_i \leq \ell_i \leq L_i a_i, \quad i \in I, \tag{9.1b}$$

$$\text{int}(T_i(x_i, \ell_i)) \cap T_j(x_j, \ell_j) = \emptyset, \quad i, j \in I \text{ with } i \neq j, \quad a_i + a_j = 2, \quad (9.1c)$$

$$T_i(x_i, \ell_i) \subset A_i, \quad i \in I \text{ with } a_i = 1, \quad (9.1d)$$

$$a_i \in \{0, 1\}, \quad x_i, \ell_i \in \mathbb{R}^1, \quad i \in I. \quad (9.1e)$$

Restriction (9.1c) ensures non-overlapping of pieces, and condition (9.1d) demands their placement within the corresponding allocation intervals. Using binary variables  $u_{ij}$  for  $i \neq j, i, j \in I$ , according to

$$u_{ij} = \begin{cases} 1, & \text{if } T_i \text{ is placed to the left of } T_j, \text{ i.e., } x_i + \ell_i \leq x_j, \\ 0, & \text{otherwise,} \end{cases}$$

restrictions (9.1c)–(9.1d) can be replaced by

$$x_i + \ell_i \leq x_j + L(3 - a_i - a_j - u_{ij}), \quad i, j \in I, \quad i \neq j, \quad (9.2a)$$

$$x_i \geq \alpha_i a_i, \quad x_i + \ell_i \leq \omega_i a_i, \quad l_i a_i \leq \ell_i \leq L_i a_i, \quad i \in I, \quad (9.2b)$$

$$u_{ij} + u_{ji} \leq a_i, \quad u_{ij} + u_{ji} \leq a_j, \quad u_{ij} + u_{ji} + 1 \geq a_i + a_j, \quad i, j \in I, \quad i < j, \quad (9.2c)$$

$$u_{ij} \in \{0, 1\}, \quad i, j \in I, \quad i \neq j.$$

Conditions (9.2a) are redundant if piece  $T_i$  or piece  $T_j$  is not allocated. However, if  $a_i + a_j = 2$  holds, then, because of  $u_{ij} + u_{ji} \geq 1$  due to (9.2c), one of the conditions (9.2a) becomes non-trivial. In the case that piece  $i$  is not placed, i.e.  $a_i = 0$ , then  $x_i = \ell_i = 0$  follows from (9.2b).

The model possesses a linear objective function (9.1a), linear restrictions, and a lot of binary variables. However, for pieces  $T_i$  and  $T_j$  with  $\text{int}(A_i) \cap A_j = \emptyset$  no  $u$ -variable is necessary. A further reduction of the number of  $u$ -variables is also possible in the case of a partial overlapping of allocation intervals. For example, if  $\omega_i - \alpha_j < l_i + l_j$ , then  $u_{ji} = 0$  follows. Since the total number of binary variables could be too large for an exact solution of practical problems, in general, we will consider two other approaches in the following.

### 9.1.3 Optimal Value Function

Related to the allocation problem (9.1), we define the *optimal value function*  $v(l)$  for  $0 \leq l \leq L$  according to

$$v(l) := \max_{a_i, x_i, \ell_i} \left\{ \sum_{i \in I} c_i(\ell_i) a_i : a_i, x_i, \ell_i \text{ fulfill (9.1e)–(9.1d) and } x_i + \ell_i \leq l, \quad i \in I \right\}.$$

Function  $v$  is monotone non-decreasing since the feasible region of the optimization problem enlarges with increasing  $l$ . The optimal value function is piecewise continuous since only a finite number of different sequences of allocated pieces exists and the valuations are assumed to be continuous. For the same reason, there exist intervals in which either  $v$  is constant (i.e., ascent 0) or increases linearly with ascent in  $\{c_1, \dots, c_m\}$ . Moreover, function  $v$  is continuous from the right.

An optimal pattern belonging to  $v(l)$  is, in general, not uniquely determined. Since  $x_i$  and  $\ell_i$  are real numbers infinitely many optimal solutions can exist even for a sequence defined by the  $u_{ij}$ -variables.

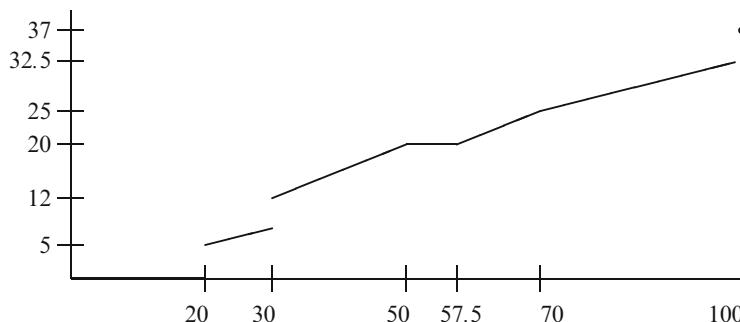
*Example 9.1* Let heterogeneous material of length  $L = 100$  be given. The piece types to be cut are:

$$\begin{aligned} T_1 : l_1 &= 30, L_1 = 50, c_1 = 0.4, A_1 = [0, 60], \\ T_2 : l_2 &= 20, L_2 = 100, c_2 = 0.25, A_2 = [0, 100], \\ T_3 : l_3 &= 30, L_3 = 50, c_3 = 0.4, A_3 = [70, 100]. \end{aligned}$$

The resulting optimal value function  $v$  is depicted in Fig. 9.2.

An optimal pattern for segment  $[0, x]$  with  $x \in [20, 30)$  contains only a piece of type  $T_2$ , and for  $x \in [30, 50]$  only a piece of type  $T_1$  with length  $x$ . Moreover, an optimal pattern for segment  $[0, x]$  with  $x \in [57.5, 70]$  possesses a piece of type  $T_1$  in  $[0, x - 20]$ , i.e., with length  $x - 20$ , and a piece of type  $T_2$  with minimum length 20 in  $[x - 20, x]$ . For  $x \in (70, 100)$  we obtain a piece of type 1 from segment  $[0, 50]$  and a piece of type 2 with length  $x - 50$ . Finally, an optimal pattern for  $[0, 100]$ , consists of a piece of type  $T_1$  in  $[0, 50]$ , of type  $T_2$  in  $[50, 70]$ , and of type  $T_3$  in  $[70, 100]$ .

If we modify the example such that for piece type  $T_1$  the minimum length  $l_1 = 50$  and for  $T_2$  the maximum length  $L_2 = 20$  are demanded, and if  $T_2$  can be used several times, then infinitely many optimal solutions appear for  $x \in (40, 50)$ . In each case, two pieces of type  $T_2$  are obtained: a first one in  $[x_1, x_1 + 20]$  and a second one in  $[x_2, x_2 + 20]$  where  $x_1$  and  $x_2$  are restricted by  $0 \leq x_1, x_1 + 20 \leq x_2$ , and  $x_2 + 20 \leq x$ . ■



**Fig. 9.2** Optimal value function of Example 9.1

In the following, we consider again the general problem as considered in Sect. 9.1.1, i.e., with  $k_i \geq 1$  for all  $i \in I$ . If in some part of the raw material the quality demand of every piece type is not fulfilled, i.e., if

$$[0, L] \setminus \bigcup_{i \in I} \bigcup_{k \in K_i} A_{ik} \neq \emptyset$$

holds, then the whole problem decomposes into some independent smaller problems. Naturally, the optimal value of the initial problem is equal to the sum of the optimal values of the decomposed smaller problems.

### 9.1.4 Upper Bounds and Solution Strategy

To develop efficient solution approaches the usage of bounds is beneficial in many cases. In order to define appropriate upper bounds we assume in the following, without loss of generality, that the piece types are given in non-increasing order with respect to their *relative profit*, i.e.,

$$c_1 \geq c_2 \geq \dots \geq c_m > c_{m+1} = 0 \quad (9.3)$$

is satisfied. We define an index  $i_1(x) \in \{1, \dots, m+1\}$  for each  $x \in [0, L]$  according to

$$i_1(x) := \min_{1 \leq i \leq m+1} \{i : \exists k \in K_i \text{ with } x \in A_{ik}\}.$$

Hence, index  $i_1(x)$  represents the best-possible piece type having an allocation interval which contains  $x$ .

**Proposition 9.1** *Let (9.3) be fulfilled. For  $x \in [0, L]$  let  $ub_1(x)$  be defined according to*

$$ub_1(x) := \int_{t=x}^L c_{i_1(t)} dt.$$

*Then there does not exist any feasible pattern for segment  $[x, L]$  whose value exceeds  $ub_1(x)$ .*

*Proof* The upper bound  $ub_1(x)$  is equal to the maximal value that can be obtained when parts of pieces are allocated in  $[x, L]$  and conditions  $\ell_i \geq l_i$  are not regarded. ■

The optimal value for the interval  $[x, L]$  can be estimated from above more precisely in comparison to  $ub_1(x)$  using

$$i_2(x, t) := \min_{1 \leq i \leq m+1} \{i : \exists k \in K_i \text{ with } t \in A_{ik}, x + l_i \leq \omega_{ik}\} \quad \text{where } x \leq t \leq L.$$

Within the definition of  $i_2(x, t)$  only such piece types are regarded for which at least the minimum length can be obtained for allocation point  $x$ .

**Proposition 9.2** *Let (9.3) be fulfilled. For  $x \in [0, L]$  let  $ub_2(x)$  be defined according to*

$$ub_2(x) := \int_{t=x}^L c_{i_2(x,t)} dt.$$

*Then, there does not exist any feasible pattern for segment  $[x, L]$  whose value exceeds  $ub_2(x)$ .*

During the computation of  $ub_1(x)$  and of  $ub_2(x)$  the following facts can be exploited: the set

$$\{\alpha_{ik}, \omega_{ik} : k \in K_i, i \in I \cup \{m + 1\}\} =: \{\pi_0, \pi_1, \dots, \pi_\gamma\},$$

which contains all starting and end points of the allocation intervals, defines a partition of  $[0, L]$  into maximal intervals  $[\pi_{j-1}, \pi_j]$ ,  $j = 1, \dots, \gamma$ , which do not contain any boundary point of an allocation interval in the interior. Hence, we have  $0 = \pi_0 < \pi_1 < \dots < \pi_\gamma = L$ .

**Proposition 9.3** *For each  $j \in \{1, \dots, \gamma\}$ , there exists a uniquely determined minimal index  $i_{1,j} \in \{1, \dots, m + 1\}$  with*

$$i_{1,j} = i_1(x), \quad x \in (\pi_{j-1}, \pi_j).$$

Proposition 9.3 implies a simple opportunity to compute  $ub_1(x)$ . Using  $j_0 := j_0(x) := \min\{j : \pi_j \geq x\}$  the bound  $ub_1(x)$  can be obtained according to

$$ub_1(x) = c_{i_{1,j_0}}(\pi_{j_0} - x) + \sum_{j=j_0+1}^{\gamma} c_{i_{1,j}}(\pi_j - \pi_{j-1}).$$

To get a similar statement for  $ub_2(x)$ , we need to regard additionally the allocation points determined by the minimum lengths. These points define together with the starting and end points of the allocation intervals another, more detailed, partition of  $[0, L]$  according to

$$\{\alpha_{ik}, \omega_{ik}, \omega_{ik} - l_i : k \in K_i, i \in I\} =: \{\zeta_0, \zeta_1, \dots, \zeta_{\widetilde{\gamma}}\}$$

into  $\widetilde{\gamma}$  maximal intervals  $[\zeta_{j-1}, \zeta_j]$  which do not contain any point  $\alpha_{ik}$ ,  $\omega_{ik}$ , and  $\omega_{ik} - l_i$  in the interior. Then, we have

**Proposition 9.4** *For each  $j \in \{1, \dots, \widetilde{\gamma}\}$ , there exists a uniquely determined minimal index  $i_{2,j} \in \{1, \dots, m + 1\}$  with*

$$i_{2,j} = i_2(x), \quad x \in (\zeta_{j-1}, \zeta_j).$$

Therefore, the upper bound  $ub_2(x)$  can be computed, similar to  $ub_1(x)$ , according to

$$ub_2(x) = c_{i_{2,j_0}}(\zeta_{j_0} - x) + \sum_{j=j_0+1}^{\gamma} c_{i_{2,j}}(\zeta_j - \zeta_{j-1})$$

using  $j_0 := j_0(x) := \min\{j : \zeta_j \geq x\}$ . The functions  $ub_1$  and  $ub_2$  are monotone non-increasing and piecewise linear. Moreover,  $ub_1$  is continuous.

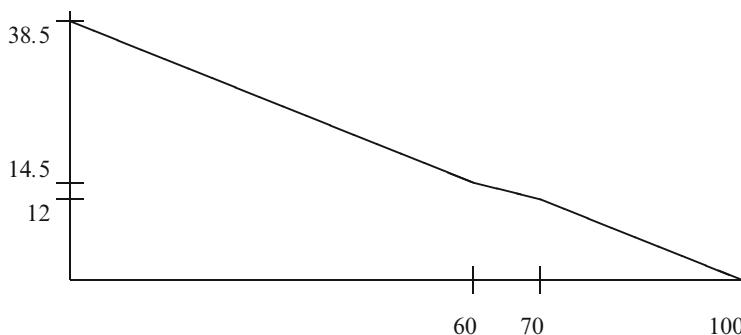
*Example 9.2* Let us consider again the situation of Example 9.1, but we identify piece types 1 and 3 due to  $c_1 = c_3 = 0.4$ ,  $l_1 = l_3 = 30$ , and  $L_1 = L_3 = 50$ . Thus, we have two allocation intervals  $A_{11} = [0, 60]$  and  $A_{12} = [70, 100]$  for piece type 1, and the allocation interval  $A_{21} = [0, 100]$  for piece type 2. Furthermore, we have  $c_1 > c_2$  which implies

$$i_1(x) = \begin{cases} 1, & \text{for } x \in [0, 60] \cup [70, 100], \\ 2, & \text{for } x \in (60, 70). \end{cases}$$

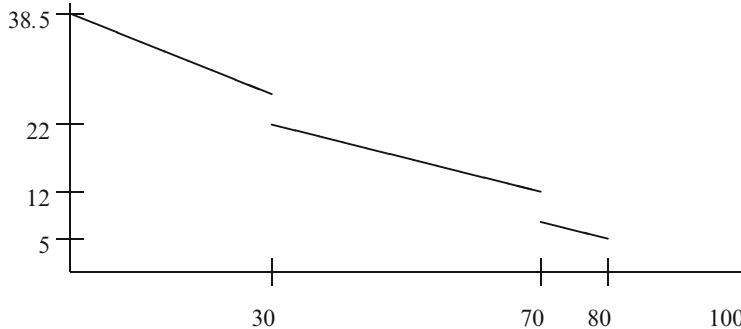
Function  $ub_1(x)$  is drawn Fig. 9.3.

Using an artificial piece type 3 with  $c_3 = 0$ , we obtain the following indices  $i_2(x, t)$  in dependence of  $x \in [0, 100]$ :

$$\begin{aligned} 80 < x \leq 100 : i_2(x, t) &= 3 \quad \text{for } t \in [x, 100], \\ 70 < x \leq 80 : i_2(x, t) &= 2 \quad \text{for } t \in [x, 100], \\ 30 < x \leq 70 : i_2(x, t) &= \begin{cases} 1 & \text{for } t \in [70, 100], \\ 2 & \text{for } t \in [x, 70], \end{cases} \\ 0 \leq x \leq 30 : i_2(x, t) &= \begin{cases} 1 & \text{for } t \in [70, 100], \\ 2 & \text{for } t \in (60, 70), \\ 1 & \text{for } t \in [x, 60]. \end{cases} \end{aligned}$$



**Fig. 9.3** Bounding function  $ub_1$  of Example 9.2



**Fig. 9.4** Bounding function  $ub_2$  of Example 9.2

The resulting function  $ub_2$  is depicted in Fig. 9.4. ■

To compute the optimal value  $v(L)$ , we consider single patterns or subsets of patterns in the following. Clearly, any feasible pattern of interval  $[0, l]$  yields a lower bound for  $v(l)$ . We denote by  $h(l)$  the currently best lower bound for an optimal pattern of segment  $[0, l]$ . Function  $h$  possesses similar properties like  $v$ , and therefore, for any  $h$  a sequence of coordinates  $\beta_k$ ,  $k = 0, 1, \dots, \delta$ , exists where  $h$  has a jump discontinuity or where the ascent is changing. Any two neighboring points  $\beta_k$ ,  $\beta_{k+1}$  define a so-called *basic* or *reference interval*  $B_k := [\beta_k, \beta_{k+1})$ .

Due to the definition of the  $\beta_k$ , for every interval  $B_k = [\beta_k, \beta_{k+1})$  there exists a representation of  $h$  in the form

$$h(x) = h_k^0 + h_k^a \cdot (x - \beta_k), \quad x \in B_k,$$

where

$$h_k^0 := h(\beta_k), \quad h_k^a := h'(\frac{1}{2}(\beta_k + \beta_{k+1})).$$

Additionally, we can assume that for the whole interval  $B_k$  a uniquely determined sequence of allocated pieces exists which provides all  $h$ -values. In view of the solution methods presented below, let  $i_k$  indicate the index of the last (right-most) piece in the sequence of allocated pieces. Provided that the last piece  $T_{i_k}$  has a uniquely determined allocation point for the whole interval  $B_k$ , then it is denoted by  $\xi_k$ .

*Example 9.3* We consider again the scenario of Example 9.1, now after allocating a piece of type  $T_i$  at allocation point  $0 \in A_{i,1}$  for  $i = 1, 2$ , and inducing monotony of  $h$ . We obtain the yield function  $h$  as shown in Fig. 9.5. For segment  $[0, x]$  with  $x \in [50, 80]$  the cutting of piece type  $T_1$  with length 50 is favorable in comparison to cut a piece of type  $T_2$  with length  $x$ . As can be seen in Fig. 9.2, the optimal value function  $v$  does not coincide with  $h(x)$  for  $x > 57.5$ . ■

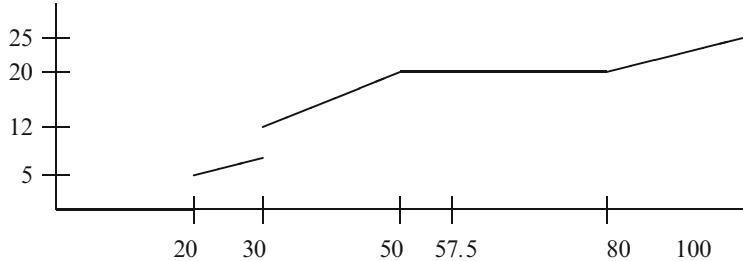


Fig. 9.5 Yield function  $h$  of Example 9.3

### 9.1.5 Allocating a Single Piece

In order to solve the allocation problem, we use a strategy similar to the *Longest Path Method* (Sect. 2.3). To this end, we construct an optimal pattern by sequentially allocating single pieces. However, the number of piece sequences which have to be considered is reduced applying bounding and dominance tests.

Because of the variability of the piece length, we cannot consider each possible allocation point separately. Instead, it is necessary to consider intervals of allocation points at the same time. To do so, we use the reference intervals  $B_k$  determined by the current yield function  $h$  as possible allocation points for the next piece..

In the following, we consider the allocation of a single piece  $T_i$  for all allocation points  $\eta \in B_q = [\beta_q, \beta_{q+1})$ . Two cases have to be distinguished.

#### 9.1.5.1 Allocating a Piece $i$ with $c_i \geq h_q^a$

We consider a piece of type  $i \in I$  with  $c_i \geq h_q^a$ . In order to find a pattern whose value is as large as possible, we have to place  $T_i$  at a feasible allocation point as small as possible and with length as long as possible because of  $c_i \geq h_q^a$ . Clearly, all such possibilities have to be considered. Hence, the allocation points are  $\eta := \beta_q$  if some  $k \in K_i$  exists with  $\beta_q \in A_{ik}$  and  $\omega_{ik} - \beta_q \geq l_i$ , as well as  $\eta := \alpha_{ik}$  for all  $k \in K_i$  with  $\beta_q < \alpha_{ik} < \beta_{q+1}$ . Without loss of generality, let  $\eta_1, \dots, \eta_\pi$  denote the appropriate allocation points such that  $\beta_q \leq \eta_1 < \dots < \eta_\pi < \beta_{q+1} =: \eta_{\pi+1}$  holds.

The length  $\ell_i$  of piece  $T_i$ , which is placed at allocation point  $\eta_p$ , is limited from above by  $\omega_p := \min\{\omega_{ik}, \eta_p + L_i\}$ . Allocating length  $\ell_i$  leads to a pattern for the intervals  $[0, x]$  with  $x \in [\eta_p + l_i, \omega_p]$ . Therefore, the following updating formula has to be applied for  $p = 1, \dots, \pi$ :

$$h(x) := \max\{h(x), h(\eta_p) + c_i(x - \eta_p)\} \quad \text{for } x \in [\eta_p + l_i, \omega_p]. \quad (9.4)$$

Besides the allocation of  $T_i$  with fixed allocation point and variable length, the allocation of  $T_i$  with maximum length and variable allocation point has to be

considered if the current allocation interval  $A_{ik}$  with  $\eta_p = \alpha_{ik}$  is longer than  $L_i$ . This results in a second updating formula for  $p = 1, \dots, \pi$ :

$$h(x) := \max \{h(x), h(x - L_i) + c_i L_i\} \quad \text{for } x \in [\eta_p + L_i, \omega_p] \quad (9.5)$$

with  $\omega_p := \min\{\omega_{ik}, \eta_{p+1} + L_i\}$ . The respective changing allocation point is  $x - L_i$ .

The updating formulas (9.4) and (9.5) are based on the following observation:

**Proposition 9.5** *Let  $i \in I$  with  $c_i \geq h_q^a$ . For any feasible placement  $T_i(x, \ell)$  of piece type  $T_i$  at allocation point  $x \in B_q$  and length  $\ell$  there exists a placement  $T_i(x^*, \ell^*)$  with  $x^* = \eta_p$  for some  $p \in \{1, \dots, \pi\}$  or with  $\ell^* = L_i$  which dominates  $T_i(x, \ell)$ .*

Hence, for solving the allocation problem, in case of  $c_i \geq h_q^a$  it is necessary to perform updating rules (9.4) and (9.5) to obtain patterns which contain  $T_i$  with allocation point in  $B_q$ . Both updating rules can be combined to save computational effort as follows:

**Update of  $h$  if  $c_i \geq h_q^a$**

- (1) Set  $\underline{x} := 0$ ,  $p := 1$ .
- (2) Packing of  $T_i$  with allocation point in  $[\eta_p, \eta_{p+1}]$ :  
Define  $\underline{x} := \max\{\underline{x}, \eta_p + l_i\}$ ,  $\bar{x} := \min\{\omega_{ik}, \eta_p + L_i\}$ , and  
 $h(x) := \max\{h(x), h(\eta_p) + c_i(x - \eta_p)\}$  for all  $x \in [\underline{x}, \bar{x}]$ .  
If  $\bar{x} = \omega_{ik}$ , then go to (3).  
Set  $\bar{x} := \min\{\omega_{ik}, \alpha_{i,k+1} + L_i\}$  and  
 $h(x) := \max\{h(x), h(x - L_i) + c_i L_i\}$  for all  $x \in (\eta_p + L_i, \bar{x}]$ .
- (3) If  $p \geq \pi$ , then stop.  
Set  $p := p + 1$ ,  $\underline{x} := \bar{x}$ , and go to (2).

### 9.1.5.2 Allocating a Piece $i$ with $c_i < h_q^a$

Let  $i \in I$  with  $c_i < h_q^a$ . In difference to the opposite case, the packing of  $T_i$  should be done with a length as short as possible and allocation point as large as possible. Hence, the following updating formula has to be applied for  $p = 1, \dots, \pi$ :

$$h(x) := \max \{h(x), h(x - l_i) + c_i l_i\} \quad \text{for } x \in [\eta_p + l_i, \omega_p] \quad (9.6)$$

where  $\omega_p := \min\{\omega_{ik}, \eta_{p+1} + l_i\}$  and  $A_{ik} = [\alpha_{ik}, \omega_{ik}]$  is the allocation interval related to  $\eta_p$ . The allocation point is always given by  $x - l_i$ . Because of  $c_i < h_q^a$  a packing of  $T_i$  with length  $\ell_i > l_i$  is not meaningful since resulting patterns are dominated.

**Proposition 9.6** *Let  $i \in I$  with  $c_i < h_q^a$ . For any feasible packing  $T_i(x, \ell)$  of piece type  $T_i$  at allocation point  $x \in B_q$  and length  $\ell$  there exists a packing  $T_i(x^*, \ell^*)$  with  $x^* = \beta_{q+1}$  or with  $\ell^* = \ell_i$  which dominates  $T_i(x, \ell)$ .*

Hence, in case of  $c_i < h_q^a$  only formula (9.6) has to be applied to regard optimal patterns which contain  $T_i$  with allocation point  $x \in B_q$ . Note that the updating rules (9.4)–(9.6) cause, in general, a change of the basic intervals  $B_k$  for  $k \geq q$ .

Note that also those segments have to be considered within the allocation problem which correspond to waste or unused parts of the raw material. Such a segment can arise, for instance, if some higher-valued piece is allocated and the remaining lengths (of the raw material) is not long enough such that another piece fits although the respective quality demand could be fulfilled [cf. Example 9.1,  $x \in (50, 57.5)$ ]. To handle this aspect, the monotony of  $h$  has to be enforced by using another updating rule:

$$h(x) := \max\{h(x), \lim_{\epsilon \rightarrow +0} h(x - \epsilon)\}, \quad x \geq \beta_q + l_i. \quad (9.7)$$

Summarizing, in any case an updating rule of the following form results:

$$h(x) := \max\{h(x), c_0 + c_1 \cdot (x - x_0)\} \quad \text{for } x \in [a, b], \quad (9.8)$$

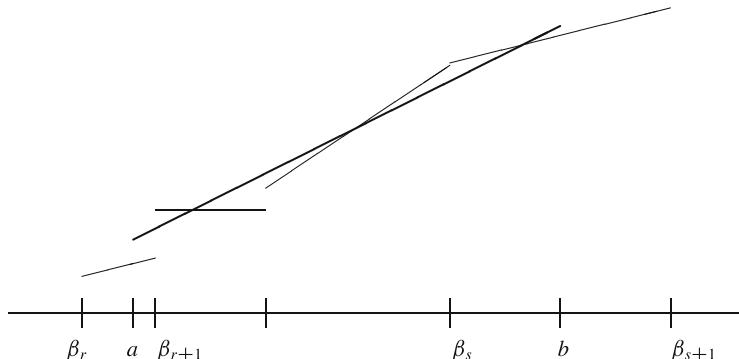
where the parameters  $a, b, c_0, c_1$ , and  $x_0$  are known. For a current partition of  $[0, L]$  into intervals  $B_k$  let indices  $r$  and  $s$  be defined according to

$$a \in [\beta_r, \beta_{r+1}) \quad \text{and} \quad b \in (\beta_s, \beta_{s+1}].$$

When applying formula (9.8) different situations have to be handled as depicted in Fig. 9.6. Therein, the thick line represents the term  $c_0 + c_1 \cdot (x - x_0)$ ,  $x \in [a, b]$ .

The restriction of profit function  $h$  onto basic interval  $B_k$  is denoted by  $h_k$ . It is determined by coefficients  $h_k^0$  and  $h_k^a$  with

$$h_k(x) = h_k^0 + h_k^a \cdot (x - \beta_k), \quad x \in B_k = [\beta_k, \beta_{k+1}).$$



**Fig. 9.6** Possible situations which can occur during the update of  $h$

Related to formula (9.8), we define

$$g(x) = c_0 + c_1 \cdot (x - x_0), \quad x \in [a, b]$$

to denote the new yield function obtained after allocating a piece. The update according to (9.8) can be realized in a certain way: all intervals  $B_k$  for  $k = r, \dots, s$  are considered successively. One variant is proposed in the following algorithm.

### Update of $h$ according to (9.8)

For  $k = r, \dots, s$ :

- Set  $\lambda_l := \beta_k$ ,  $\lambda_r := \beta_{k+1}$ .  
If  $k = r$ , then  $\lambda_l := a$ .  
If  $k = s$ , then  $\lambda_r := b$ .
- If  $h_k(\lambda_l) \geq g(\lambda_l)$  and  $h_k(\lambda_r) \geq g(\lambda_r)$ , then no changes of  $B_k$  are necessary.
- If  $h_k(\lambda_l) \leq g(\lambda_l)$  and  $h_k(\lambda_r) \leq g(\lambda_r)$ , then replace  $h_k$  by  $g$  in  $[\lambda_l, \lambda_r]$ . In case of  $k = r$  or  $k = s$  the interval  $B_k$  has to be replaced by two intervals, respectively.
- If  $h_k(\lambda_l) > g(\lambda_l)$  and  $h_k(\lambda_r) < g(\lambda_r)$ , then the intersection point  $\bar{x}$  of  $h_k$  and  $g$  has to be computed according to  $\bar{x} := (h_k^0 - h_k^a \cdot \lambda_l - c_0 + c_1 \cdot x_0) / (c_1 - h_k^a)$  and  $B_k$  is dissected into two intervals  $[\lambda_l, \bar{x}]$  and  $[\bar{x}, \lambda_r]$ .
- If  $h_k(\lambda_l) < g(\lambda_l)$  and  $h_k(\lambda_r) > g(\lambda_r)$ , then perform in a similar way.

As a result of the update of  $h$  the number of basic intervals  $B_k$  increases, in general. Since these intervals are used as reference intervals in the following, their number  $\delta$  should be held as small as possible. To this end, neighboring intervals should be united, if possible. For instance, as follows: let  $B_k = [\beta_k, \beta_{k+1}]$  and  $B_{k+1} = [\beta_{k+1}, \beta_{k+2}]$  be two neighboring intervals, then they can be united if  $h_k^a = h_{k+1}^a$  and  $h_{k+1}^0 = h_k^0 + h_k^a \cdot (\beta_{k+1} - \beta_k)$  hold and the types of the last pieces in the respective sequences coincide as well as the related allocation points. The connection of intervals can be done immediately during the updating process.

Moreover, the application of formula (9.7) can also be performed during the update of  $h$ . In the particular case that

$$\lim_{\epsilon \rightarrow +0} h(\beta_{k+1} - \epsilon) > \lim_{\epsilon \rightarrow +0} h(\beta_k - \epsilon) > h(\beta_k) = \lim_{\epsilon \rightarrow +0} h(\beta_k + \epsilon),$$

holds, which possibly occurs during the updating process for  $B_{k-1}$  and  $B_k$ , then the interval  $B_k$  has to be split into two subintervals to enforce the monotony of  $h$ . In the other cases, one has to perform in a similar way.

### 9.1.6 Solution Approaches

In this subsection we provide two solution approaches for the problem with pieces of variable length where we apply the update rules discussed in the previous subsection.

#### 9.1.6.1 Branch and Bound Algorithm

The b&b algorithm presented in Fig. 9.7 is based on the LIFO strategy. Appropriate upper bounds, denoted by  $\tilde{u}$ , are defined in Sect. 9.1.4 satisfying the inequality  $\tilde{u}(x) \geq v(L) - v(x)$  for each allocation point  $x \in [0, L]$ . Without loss of generality, we assume again that  $c_1 \geq c_2 \geq \dots \geq c_m$  holds. Note that branching is made with respect to

- the basic intervals and
- the pieces which can be allocated next according to the chosen basic interval.

#### B&B algorithm for 1D cutting problems with variable lengths

(1) *Initialization*

Set  $B_0 := [0, L]$ ,  $\alpha_0 := 0$ ,  $\beta_0 := 0$ ,  $v_0 = 1$ ,  $\mu := 0$ .

(2) *Selection of the next basic interval for branching*

Select the basic interval  $B_j$  with largest  $\beta_j$ -coordinate and  $v_j = 1$ . If such an interval  $B_j$  does not exist, then go to (5). Otherwise, set  $\tilde{B}_\mu := B_j$  and  $v_j := 0$ . If  $\max_{x \in \tilde{B}_\mu} \{h(x) + \tilde{u}(x)\} \leq h(L)$ , then go to (2). If  $L \in \tilde{B}_\mu$ , then go to (4).

(3) *Selection of the next piece type for branching*

Select the next piece type  $i_\mu \in I$  (not considered so far for branching) which can be allocated at an allocation point in  $\tilde{B}_\mu$ . If such a piece does not exist, then set  $\mu := \mu - 1$  and go to (2).

If  $\max_{x \in \tilde{B}_{\mu-1}} \{h(x) + c_i l_i + \tilde{u}(x + l_i)\} \leq h(L)$ , then go to (3).

(4) *Allocation of pieces of type  $i_\mu \in I$*

If  $L \in \tilde{B}_\mu$ , then allocate (one after another) all feasible pieces at allocation point  $\eta_\mu$  with all feasible lengths: a new partition of  $\tilde{B}_\mu = [\beta_\mu, L]$  results; set  $v_j := 1$  for all  $B_j$  with  $\beta_j \geq \beta_\mu$ . Otherwise, if  $L \notin \tilde{B}_\mu$ , then allocate pieces of type  $i_\mu$  for all feasible lengths and all allocation points within  $\tilde{B}_\mu$  and update  $h$  in accordance with (9.4)–(9.7). If  $h$  is increased in some interval  $B_{j'}$  during the updating process, then set  $v_{j'} := 1$ . If  $h$  has not been increased anywhere, then go to (3), otherwise set  $\mu := \mu + 1$  and go to (2).

(5) *Identify an optimal pattern.*

**Fig. 9.7** A sketch of a branch-and-bound algorithm

The index  $\mu$  denotes the branching depth in the algorithm. For any basic interval  $B_j$ , the label  $v_j$  denotes whether the basic interval  $B_j$  is already investigated for further branching (then  $v_j = 0$ ), or if it still has to be considered (then  $v_j = 1$ ). Furthermore,  $\tilde{B}_\mu = [\tilde{\beta}_\mu, \hat{\beta}_\mu]$  denotes the current basic interval. Note that the initial basic interval  $\tilde{B}_0 = [0, L]$  is reduced in Step (4) when the first pieces are placed, and the same can happen for other basic intervals during the algorithm. The branching strategy presented here uses the LIFO principle (depth first search), but modifications are obviously possible.

### 9.1.6.2 Forward Dynamic Programming Algorithm

Besides branch-and-bound algorithms, a *Forward Dynamic Programming* (FDP) method can be applied as well for solving the considered cutting problem. The general principle of the FDP, presented in Fig. 9.8, is similar to the FDP (longest path) algorithm for the knapsack problem in Sect. 2.3. It computes optimal values  $v(x)$  and corresponding patterns for each  $x \leq L$  where  $x$  is successively increased. To this end, the values  $v(x)$  are obtained by updating the function  $h$  in such a way that, starting from a known optimal solution, feasible pieces are placed with all suitable lengths  $l$  to possibly get a better solution for larger segments of the raw material.

In the FDP algorithm intervals of allocation points are considered as well as in the b&b approach. In the latter, after investigating a basic interval, say  $B_j$ , other intervals  $B_{j'}$  with  $\beta_{j'+1} \leq \beta_j$  have to be considered, in general, due to backtracking. Therefore it can happen that  $B_j$  (or a subset of it) has to be considered again if  $h_j$  has

| <b>FDP algorithm for 1D cutting problems with variable lengths</b> |   |
|--|---|
| (1)  | <i>Initialization</i><br>Set $B_0 := [0, L]$ , $\alpha_0 := 0$ , $\beta_0 := 0$ . Allocate (one after another) all feasible pieces at $\xi = 0$ with all feasible lengths. A first partition of $[0, L]$ , i.e., a first finite sequence $(\beta_j)$ is obtained.<br>If $\beta_1 > \min\{\min\{l_i \mid i \in I\}, \min\{\alpha_{i,1} \mid \alpha_{i,1} > 0, i \in I\}\} =: \tilde{\beta}$ , then split $B_0$ into $[0, \beta]$ and $[\tilde{\beta}, \beta_1]$ , and renumber. Set $j := 0$ . |
| (2)  | <i>Selection of the next basic interval</i><br>Set $j := j + 1$ . If $\beta_j + \min\{l_i \mid i \in I\} > L$ , then go to (4).<br>If $\max_{x \in B_j} \{h(x) + u(x)\} \leq h(L)$ , then go to (2).  |
| (3)  | <i>Allocating pieces at allocation points in <math>B_j</math></i><br>Allocate (one after another) all feasible pieces of type $i$ at every allocation point $\xi \in B_j$ and with all feasible lengths according to (9.4)–(9.7). A new partition of $[0, L]$ , i.e., a new finite sequence $(\beta_j)$ , is obtained, but changes of the $\beta_{j'}$ -values are only possible for $j' \geq j + 1$ . Go to (2).   |
| (4)  | <i>Identify an optimal pattern.</i>   |

**Fig. 9.8** A sketch of an FDP algorithm

(partially) been increased. However, using the FDP approach, the basic interval  $B_j$  is considered exactly once, namely if  $h(x) = v(x)$  holds for all  $x \in B_j$ . This can also be guaranteed in a b&b algorithm when an appropriate branching strategy (based on breadth first search) is used.

Note that during the allocation of a piece at allocation points in  $B_j$  further tests with upper bounds can be used as in the b&b algorithm. If piece  $i$  is feasible and  $\beta_j + l_i < \beta_{j+1}$  holds, then  $B_j$  can be split (i.e.,  $\beta_{j+1}$  can be reduced) in a suitable way. Moreover, during the updating process only partitions of  $[0, L - \min\{l_i \mid i \in I_L\}]$  have to be considered since there is no feasible allocation point within the interval  $(L - \min\{l_i \mid i \in I_L\}, L]$ .

In the worst case,  $O(m|S_{ap}|)$  updates according to (9.4)–(9.7) have to be done in Step (3) of the algorithm where  $S_{ap}$  denotes the set of potential allocation points induced by  $l_i$  and  $L_i$ ,  $i \in I$ , and the allocation intervals. A single update requires at most  $O(L_i)$  time.

An advantage of the FDP approach can be the relatively constant expense needed to solve instances of nearly the same size which, moreover, can be well estimated in advance (pseudo-polynomiality of the algorithm). In general, this is not possible for a b&b method where some examples can require much more computation time as in average. However, in general, good (near optimal) solutions are found quickly by a b&b algorithm with LIFO strategy so that a termination after a predefined time span is reasonable for online scenarios.

### 9.1.7 Example

With this example, which is solved by the b&b algorithm, we illustrate in particular the change of the  $h$ -function towards the optimal value function  $v$ . The following input data are given:

- the length  $L = 205$  of the board (raw material),
- technological parameters: kerf  $\phi_s = 5$ , size of left and right trimming cut  $\phi_l = \phi_r = 5$  (initial and end chop cut), the minimum distance between two cut positions  $\phi_m = 20$ ,
- piece type  $i = 1$ :  $l_1 = 40, L_1 = 200, c_1 = 0.4$ ,
- piece type  $i = 2$ :  $l_2 = 30, L_2 = 200, c_2 = 0.3$ ,
- a defective region in  $[100, 125]$ ,
- quality demands: the defect is not allowed for piece type 1, but for piece type 2.

According to the input data, we obtain three allocation intervals, two for piece type 1 and one for the second piece type:

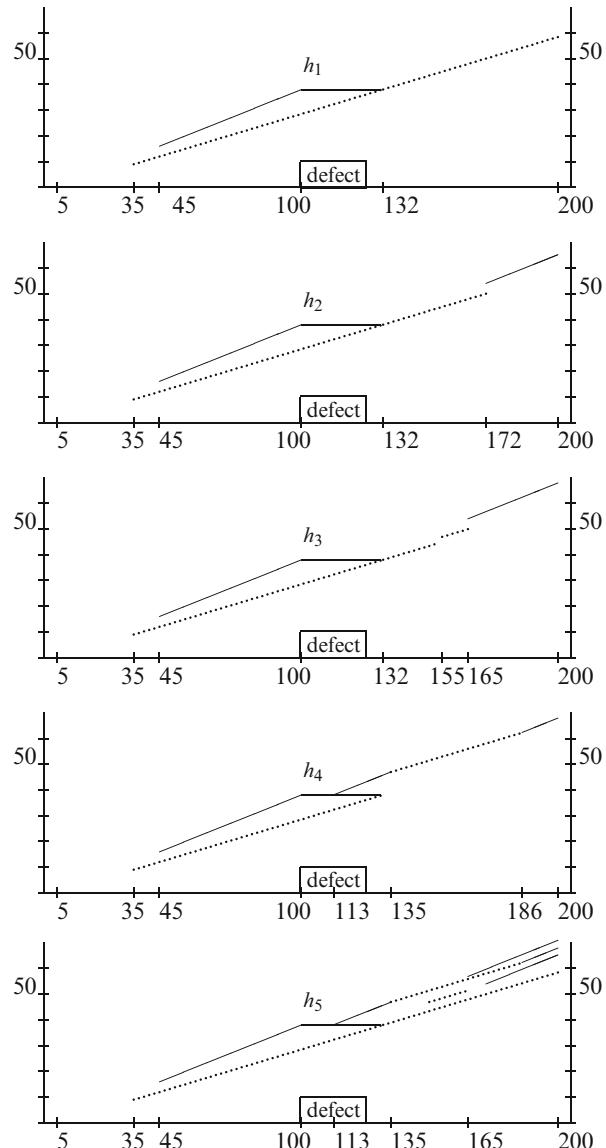
$$A_{11} = [5, 100], \quad A_{12} = [125, 200], \quad A_{21} = [5, 200].$$

To initialize the solution process, somewhat different to the LIFO strategy, we allocate both piece types at  $\eta = 5$  according to the demanded initial chop cut. After

that placing of both piece types function  $h_1$  results, shown in Fig. 9.9 and Table 9.1. Note that within the figures, the respective  $h$ -function is always the maximum-function of the drawn graphs.

To better illustrate the effect of a real kerf, we use a relatively large value in comparison to  $l_1$ ,  $l_2$ , and  $L$  within the example. As potential cut positions, we consider only integer coordinates according to the input data.

**Fig. 9.9** Yield functions  
 $h_1, \dots, h_5$



**Table 9.1** Yield functions  
 $h_1, \dots, h_5$

|        | $k$ | $B_k$     | $i_k$ | $h_k^0$ | $h_k^a$ | $t_k$ | $\eta_k$ | $\theta_k$ |
|--------|-----|-----------|-------|---------|---------|-------|----------|------------|
| $h_1:$ | 1   | [0,35)    | 0     | 0.0     | 0.0     | —     | 0        | 2          |
|        | 2   | [35,45)   | 2     | 9.0     | 0.3     | var   | 5        | 0          |
|        | 3   | [45,100]  | 1     | 16.0    | 0.4     | var   | 5        | 0          |
|        | 4   | (100,132) | 0     | 38.0    | 0.0     | —     | 100      | 0          |
|        | 5   | [132,200] | 2     | 38.1    | 0.3     | var   | 5        | 0          |
| $h_2:$ | 1   | [0,35)    | 0     | 0.0     | 0.0     | —     | 0        | 2          |
|        | 2   | [35,45)   | 2     | 9.0     | 0.3     | var   | 5        | 0          |
|        | 3   | [45,100]  | 1     | 16.0    | 0.4     | var   | 5        | 0          |
|        | 4   | (100,132) | 0     | 38.0    | 0.0     | —     | 100      | 0          |
|        | 5   | [132,172) | 2     | 38.1    | 0.3     | var   | 5        | 1          |
|        | 6   | [172,200] | 1     | 54.1    | 0.4     | var   | 132      | 0          |
| $h_3:$ | 1   | [0,35)    | 0     | 0.0     | 0.0     | —     | 0        | 2          |
|        | 2   | [35,45)   | 2     | 9.0     | 0.3     | var   | 5        | 0          |
|        | 3   | [45,100]  | 1     | 16.0    | 0.4     | var   | 5        | 0          |
|        | 4   | (100,132) | 0     | 38.0    | 0.0     | —     | 100      | 2          |
|        | 5   | [132,155) | 2     | 38.1    | 0.3     | var   | 5        | 2          |
|        | 6   | [155,165) | 2     | 47.0    | 0.3     | var   | 125      | 2          |
|        | 7   | [165,200] | 1     | 54.0    | 0.4     | var   | 125      | 2          |
| $h_4:$ | 1   | [0,35)    | 0     | 0.0     | 0.0     | —     | 0        | 2          |
|        | 2   | [35,45)   | 2     | 9.0     | 0.3     | var   | 5        | 0          |
|        | 3   | [45,100]  | 1     | 16.0    | 0.4     | var   | 5        | 0          |
|        | 4   | (100,113) | 0     | 38.0    | 0.0     | —     | 100      | 2          |
|        | 5   | [113,136) | 2     | 38.2    | 0.4     | min   | -        | 0          |
|        | 6   | [136,186) | 2     | 47.3    | 0.3     | var   | 105      | 0          |
|        | 7   | [186,200] | 1     | 62.4    | 0.4     | var   | 125      | 2          |
| $h_5:$ | 1   | [0,35)    | 0     | 0.0     | 0.0     | —     | 0        | 2          |
|        | 2   | [35,45)   | 2     | 9.0     | 0.3     | var   | 5        | 2          |
|        | 3   | [45,100]  | 1     | 16.0    | 0.4     | var   | 5        | 2          |
|        | 4   | (100,113) | 0     | 38.0    | 0.0     | —     | 100      | 2          |
|        | 5   | [113,136) | 2     | 38.2    | 0.4     | min   | -        | 2          |
|        | 6   | [136,165) | 2     | 47.3    | 0.3     | var   | 105      | 2          |
|        | 7   | [165,200] | 1     | 57.0    | 0.4     | var   | 125      | 2          |

Parameter  $t_k$  indicates whether piece  $T_{i_k}$  is allocated with fix or variable length. Index  $\theta_k$  is used in a somewhat different form in comparison to  $v_k$  in the b&b algorithm. If  $\theta_k = 0$ , then both piece types still have to be considered for branching with respect to the corresponding basic interval. Otherwise,  $\theta_k$  indicates which piece types have already been considered for constructing new patterns having their allocation point in  $B_k$ . Moreover, the entries *var* and *min*, used in Table 9.1, indicate that the piece of type  $i_k$  has a varying length  $x - \eta_k$  or is placed with minimal length and, therefore, has varying allocation point  $x - l_{i_k}$ , respectively.

According to the LIFO strategy, basic interval  $B_5 = [132, 200]$  is chosen as the next reference interval. Piece type  $i = 1$  is considered first. Because of  $c_1 = 0.4 > h_{15} = 0.3$  piece type 1 is placed at  $\eta = 132$  and length  $\ell$  in  $[40, 68]$ . This leads to an increased  $h$ -function in  $[172, 200]$  and, therefore, to function  $h_2$ .

The consideration of reference interval  $[172, 200]$  yields no improvement. As the next reference interval we choose  $[132, 172)$  and piece type  $i = 2$ . Again, no improvement is obtained.

Then, according to the LIFO strategy,  $B_4 = [100, 132)$  becomes the next reference interval. Because of  $i_4 = 0$  (since  $B_4$  is caused by the monotony property) an unusable region results. In the case of a cut at position 100, then the position of the right neighbored cut has to be not smaller than 120 due to  $\phi_m = 20$ . Since the kerf has width 5, and because of  $\alpha_{12} = 125$  the placement of a further piece of type 1 is possible. Piece type  $i = 1$  with allocation point  $\eta = 125$  yields an improvement in  $[165, 200]$ .

Afterwards we obtain with  $i = 2$  and  $\eta = 125$  an improvement of the yield in  $[155, 165)$ . As an intermediate result, we have  $h_3$ .

For  $B_3 = [45, 100]$  and piece type  $i = 1$  we obtain no improvement, but for piece type  $i = 2$ . Because of  $h(78) = 0.4 \cdot 73 = 29.2$  and  $c_2 l_2 = 9$ , the allocation of a piece of type 2 within  $[78 + \phi_s, 113]$  yields the value  $38.2 > h_3(100)$ . Moreover, placing a piece with length  $l_2$ , we obtain a larger value for  $\ell \in [113, 135]$ . The corresponding allocation point is  $\eta = \ell - 30$ , i.e. varying.

Since  $\omega_{11} = 100$  and limited by the upper margin of the current reference interval, only the lower valued piece type can be enlarged from the maximal allocation point  $\eta = 100$ . The allocation of a piece of type 2 within the segment  $[100 + \phi_s, \ell]$  of the raw material yields the improved values of function  $h_4$  for  $\ell \in (135, 185]$ .

Finally, we obtain function  $h_5$ , shown in Fig. 9.9 and Table 9.1, which coincide with the optimal value function  $v$ . Hence, the maximal value is  $v = 71$ . A corresponding optimal pattern can be obtained using the information related to  $h_5$ :

- first cut at  $p_1 = 200$ , i.e., we have a kerf in  $[200, 205]$ ,  $v(200) = 71$ . Because of  $\eta_6 = 125$  and regarding  $\phi_s = 5$ , we obtain:
- second cut at  $p_2 = 120$  with kerf in  $[120, 125]$ . A piece of type 1 with length 75 is obtained from board segment  $[125, 200]$ . It holds  $v(200) - c_1(75) = 71 - 30 = 41 = v(120)$ . The last (right-most) piece of type  $i_5 = 2$  belonging to  $\ell = 120$  has minimal length. Therefore:
- third cut at  $p_3 = 120 - l_2 - \phi_s = 85$  (kerf in  $[85, 90]$ ). Board segment  $[90, 120]$  yields a piece of type 2 with minimal length 30. Because of  $v(120) - c_2(30) = 41 - 9 = 32 = v(85) = c_1(80)$ ,  $\eta_3 = 5$ , and  $i_3 = 1$ , we obtain:
- cut 4 at  $p_4 = 0$ . From board segment  $[5, 85]$  we get a piece of type 1 with length 80 (kerf in  $[0, 5]$ ). Obviously,  $v(85) - c_1(80) = 32 - 32 = 0$ .

Further optimal patterns can be obtained by a simultaneous ‘shifting’ of both cut points, 85 and 120, by  $\Delta$  length-units with  $\Delta \in (0, 15]$ .

## 9.2 Defective or Forbidden Regions

As a generalization of the problem considered in the previous subsection, we will address the two-dimensional case where quality demands have to be regarded. Applying guillotine cuts, rectangular pieces have to be obtained from heterogeneous raw material which contains defective regions.

For a simpler description, we restrict ourselves to the case that all desired pieces should not have any defective part, or with other words, have to be produced from defect-free material. For more general cases including problems with different quality demands and applications in wood cutting, we refer to [1] and [5]. Moreover, we do not regard a real kerf.

### 9.2.1 Problem Formulation

Among various similar problems, we consider in our description a problem arising in hardwood cutting. Obviously, the wooden raw material possesses, in general, different characteristics such as non-desired coloring, knotholes, wane, etc. Depending on the quality demands, some of them may be allowed or not allowed to occur on the products.

In the following, we consider a cutting problem where no defects (predefined characteristics) are allowed to be on the products. Let a rectangle of length  $L$  and width  $W$  represent a minimal (rectangular) envelopment of the wooden raw material (board). The task is to cut pieces of rectangular type  $T_i$ ,  $i \in I = \{1, \dots, m\}$ , in such way that the total yield is maximized. A piece of type  $i$  has length  $\ell_i$ , width  $w_i$  and value (profit, yield)  $e_i$ . The unique quality demand for all piece types is simply that they have to be obtained from defect-free parts of the board. In our considerations, the cutting technology is restricted to be of *exact 3-stage guillotine cutting* type with vertical (chop) cuts in the first stage.

Since only guillotine cuts are permitted, we can assume without loss of generality that all *defective regions* (defects) of the board are given as union of rectangles. We describe the defects by the rectangles

$$D_q = D_q(a_q, b_q, c_q, d_q) := \{(x, y); a_q \leq x \leq c_q, b_q \leq y \leq d_q\}, \quad q \in Q := \{1, \dots, p\}.$$

To unify the description, we assume that the (degenerated) rectangles

$$D_l = D_l(0, 0, 0, W), \quad D_r = D_r(L, 0, L, W), \quad D_t = D_t(0, W, L, W), \quad D_b = D_b(0, 0, L, 0)$$

also belong to the set of defects representing the left, right, upper (top), and lower (bottom) edge (border) of the board, respectively. Furthermore, we assume again that all input data are integers. This is possible due to a sufficiently small unit of measure, e.g.. 0.1 mm.

In a general scenario the wooden board, to be cut into desired rectangular products, has curved edges, frequently containing wane. To differentiate defective regions along the bottom and top edge from defects within the interior of the board, let

$$Q_t := \{q \in Q : d_q = W\}, \quad Q_b := \{q \in Q : b_q = 0\}$$

denote respective index sets.

Although the representation of defects describing the bottom and top edge by means of rectangles is possible (and we will use it, too), we propose another, more compact way by means of two piecewise constant functions

$$\psi : [0, L] \rightarrow [0, W], \quad \psi(x) := \min\{b_q : x \in D_q, q \in Q_t\} \quad (\text{upper edge})$$

and

$$\omega : [0, L] \rightarrow [0, W], \quad \omega(x) := \max\{d_q : x \in D_q, q \in Q_b\} \quad (\text{lower edge})$$

to define the curved edges in length direction to keep small the number of defects. The representation of the defect information is sketched in Fig. 9.10.

The discontinuity points of the stair-function  $\psi$  are denoted by

$$\mu_q, \quad q \in Q_\psi := \{0, 1, \dots, p_\psi\}, \quad \text{assuming } 0 = \mu_0 < \mu_1 < \dots < \mu_{p_\psi} = L,$$

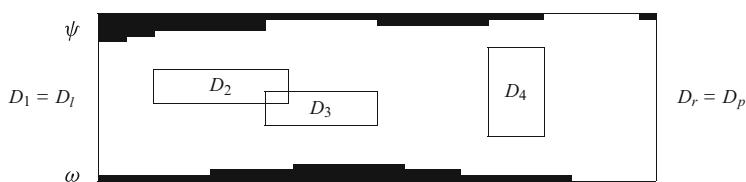
those of  $\omega$  by

$$v_q, \quad q \in Q_\omega := \{0, 1, \dots, p_\omega\}, \quad \text{assuming } 0 = v_0 < v_1 < \dots < v_{p_\omega} = L.$$

Furthermore, as abbreviations, we define

$$\psi_q := \psi(x), \quad x \in (\mu_{q-1}, \mu_q), \quad q \in Q_\psi \setminus \{0\}, \quad \psi_0 := 0, \quad \psi_{p_\psi+1} := 0,$$

$$\omega_q := \omega(x), \quad x \in (v_{q-1}, v_q), \quad q \in Q_\omega \setminus \{0\}, \quad \omega_0 := W, \quad \omega_{p_\omega+1} := W.$$



**Fig. 9.10** Description of defects by *rectangles*

Moreover, without loss of generality, we assume that the following conditions are fulfilled by  $\psi$  and  $\omega$  so that the edges of the board are represented in a manner suitable for the solution approaches:

- $\psi(x) \geq \omega(x), \quad 0 < x < L,$
- $\psi_{q-1} < \psi_q, \psi_q > \psi_{q+1} \Rightarrow \mu_q - \mu_{q-1} \geq \min_{i \in I} \ell_i, \quad q = 1, \dots, p_\psi,$
- $\omega_{q-1} > \omega_q, \omega_q < \omega_{q+1} \Rightarrow v_q - v_{q-1} \geq \min_{i \in I} \ell_i, \quad q = 1, \dots, p_\omega.$

Obviously, considering an already *trimmed board*, then we have  $\psi(x) = W$  and  $\omega(x) = 0$  for  $x \in (0, L)$ .

### 9.2.2 Basic Recursion

Since the cutting is performed in three stages, we define a *segment* (of the board)

$$(x, 0, x + r, W) := \{(p, q) \in \mathbb{R}^2 : x \leq p \leq x + r, 0 \leq q \leq W\}$$

of length  $r$  and position  $x$  obtained by two vertical guillotine cuts of the first stage at  $x$  and  $x + r$ . Using horizontal guillotine cuts of the second stage, we obtain a so-called *strip*

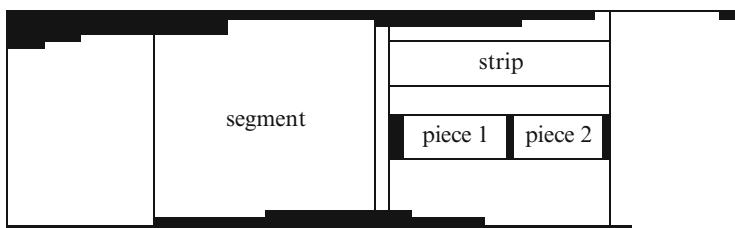
$$(x, y, x + r, y + w) := \{(p, q) \in \mathbb{R}^2 : x \leq p \leq x + r, y \leq q \leq y + w\}$$

with width  $w$  and position  $y$ .

Finally, applying again vertical cuts (now of the third stage), the desired pieces are obtained from the defect-free parts of strip  $(x, y, x + r, y + w)$ .

Due to the demand of *exact* guillotine cutting (trimming is not allowed) only pieces of width  $w$  are cut (Fig. 9.11). Since the width of a strip has to be equal to the width of a desired piece, we define non-empty index sets  $I_k$  and the different strip widths  $\bar{w}_k$  as follows:

$$I_k := \{i \in I : w_i = \bar{w}_k\} \quad \text{with} \quad \bar{w}_1 < \dots < \bar{w}_s, \quad \bigcup_{k \in K} I_k = I, \quad K := \{1, \dots, s\}.$$



**Fig. 9.11** Exact 3-stage guillotine cutting

As length  $r$  of a segment any (non-negative integer) combination of piece lengths is meaningful and its position can also be determined by the coordinates of the defects. In applications, restricted sets of segment lengths and possible positions are frequently considered. We denote by

$$R = \{r_j : j \in J := \{1, \dots, \alpha\}\} \quad \text{with} \quad r_1 < \dots < r_\alpha$$

the set of feasible segment lengths. Sometimes,

$$R := \{\ell_i : i \in I\} \quad \text{or} \quad R := \{r : r = \sum_{i \in I} \ell_i \lambda_i \leq \max_{i \in I} \ell_i, \lambda_i \in \mathbb{Z}_+, i \in I\}$$

are used. Let  $r \in R$ ,  $x \in [0, L] \cap \mathbb{Z}$ , and  $y \in [0, W] \cap \mathbb{Z}$ . If  $H(x, r, y, w)$  denotes the maximal yield obtainable from strip  $(x - r, y - w, x, y)$  and  $G(x, r, y)$  the maximal yield of the (partial) segment  $(x - r, 0, x, y)$ , then we can formulate the following recursion:

$$G(x, r, y) := 0, \quad y = 0, \dots, \bar{w}_1 - 1,$$

$$G(x, r, y) := \max\{H(x, r, y, \bar{w}_k) + G(x, r, y - \bar{w}_k) : k \in K, \bar{w}_k \leq y\}, \quad (9.9)$$

$$y = \bar{w}_1, \dots, W.$$

Moreover, if we denote the maximal yield obtainable from the part  $(0, 0, x, W)$  of the board by  $F(x)$ , then the following recursion holds:

$$F(x) := 0, \quad x = 0, \dots, r_1 - 1,$$

$$F(x) := \max\{G(x, r_j, W) + F(x - r_j) : r_j \leq x, j \in J\}, \quad x = r_1, \dots, L.$$

Hence, the value of an optimal pattern is given by  $F(L)$ .

During the computation of  $F(L)$  several (many)  $G$ - and  $H$ -values have to be computed. Fortunately, there exist various possibilities to reduce the total effort as we will see in the next subsection.

### 9.2.3 Improvements

Since the cutting has to be performed without any (horizontal) trimming cuts, the optimal value  $H(x, r, y, w)$  for strip  $(x - r, y - w, x, y)$  can be computed easily. If we have identified the disjunctive intervals in  $[x - r, x]$  which do not (partially) contain some defect  $D_q$ , then  $H(x, r, y, w)$  is the sum of the yields of these defect-free intervals with strip width  $w$ . We denote by  $h_k(l)$  the maximal value of a defect-

free strip of length  $l$  and width  $\bar{w}_k$ :

$$h_k(l) := \max\left\{\sum_{i \in I_k} e_i a_i : \sum_{i \in I_k} \ell_i a_i \leq l, a_i \in \mathbb{Z}_+, i \in I_k\right\}, \quad k \in K.$$

These values are obtainable by solving appropriate knapsack problems. In particular, they do not depend on the real position on the board. Moreover, let  $Q(x, r)$  and  $Q(x, r, y, w)$  represent the defects relevant for segment  $(x - r, 0, x, W)$  and strip  $(x - r, y - w, x, y)$ , respectively, i.e.,

$$Q(x, r) := \{q \in Q : x - r < c_q, a_q < x\},$$

$$Q(x, r, y, w) := \{q \in Q(x, r) : y - w < d_q, b_q < y\}.$$

If the intervals

$$[\underline{x}_j, \bar{x}_j], \quad j = 1, \dots, \kappa \quad (\kappa \geq 0),$$

with

$$x - r \leq \underline{x}_1, \quad \underline{x}_j < \bar{x}_j < \underline{x}_{j+1}, \quad j = 1, \dots, \kappa - 1, \quad \bar{x}_\kappa \leq x,$$

represent the defect-free parts of strip  $(x - r, y - w, x, y)$ , i.e.,

$$(\underline{x}_j, \bar{x}_j) \cap [a_q, c_q] = \emptyset \quad \text{for all } q \in Q(x, r, y, w), \quad j = 1, \dots, \kappa,$$

then, obviously,

$$H(x, r, y, \bar{w}_k) = \sum_{j=1}^{\kappa} h_k(\bar{x}_j - \underline{x}_j)$$

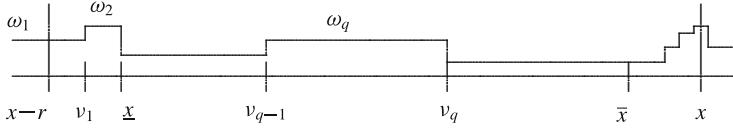
holds. Within recursion (9.9) the computation of  $H(x, r, y, \bar{w}_k)$  is not needed if it is equal to  $H(x, r, y - 1, \bar{w}_k)$ . Clearly, this happens if the set of relevant defects does not change, i.e., if

$$Q(x, r, y - 1, \bar{w}_k) = Q(x, r, y, \bar{w}_k)$$

is fulfilled. Another, but more expensive criterion consists in verifying whether the set of defect-free intervals is changed. In particular, the defect-free intervals have to possess the minimum length  $\min\{\ell_i : i \in I_k\}$ .

To transfer these ideas to the length direction, we can state the following

**Proposition 9.7** *The optimal values  $G(x, r, W)$  and  $G(x - 1, r, W)$  are equal if the condition  $H(x - 1, r, y, w) = H(x, r, y, w)$  is satisfied for all  $y = 1, \dots, W$ .*



**Fig. 9.12** Definition of set  $Y_d$

In case of  $Q(x-1, r) \neq Q(x, r)$  the  $H$ -values have to be computed anew, in general, but, depending on the  $y$ -coordinates of the defects in  $Q(x, r) \setminus Q(x-1, r)$ , possibly only a few number of them. Further opportunities, briefly discussed below, exist to reduce the computational effort of the recursions.

Moreover, the computation of  $H(x, r, y, \bar{w}_k)$  is not needed if it is equal to  $H(x, r, y-1, \bar{w}_k)$ . We consider the situation as depicted in Fig. 9.12 which shows the lower edge within the board segment  $(x-r, 0, x, W)$ . If  $v_1 - x + r < \min_{i \in I} l_i$ , the discontinuity point  $v_1$  of  $\omega$  does not effect the  $H$ -values within this segment if  $\omega_1 < \omega_2$  holds. A similar situation has to be regarded on the right edge of the board segment where again  $\min_{i \in I} l_i$  determines this segment. In Fig. 9.12 let  $\underline{x} := x - r + \min_{i \in I} l_i$  and  $\bar{x} := x - \min_{i \in I} l_i$ . Only the discontinuity points  $v_q$  between  $\underline{x}$  and  $\bar{x}$  can have some effect for the  $H$ -values.

To formulate a sufficient criterion we define a set of  $y$ -coordinates where changes of values can be possible:

$$\begin{aligned}
 Y_d(x, r, k) := & \{d_q : a_q < x, c_q > x - r, q \in Q\} \cup \\
 & \{\omega_q : x - r + \min_{i \in I_k} \ell_i \leq v_q, v_{q-1} + \min_{i \in I_k} \ell_i \leq x, q \in Q_\omega\} \cup \\
 & \{\omega_q : x - r < v_q < x - r + \min_{i \in I_k} \ell_i, \omega_q > \omega_{q+1}, q \in Q_\omega\} \cup \\
 & \{\omega_q : \omega_q > \omega_{q-1}, x - \min_{i \in I_k} \ell_i < v_{q-1} < x, q \in Q_\omega\}, \\
 Y_b(x, r, k) := & \{b_q : a_q < x, c_q > x - r, q \in Q\} \cup \\
 & \{\psi_q : x - r + \min_{i \in I_k} \ell_i \leq \mu_q, \mu_{q-1} + \min_{i \in I_k} \ell_i \leq x, q \in Q_\psi\} \cup \\
 & \{\psi_q : x - r < \mu_q < x - r + \min_{i \in I_k} \ell_i, \psi_q < \psi_{q+1}, q \in Q_\psi\} \cup \\
 & \{\psi_q : \psi_q < \psi_{q-1}, x - \min_{i \in I_k} \ell_i < \mu_{q-1} < x, q \in Q_\psi\}.
 \end{aligned}$$

When passing from strip  $(x-r, y-1-\bar{w}_k, x, y-1)$  to strip  $(x-r, y-\bar{w}_k, x, y)$ , in case of  $y-\bar{w}_k \in Y_d(x, r, k)$  new possibilities to place pieces result since at least one defective region is no longer contained. On the other hand, if  $y-1 \in Y_b(x, r, k)$ , then additional defects have to be regarded which can lead to changed  $H$ -values. Altogether, we have the following sufficient condition:

**Proposition 9.8** *If  $y-\bar{w}_k \notin Y_d(x, r, k)$  and  $y-1 \notin Y_b(x, r, k)$  hold, then  $H(x, r, y, \bar{w}_k) = H(x, r, y-1, \bar{w}_k)$  holds.*

In a similar way, the computation of  $G(x, r, y)$  can be avoided in some particular cases.

**Proposition 9.9** If  $y - \bar{w}_k \notin Y_d(x, r, k)$  and  $y - 1 \notin Y_b(x, r, k)$  hold for all  $k \in K$  with  $\bar{w}_k \leq y$ , then we have  $G(x, r, y) = G(x, r, y - 1)$ .

To transfer these ideas to the length direction, we define sets of points

$$X_a := \{a_q : q \in Q\} \cup \{\mu_q : \psi_{q+1} < \psi_q, q \in Q_\psi\} \cup \{v_q : \omega_{q+1} > \omega_q, q \in Q_\omega\},$$

$$X_c := \{c_q : q \in Q\} \cup \{\mu_q : \psi_{q+1} > \psi_q, q \in Q_\psi\} \cup \{v_q : \omega_{q+1} < \omega_q, q \in Q_\omega\}.$$

If  $x - r \in X_c$ , then there exists at least a defect which is relevant for segment  $(x - 1 - r, 0, x - 1, W)$ , but not for segment  $(x - r, 0, x, W)$ . In case of  $x - 1 \in X_a$ , then there exists at least a defective region which does not intersect segment  $(x - 1 - r, 0, x - 1, W)$ , but segment  $(x - r, 0, x, W)$ . A possible change of the  $G$ -values can also happen if the shifting of the segment by one unit leads to a changed  $H$ -value.

**Proposition 9.10** Let  $j_0 \in J$ . If  $x - r_j \notin X_c$  and  $x - r_{j_0} + r_j - 1 \notin X_a$  hold for all  $j = 0, \dots, j_0$  (where  $r_0 = 0$ ), then  $G(x, r_{j_0}, W) = G(x - 1, r_{j_0}, W)$  follows.

Moreover, if the assumptions of Proposition 9.10 are satisfied for some  $x \notin R$  and all  $r_{j_0} \leq x$  and if  $F(x - 1 - r_j) = F(x - r_j)$  holds for all  $r_j < x$ , then  $F(x) = F(x - 1)$  follows, too.

### 9.2.4 Examples and Extensions

*Example 9.4* Let a board (raw material) with length  $L = 418$  and width  $W = 41$  be given. As possible products, six (non-rotatable) piece types are defined in Table 9.2. All defective regions (defects) occurring on the board are listed in Table 9.3 as set of

**Table 9.2** Input data of the piece types of Example 9.4

| $i$      | 1  | 2  | 3  | 4  | 5  | 6  |
|----------|----|----|----|----|----|----|
| $\ell_i$ | 29 | 33 | 45 | 46 | 50 | 83 |
| $w_i$    | 2  | 3  | 3  | 3  | 6  | 6  |
| $e_i$    | 6  | 11 | 15 | 16 | 39 | 75 |

**Table 9.3** Input data of defects of Example 9.4

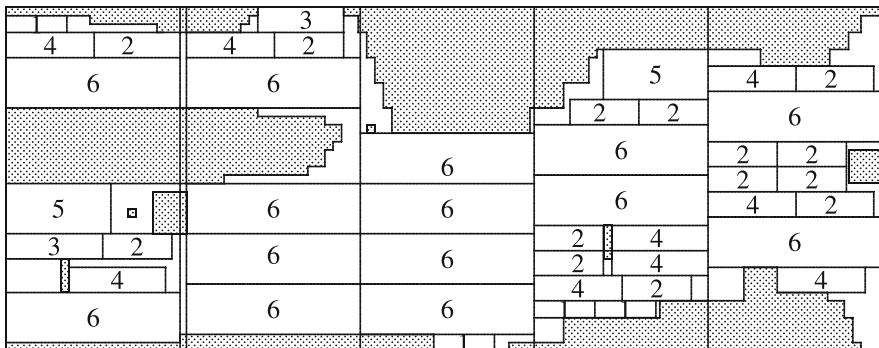
| $a_q$ | $b_q$ | $c_q$ | $d_q$ |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 83    | 1     | 0     | 40    | 120   | 41    | 168   | 38    | 408   | 40    | 260   | 1     | 404   | 4     |
| 0     | 20    | 104   | 21    | 26    | 7     | 30    | 11    | 172   | 36    | 402   | 38    | 285   | 11    | 289   | 15    |
| 0     | 21    | 144   | 22    | 40    | 39    | 116   | 40    | 172   | 26    | 176   | 27    | 312   | 4     | 400   | 6     |
| 0     | 22    | 152   | 24    | 58    | 16    | 62    | 17    | 172   | 35    | 282   | 36    | 352   | 6     | 368   | 10    |
| 0     | 24    | 156   | 25    | 70    | 14    | 86    | 19    | 176   | 32    | 278   | 35    | 360   | 34    | 393   | 36    |
| 0     | 25    | 160   | 27    | 72    | 38    | 112   | 39    | 180   | 29    | 266   | 32    | 368   | 6     | 392   | 7     |
| 0     | 27    | 152   | 28    | 84    | 0     | 204   | 2     | 184   | 26    | 250   | 29    | 402   | 20    | 418   | 24    |
| 0     | 28    | 120   | 29    | 161   | 40    | 418   | 41    | 240   | 0     | 408   | 1     |       |       |       |       |

**Table 9.4** Description of defects of Example 9.4 using  $\psi$  and  $\omega$ 

| $a_q$ | $b_q$ | $c_q$ | $d_q$ |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 20    | 104   | 21    | 0     | 25    | 160   | 27    | 26    | 7     | 30    | 11    | 172   | 26    | 176   | 27    |
| 0     | 21    | 144   | 22    | 0     | 27    | 152   | 28    | 58    | 16    | 62    | 17    | 285   | 11    | 289   | 15    |
| 0     | 22    | 152   | 24    | 0     | 28    | 120   | 29    | 70    | 14    | 86    | 19    | 402   | 20    | 418   | 24    |
| 0     | 24    | 156   | 25    |       |       |       |       |       |       |       |       |       |       |       |       |

|            |     |     |     |     |     |     |     |     |     |     |     |     |     |  |  |
|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--|--|
| $\mu_q$    | 0   | 40  | 72  | 112 | 116 | 120 | 161 | 168 | 172 | 176 | 180 | 184 | 250 |  |  |
| $\psi_q$   | -   | 40  | 39  | 38  | 39  | 40  | 41  | 40  | 38  | 35  | 32  | 29  | 26  |  |  |
| $\mu_q$    | 266 | 278 | 282 | 360 | 393 | 402 | 408 | 418 |     |     |     |     |     |  |  |
| $\psi_q$   | 29  | 32  | 35  | 36  | 34  | 36  | 38  | 40  |     |     |     |     |     |  |  |
| $v_q$      | 0   | 83  | 204 | 240 | 266 | 312 | 352 | 368 | 392 | 400 | 404 | 408 | 418 |  |  |
| $\omega_q$ | -   | 1   | 2   | 0   | 1   | 4   | 6   | 10  | 7   | 6   | 4   | 1   | 0   |  |  |

**Fig. 9.13** Cutting pattern of Example 9.4

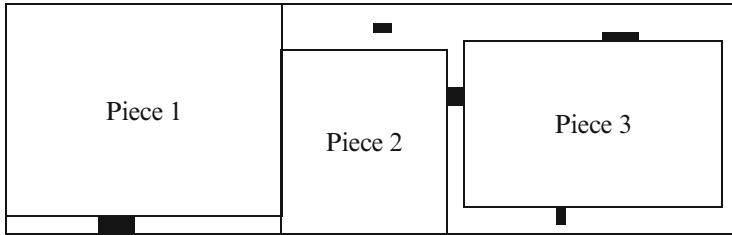
rectangles. For comparison, all input data of the defects are also given in Table 9.4 using the functions  $\psi$  and  $\omega$ . A real kerf will not be considered within this example.

As result of the optimization based on the recursions of Sect. 9.2.2, we obtain the pattern depicted in Fig. 9.13. ■

Note that more general, and therefore, technologically more complex patterns as, for instance, shown in Fig. 9.14, can be considered in a similar way. It is obvious that applying more general cutting technologies can lead to a higher yield (material utilization), but require, in general, higher computational effort and can increase the cutting costs. Their usage has to be decided from case to case.

The input data of the following cutting problem are taken from [2].

*Example 9.5* Rectangular piece types as listed in Table 9.5 can be obtained from a board of length  $L = 239$  and width  $W = 120$  using guillotine cuts. We aim to maximize the total yield. In this example rotation of the pieces by  $90^\circ$  is allowed. On the left border of the pallet, a chop cut of at least 5 units has to be performed,



**Fig. 9.14** A more general pattern of a strip

**Table 9.5** Input data of piece types of Example 9.5

| $i$      | 1   | 2   | 3   | 4   | 5  | 6  | 7  |
|----------|-----|-----|-----|-----|----|----|----|
| $\ell_i$ | 120 | 112 | 90  | 90  | 60 | 38 | 20 |
| $w_i$    | 36  | 74  | 40  | 36  | 42 | 18 | 18 |
| $e_i$    | 230 | 770 | 166 | 138 | 89 | 12 | 5  |

**Table 9.6** Input data of defects of Example 9.5

| $a_q$ | $b_q$ | $c_q$ | $d_q$ |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 10    | 15    | 0     | 68    | 4     | 120   | 170   | 15    | 180   | 19    | 198   | 100   | 208   | 104   |
| 0     | 15    | 12    | 25    | 52    | 0     | 72    | 15    | 193   | 15    | 203   | 19    | 212   | 15    | 222   | 19    |
| 0     | 40    | 26    | 68    | 124   | 15    | 134   | 19    |       |       |       |       |       |       |       |       |

i.e., the position  $x$  of the first (vertical) guillotine cut is limited by  $x \geq 5$ . The value (yield)  $e_i$ ,  $i \in I$  is defined by

$$e_i := \lceil \beta(1 + \alpha \ell_i w_i) \ell_i w_i \rceil \quad \text{with} \quad \beta = 0.01, \alpha = 0.001.$$

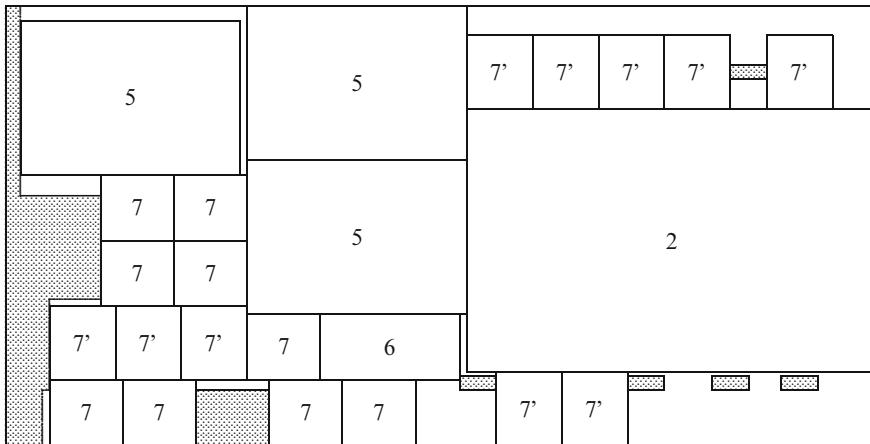
This definition realizes a stronger valuation of larger pieces in comparison to smaller ones. The unusable (defective) regions are given in Table 9.6.

The pattern obtained by the optimization process is drawn in Fig. 9.15. In dependence of the parameters  $\alpha$  and  $\beta$  which control the piece type values, various different patterns can be obtained.

Note that the position of the vertical cuts of the first stage can easily be limited within the recursion so that they are not too near to the margins of the board. ■

### 9.3 Exercises

**Exercise 9.1** Let a (one-dimensional) board of length  $L = 200$  cm be given which possesses sound knots with diameter at most 1 cm in the intervals  $[60, 61]$ ,  $[80, 81]$ , and  $[130, 131]$ , and a black knot with diameter 2 cm in  $[120, 122]$ . Pieces of type 1 should have a length at least 30 cm and those of type 2 at least 50 cm. Moreover,



**Fig. 9.15** Cutting pattern of Example 9.5

pieces of type 2 must not contain any knots, however, pieces of type 1 may contain at most one knot with diameter at most 1 cm within any interval of length 30 cm. The task is to identify all allocation intervals for both piece types.

**Exercise 9.2** Let a board with length  $L = 100$  be given which contains knots within the intervals  $[60, 62]$  and  $[67, 70]$  in  $x$ -direction. For piece type  $i = 1$  no knots are allowed, the minimum length is 30 and the maximum length 50. Knots are allowed for piece type  $i = 2$  where the minimum length is 20. Define appropriate allocation intervals and compute an optimal pattern using the yield coefficients

- (a)  $c_1 = 0.8, c_2 = 0.3,$
- (b)  $c_1 = 0.8, c_2 = 0.5.$

**Exercise 9.3** Consider the same situation as in Exercise 9.2 and let  $c_1 = 0.5, c_2 = 0.4$ . Compute the optimal value function and all optimal patterns for the following cases:

- (a)  $l_1 = L_1 = 30, l_2 = 20, L_2 = 100,$
- (b)  $l_1 = L_1 = 40, l_2 = 20, L_2 = 100,$
- (c)  $l_1 = L_1 = 30, l_2 = L_2 = 20.$

**Exercise 9.4** In addition to Exercise 9.3 a kerf of width 2 has to be regarded.

**Exercise 9.5** Compute the bounding functions  $ub_1(x)$  and  $ub_2(x)$  for Exercise 9.3.

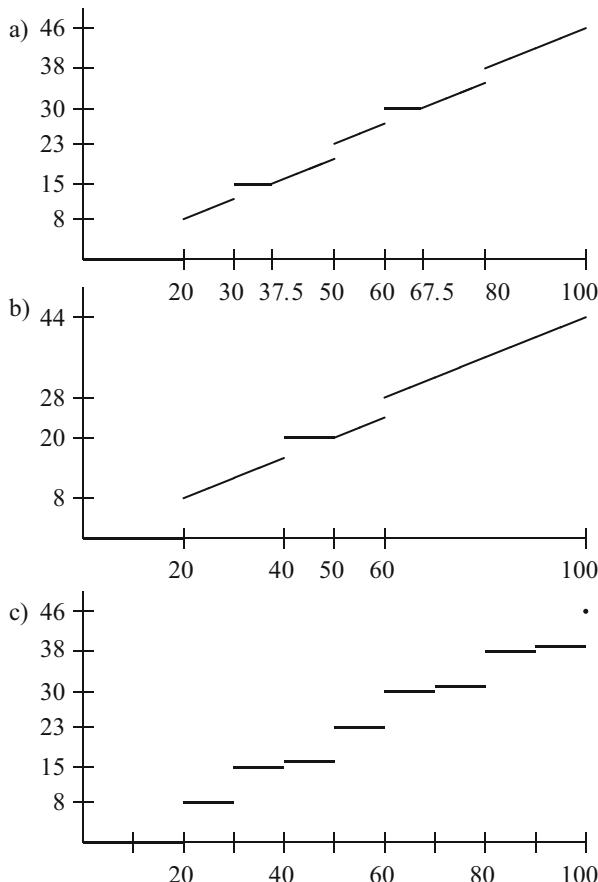
## 9.4 Solutions

**To Exercise 9.1** We obtain the following allocation intervals:

$$A_{11} = [0, 80], A_{12} = [61, 120], A_{13} = [122, 200], A_{21} = [0, 50], A_{22} = [131, 200].$$

**To Exercise 9.2** Three allocation intervals result:  $A_{1,1} = [0, 60]$ ,  $A_{1,2} = [70, 100]$ , and  $A_{2,1} = [0, 100]$ . In case (a), the pattern consisting of three pieces of types 1 with length 30 (yield 72) is better than the pattern consisting of two pieces of type 1 having length 50 and 30 and one piece of type 2 with length 20 (yield 70). However, in case (b), we find a reverse situation (72 and 74, respectively).

**To Exercise 9.3** The optimal value functions are drawn in Fig. 9.16.

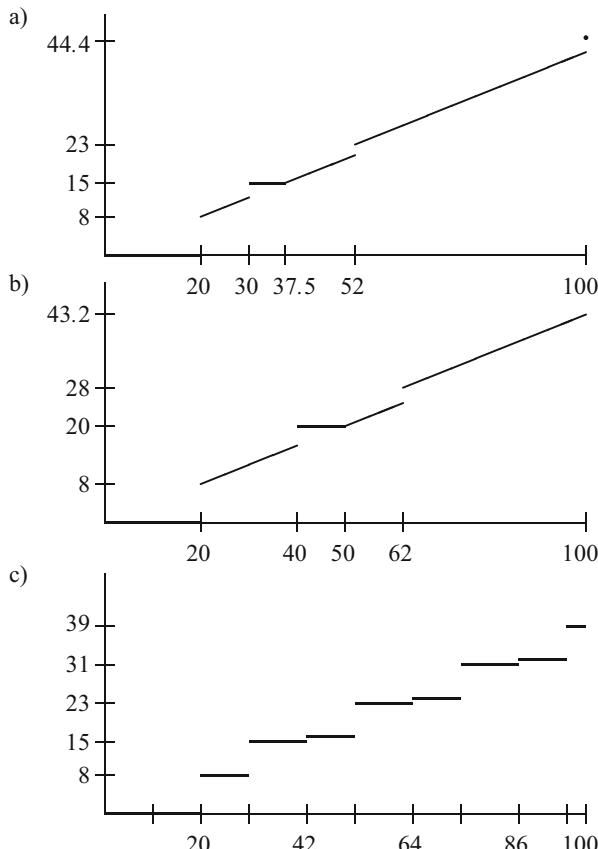


**Fig. 9.16** Optimal value functions of Exercise 9.3 (a)–(c)

In case (a), we obtain an optimal pattern which contains  $T_1$  in  $[0, 30]$  and  $[30, 60]$  as well as  $T_2$  in  $[60, 100]$  with total yield 46. Moreover, in case (b), we get  $T_1$  from  $[0, 40]$  and  $T_2$  from  $[40, 100]$ . The maximal yield is 44. Finally, in case (c), we obtain, in comparison to (a), instead of one piece of type 2 with length 40 two pieces with length 20.

**To Exercise 9.4** The optimal value functions are shown in Fig. 9.17.

In case (a), due to the kerf of width 2, we obtain an optimal pattern with total yield 44.4:  $T_1$  in  $[0, 30]$  and  $[70, 100]$ , and  $T_2$  in  $[32, 68]$ . Furthermore,  $T_1$  is cut from  $[0, 40]$  and  $T_2$  from  $[42, 100]$ . In case (b). The total yield is 43.2. After all, in case (c), one piece of type 1 from  $[0, 30]$  and three pieces-of type 2 from  $[32, 52]$ ,  $[54, 74]$ , and  $[76, 96]$  are obtained. Obviously, shifting some pieces leads to other optimal patterns.



**Fig. 9.17** Optimal value functions of Exercise 9.4 (a)–(c)

**To Exercise 9.5** We have  $c_1 = 0.5 > c_2 = 0.4$ . Therefore, in case (a), we have

$$i_1(x) = \begin{cases} 1, & x \in [0, 60] \cup [70, 100], \\ 2, & \text{otherwise.} \end{cases}$$

The bounding function  $ub_1(x)$  is depicted in Fig. 9.18.

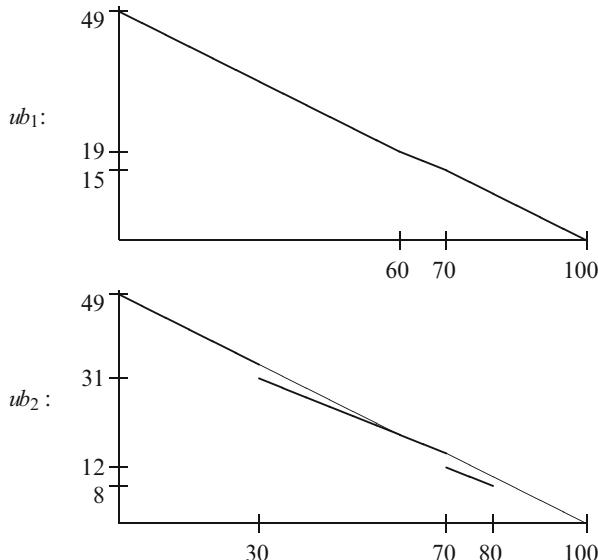
Using an artificial piece type 3 with  $c_3 = 0$ , we obtain for  $x \in [0, 100]$ :

$$\begin{aligned} 80 < x \leq 100 : i_2(x, t) &= 3 \quad \text{for } t \in [x, 100], \\ 70 < x \leq 80 : i_2(x, t) &= 2 \quad \text{for } t \in [x, 100]. \\ 30 < x \leq 70 : i_2(x, t) &= \begin{cases} 1 & \text{for } t \in [70, 100], \\ 2 & \text{for } t \in [x, 70], \end{cases} \\ 0 \leq x \leq 30 : i_2(x, t) &= \begin{cases} 1 & \text{for } t \in [70, 100], \\ 2 & \text{for } t \in (60, 70), \\ 1 & \text{for } t \in [x, 60]. \end{cases} \end{aligned}$$

The resulting bounding function  $ub_2$  is also shown in Fig. 9.18. For comparison,  $ub_1$  is drawn in the lower figure as thin line as well.

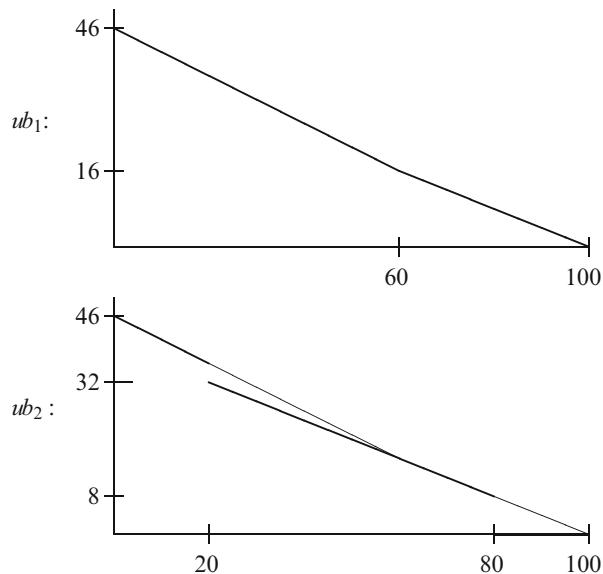
In case (b) with  $\ell_1 = 40$ , we obtain the following changed values where we have to regard that only the allocation interval  $A_{11}$  is used for piece type 1:

$$i_1(x) = \begin{cases} 1 & \text{for } x \in [0, 60], \\ 2 & \text{otherwise.} \end{cases}$$



**Fig. 9.18** Bounding functions  $ub_1$  and  $ub_2$  of Exercise 9.5 (a)

**Fig. 9.19** Bounding functions  $ub_1$  and  $ub_2$  of Exercise 9.5 (b)



Using  $c_3 = 0$ , we have for  $x \in [0, 100]$ :

$$\begin{aligned} 80 < x \leq 100 : i_2(x, t) &= 3 \text{ for } t \in [x, 100], \\ 20 < x \leq 80 : i_2(x, t) &= 2 \text{ for } t \in [x, 100], \\ 0 \leq x \leq 20 : i_2(x, t) &= \begin{cases} 2 & \text{for } t \in (60, 100], \\ 1 & \text{for } t \in [x, 60]. \end{cases} \end{aligned}$$

The resulting bounding functions  $ub_1$  and  $ub_2$  are shown in Fig. 9.19. For comparison,  $ub_1$  is drawn in the lower figure as thin line as well.

For (c), we obtain the same bounding functions as in case (a).

## References

1. A. Fischer, G. Scheithauer, Cutting and packing problems with placement constraints, in *Optimized Packings with Applications*, vol. 105, ed. by G. Fasano, J.D. Pinter (Springer, Cham, 2015), pp. 119–156
2. S.G. Hahn, On the optimal cutting of defective sheets. Oper. Res. **16**(6), 1100–1114 (1968)
3. M. Rönnqvist, E. Astrand, Integrated defect detection and optimization for cross cutting of wooden boards. Eur. J. Oper. Res. **108**, 490–508 (1998)
4. G. Scheithauer, The solution of packing problems with pieces of variable length and additional allocation constraints. Optimization **34**, 81–96 (1995)
5. G. Scheithauer, J. Terno, Optimale Positionierung beim Vollholzzuschnitt durch automatisierte Fehlererkennung. Wiss. Z. TU Dresden **48**(2), 78–81 (1999)

6. U. Twisselmann, Cutting rectangles avoiding rectangular defects. *Appl. Math. Lett.* **12**, 135–138 (1999)
7. F.J. Vasko, D.D. Newhart, K.L. Stott Jr., A hierarchical approach for one-dimensional cutting stock problems in the steel industry that maximizes yield and minimizes overgrading. *Eur. J. Oper. Res.* **114**, 72–82 (1999)

# Chapter 10

## Pallet Loading

An important practical optimization problem frequently arising when goods have to be transported from the producer to a distribution center is topic of this chapter. As variants, Bischoff and Ratcliff [5] distinguished between the *manufacturer's pallet loading problem* (M'sPLP) and the *distributor's pallet loading problem* (D'sPLP). In the M'sPLP, identical box-shaped pieces (items, boxes) have to be loaded on a given pallet such that their number is maximized and natural restrictions are met. In the D'sPLP, a number of different shaped box-types with given supply has to be packed onto (identical) pallets while minimizing the number of pallets needed.

In the following, we first consider the two-dimensional variant of the M'sPLP and describe a heuristic which computes optimal patterns if not more than 50 items can be packed and some ratio of the piece dimensions is fulfilled. Moreover, we investigate the special case of the *guillotine pallet loading problem* (GPLP) and show that an optimal pattern can be computed in polynomial time. Furthermore, we describe an efficient heuristic for the D'sPLP.

### 10.1 The Standard Pallet Loading Problem

In case of the M'sPLP, frequently, a layer-wise packing of the boxes is applied in which all items placed in the layer have the same height. Doing so, we can reduce the three-dimensional problem to a two-dimensional one. The latter is usually called the *standard pallet loading problem* or, simply, the *pallet loading problem* (PLP).

### 10.1.1 Problem Statement

The (orthogonal) three-dimensional pallet loading problem is the following: identical boxes of size  $\ell \times w \times h$  have to be placed on a (rectangular) pallet of size  $L \times W$  such that the number of packed pieces is maximal and all restrictions are met, i.e., the boxes must not intersect, their projections onto the  $L$ - $W$ -plane lie within  $L \times W$ , and the predefined maximum packing height is not exceeded. Sometimes additional restrictions how to place the boxes have to be regarded. For instance, frequently the so-called *This site up!* condition has to be met which leads to a layer-wise packing. In this case, the three-dimensional M'sPLP can be reduced to a two-dimensional rectangle packing problem.

Due to its importance, we will consider the two-dimensional PLP in more detail: which maximal number of rectangles of size  $\ell \times w$  can be placed on the rectangular pallet  $L \times W$  without overlap? Naturally, rotation by  $90^\circ$  is allowed,

The PLP also belongs to the class of NP-hard problems, i.e., there is not known any polynomial-time solution algorithm for it. Therefore, b&b algorithms have to be applied, in general, to obtain a proved optimal pattern. For instance, an appropriate modification of a b&b algorithm designed for the 2OPP (Chap. 5) can be used.

Within the next subsection, the equivalence of PLP instances is investigated. After that, various upper bounds for the PLP are presented.

As usual, we describe an instance of the (two-dimensional) PLP by a quadruple  $(L, W, \ell, w)$ . Without loss of generality, we assume  $L, W, \ell, w \in \mathbb{N}$ ,  $L \geq W \geq \max\{\ell, w\}$ , and  $\gcd(\ell, w) = 1$  where  $\gcd(\ell, w)$  denotes the greatest common divisor of  $\ell$  and  $w$ . Observe, in case of  $\gcd(\ell, w) = q > 1$ , we can reduce the instance  $(L, W, \ell, w)$  to the equivalent instance  $(\lfloor L/q \rfloor, \lfloor W/q \rfloor, \ell/q, w/q)$ . Furthermore, we assume  $\ell \geq w$  in this section.

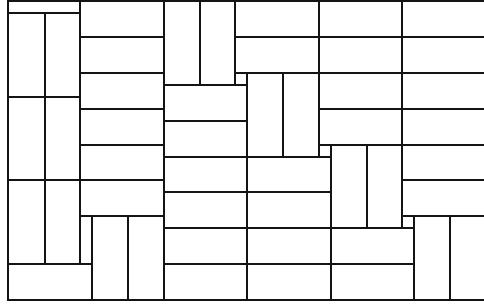
### 10.1.2 Equivalence of PLP Instances

Let us consider two PLP instances  $(40, 25, 7, 3)$  and  $(52, 33, 9, 4)$ . Although their input data are different, both instances possess the same optimal pattern (with respect to the number of packed pieces and their configuration) as shown in Fig. 10.1. However, the two instances are not equivalent in the following sense (Dowsland [6]):

**Definition 10.1** Two instances  $(L, W, \ell, w)$  and  $(L', W', \ell', w')$  of the PLP are *equivalent*, if for every feasible pattern of  $(L, W, \ell, w)$  there exists a feasible pattern of  $(L', W', \ell', w')$ , and reversely.

Using binary variables  $u_{ij}$  and  $v_{ij}$  of the Padberg-type model for the 2OPP (Sect. 5.2.2), two PLP instances are equivalent if and only if whenever  $u_{ij}$  and  $v_{ij}$ ,  $i \neq j \in I$ , represent a feasible pattern for the first instance, then they are also feasible for the second one, where  $I$  denotes an index set of the packed pieces.

**Fig. 10.1** Optimal pattern of the PLP instances  
(40, 25, 7, 3) and  
(52, 33, 9, 4)



To efficiently identify equivalent instances, we define the set of *efficient combinations* with respect to  $L$

$$E(L, \ell, w) := \{(p, q) : L - w < \ell p + wq \leq L, p, q \in \mathbb{Z}_+\}$$

and  $W$

$$E(W, \ell, w) := \{(p, q) : W - w < \ell p + wq \leq W, p, q \in \mathbb{Z}_+\}.$$

**Theorem 10.1 (Dowsland [6])** *Two PLP instances  $(L, W, \ell, w)$  and  $(L', W', \ell', w')$  are equivalent if and only if*

$$E(L, \ell, w) = E(L', \ell', w'), \quad E(W, \ell, w) = E(W', \ell', w'),$$

$$\lfloor L/\ell \rfloor = \lfloor L'/\ell' \rfloor, \quad \text{and} \quad \lfloor W/\ell \rfloor = \lfloor W'/\ell' \rfloor.$$

*Proof* Due to the assumptions, the polytope  $P(L) := \text{conv}(\{(x, y) \in \mathbb{Z}_+^2 : \ell x + wy \leq L\})$  coincides with the polytope  $P'(L') := \text{conv}(\{(x, y) \in \mathbb{Z}_+^2 : \ell'x + w'y \leq L'\})$ , since all extreme points coincide. Moreover, we have  $P(W) = P'(W')$ , too. Therefore, all feasible patterns of  $(L, W, \ell, w)$  are also feasible for  $(L', W', \ell', w')$ , and reversely. ■

Considering again the two PLP instances (40, 25, 7, 3) and (52, 33, 9, 4) from the beginning, their efficient combinations are equal except (1, 11) and (1, 10).

Based on this equivalence relation, the set of PLP instances can be gathered quantitatively. To this end, we consider PLP instances of type I, introduced in [16], which are characterized by the following conditions:

$$1 \leq \frac{L}{W} \leq 2, \quad 1 \leq \frac{\ell}{w} \leq 4, \quad 6 \leq \frac{LW}{\ell w} < 51,$$

and which are, in particular, important for practical purposes. In [16], this set is named *COVER I* and therein its cardinality is presented, too:

$$\text{card}(\text{COVER I}) = 8.274.$$

PLP instances of type II are characterized by

$$1 \leq \frac{L}{W} \leq 2, \quad 1 \leq \frac{\ell}{w} \leq 4, \quad 51 \leq \frac{LW}{\ell w} < 101.$$

They differ from that of type I only by the maximum number of pieces which can be placed. A corresponding set of representatives of the equivalence classes, called *COVER II*, has the cardinality

$$\text{card}(\text{COVER II}) = 41.831.$$

As a representative of an equivalence class one can use an instance with minimal (integer) data, a so-called *minimum size instance* [13]. An instance  $(L, W, \ell, w)$  is called minimum size instance if we have  $L \leq L'$ ,  $W \leq W'$ ,  $\ell \leq \ell'$ , and  $w \leq w'$  for all instances  $(L', W', \ell', w')$  which are equivalent to  $(L, W, \ell, w)$ . The existence and uniqueness of such a representative of an equivalence class is shown in [13].

Since there is not known any polynomial time algorithm for the PLP, b&b algorithms have to be used to get optimal solutions, in general. In principle, the b&b algorithm proposed for the ZOPP (Sect. 5.6.2) can be applied. Other b&b approaches designed for the PLP are presented in [3, 13], and [16]. Moreover, an application of a Tabu Search algorithm can be found in [1].

### 10.1.3 Upper Bounds

Various possibilities are known to get an upper bound of the optimal value for a given PLP instance  $(L, W, \ell, w)$ . A first, rather trivial bound results from the ratio of pallet and piece area:

$$ub_a(L, W) := \lfloor (LW)/(\ell w) \rfloor.$$

As usual, this bound is named *area* or *material bound*. Obviously, an improved upper bound  $ub_r$  can be obtained if instead of  $L$  and  $W$  the actual maximal usable length  $L'$  and width  $W'$  are used according to the concept of potential allocation points (Sect. 2.7). We have

$$\begin{aligned} L' &= \max\{p\ell + qw : (p, q) \in E(L, \ell, w)\}, \\ W' &= \max\{p\ell + qw : (p, q) \in E(W, \ell, w)\}, \quad \text{and} \\ ub_r(L, W) &:= ub_a(L', W'). \end{aligned}$$

Without loss of generality, we assume  $L = L'$  and  $W = W'$  in the following.

Barnes proposed another upper bound  $ub_B(L, W)$  in [2] which is based on the observation that each feasible arrangement of (rotatable) rectangles  $\ell \times w$  within a

pallet  $L \times W$  is a feasible arrangement of items  $\ell \times 1$  or of items  $1 \times w$  as well. This corresponds to the philosophy of the bar relaxation for two-dimensional rectangle cutting or packing (Chap. 5).

For short, let

$$r_\ell := L \bmod \ell, \quad s_\ell := W \bmod \ell, \quad r_w := L \bmod w, \quad s_w := W \bmod w.$$

Due to [2] the minimum waste  $A_\ell$ , necessarily arising when  $\ell \times 1$ -items are placed, is equal to

$$A_\ell = \min\{r_\ell s_\ell, (\ell - r_\ell)(\ell - s_\ell)\}.$$

Similarly, when  $1 \times w$ -items are placed, the minimum waste  $A_w$  is equal to

$$A_w = \min\{r_w s_w, (w - r_w)(w - s_w)\}.$$

With these two values the *Barnes bound* is defined as follows:

$$ub_B(L, W) := \lfloor (LW - \max\{A_\ell, A_w\}) / (\ell w) \rfloor.$$

Another strong bound is obtained by minimizing the area bound  $ub_a$  with respect to all equivalent instances. The resulting bound is denoted by  $ub_e(L, W)$ . In [6, 9] an efficient method to compute

$$ub_e(L, W) := \min\{\lfloor (L'W') / (\ell'w') \rfloor : (L', W', \ell', w') \text{ is equivalent to } (L, W, \ell, w)\}$$

is described. The computation of  $ub_e(L, W)$  requires to solve a (finite) sequence of simple linear programming problems.

A further upper bound  $ub_I(L, W)$  was proposed by Isermann [11]. This bound exploits a combination of one-dimensional relaxations (in vertical and horizontal direction). The bar relaxation bound for the 2OPP considered in Sect. 5.3.2 yields the *Isermann bound* if only a single rotatable rectangular piece type is given. However, in history, the Barnes and the Isermann bound constitute the source of the development of the bar relaxation.

Let  $x_{ij}$  denote the frequency how often efficient combination  $(i, j) \in E(L, \ell, w)$  is used as horizontal strip-pattern of width 1, and let  $y_{pq}$  indicate the number how often  $(p, q) \in E(W, \ell, w)$  is used as vertical pattern. Therefore, the computation of  $ub_I(L, W)$  requires to solve the following integer linear programming problem:

### Isermann bound for the PLP

$$ub_I(L, W) := \max \left[ \sum_{(i,j) \in E(L, \ell, w)} \frac{i x_{ij}}{w} + \sum_{(p,q) \in E(W, \ell, w)} \frac{p y_{pq}}{w} \right] \quad \text{s. t.} \quad (10.1a)$$

$$\sum_{(i,j) \in E(L, \ell, w)} x_{ij} \leq W, \quad \sum_{(p,q) \in E(W, \ell, w)} y_{pq} \leq L, \quad (10.1b)$$

$$\sum_{(i,j) \in E(L, \ell, w)} \ell i x_{ij} = \sum_{(p,q) \in E(W, \ell, w)} w q y_{pq}, \quad (10.1c)$$

$$\sum_{(i,j) \in E(L, \ell, w)} w j x_{ij} = \sum_{(p,q) \in E(W, \ell, w)} \ell p y_{pq}, \quad (10.1d)$$

$$x_{ij} \in \mathbb{Z}_+, \quad (i,j) \in E(L, \ell, w), \quad y_{pq} \in \mathbb{Z}_+, \quad (p,q) \in E(W, \ell, w). \quad (10.1e)$$

Since, in general, the integer demand can lead to a high computational effort, frequently the optimal value  $ub_{LP}(L, W)$  of the LP relaxation of (10.1) is used. A detailed analysis of upper bounds of the PLP can be found in [12].

## 10.2 The G4 Heuristic

The following heuristic results as generalization of various others applying a recursive approach.

### 10.2.1 Notations, Block-Heuristics

Within this subsection we assume  $\ell > w$ . A horizontally oriented piece, i.e., the rectangle  $\ell \times w$ , is called *H-piece*, and a vertically oriented, i.e.,  $w \times \ell$ , is named *V-piece*. The position (allocation point) of a piece is again defined to be the position of its lower left corner. As usual, the pallet is represented by the point set

$$P(L, W) := \{(x, y) \in \mathbb{R}^2 : 0 \leq x < L, 0 \leq y < W\}.$$

Then, the occupied area of an *H*-piece with allocation point  $(x_i, y_i)$  is described by

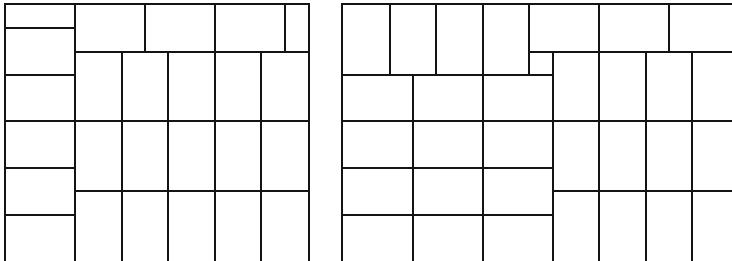
$$(x_i, y_i, H) := \{(x, y) \in \mathbb{R}^2 : x_i \leq x < x_i + \ell, y_i \leq y < y_i + w\}$$

and that of a *V*-piece by

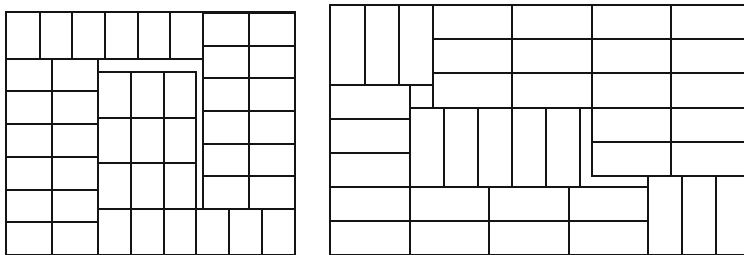
$$(x_i, y_i, V) := \{(x, y) \in \mathbb{R}^2 : x_i \leq x < x_i + w, y_i \leq y < y_i + \ell\}.$$

Moreover, a *pattern A* of  $n$  pieces is given by

$$A = A(I) = \{(x_i, y_i, o_i) : i \in I\} \quad \text{with } I = \{1, \dots, n\},$$



**Fig. 10.2** Optimal 3-block pattern for  $(13, 11, 3, 2)$  and optimal 4-block pattern for  $(17, 11, 3, 2)$



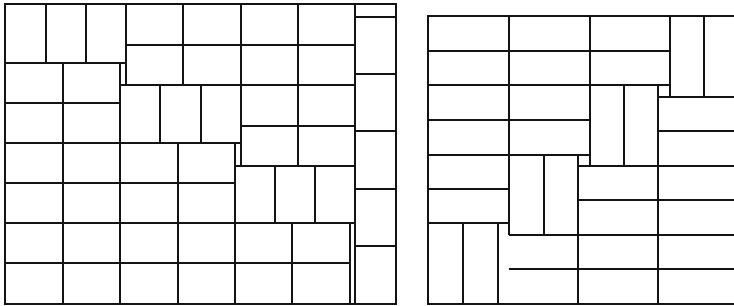
**Fig. 10.3** Optimal 5-block pattern for  $(44, 37, 7, 5)$  and optimal 7-block pattern for  $(37, 22, 7, 3)$

where  $(x_i, y_i)$  denotes the *allocation point* of the  $i$ th piece, and  $o_i \in \{H, V\}$  indicates the *orientation* of it. A pattern  $A(I)$  with  $I = \{1, \dots, n\}$  is called *homogeneous* if all placed pieces have the same orientation (cf. Sect. 6.5).

**Definition 10.2** A pattern  $A(I)$  of pallet  $P(L, W)$  with  $I = \{1, \dots, n\}$  is called *k-block pattern* if a partition of  $I$  in  $k$  subsets  $I_j$  and corresponding disjunctive rectangles  $R_j \subseteq P(L, W)$ ,  $j = 1, \dots, k$ , exist such that the restriction of  $A(I_j)$  to  $R_j$  for all  $j$  is a homogeneous pattern.

As easily can be seen,  $k$ -block patterns with  $k \leq 3$  are guillotine patterns, cf. Sect. 6.5. However, for  $k > 3$ , a  $k$ -block pattern is, in general, not of guillotine type. In Fig. 10.2 an optimal 3-block pattern for the instance  $(13, 11, 3, 2)$  and an optimal 4-block pattern for the instance  $(17, 11, 3, 2)$  are depicted.

The computation of optimal  $k$ -block patterns with  $k \in \{5, 7, 9\}$  or other block patterns requires considerable effort, in general. Therefore, we abstain from describing respective algorithms. Instead of that, we describe another approach in the next subsections which produces optimal patterns in the most cases. In Fig. 10.3 an optimal 5-block pattern for the instance  $(44, 37, 7, 5)$  and an optimal 7-block pattern for the instance  $(37, 22, 7, 3)$  are shown. An optimal 9-block pattern for the instance  $(68, 52, 10, 7)$  and an optimal *diagonal-block pattern* for the instance  $(27, 25, 7, 3)$  are drawn in Fig. 10.4.



**Fig. 10.4** Optimal 9-block pattern for  $(68, 52, 10, 7)$  and optimal diagonal-block pattern for  $(27, 25, 7, 3)$

### 10.2.2 The G4-Structure

Let  $A = A(I)$  represent a (packing) pattern for pallet  $P(L, W)$ . Subset  $\widetilde{A} = A(\widetilde{I}) = \{(x_i, y_i, o_i) : i \in \widetilde{I}\}$  of  $A$  is called *block pattern*, if a subset  $\widetilde{I}$  of  $I = \{1, \dots, n\}$  and a rectangle

$$R(\underline{x}, \underline{y}, \bar{x}, \bar{y}) := \{(x, y) \in \mathbb{R}^2 : \underline{x} \leq x < \bar{x}, \underline{y} \leq y < \bar{y}\}$$

exist such that

$$\bigcup_{i \in \widetilde{I}} (x_i, y_i, o_i) \subseteq R(\underline{x}, \underline{y}, \bar{x}, \bar{y}) \quad \text{and} \quad \bigcup_{i \in I \setminus \widetilde{I}} (x_i, y_i, o_i) \cap \text{int}(R(\underline{x}, \underline{y}, \bar{x}, \bar{y})) = \emptyset$$

hold.

**Definition 10.3** Pattern  $A(I)$  of pallet  $P(L, W)$  with  $I = \{1, \dots, n\}$  has *guillotine structure (G-structure)*, if  $n \leq 3$  or if a partition  $I_1$  and  $I_2 = I \setminus I_1$  of  $I$  exists such that  $A(I_1)$  and  $A(I_2)$  have guillotine structure.

**Definition 10.4** Pattern  $A(I)$  of pallet  $P(L, W)$  with  $I = \{1, \dots, n\}$  has per definition *1-block structure* if it is homogeneous, and  $A(I)$  has *k-block structure* with  $k \in \mathbb{N}$ ,  $k \geq 2$ , if a partition of  $I$  in  $q$  subsets  $I_j, j = 1, \dots, q$ , and corresponding disjunctive rectangles  $R_j \subseteq P(L, W)$  exist such that for each  $j$  the restriction of  $A(I)$  to  $R_j$  yields a pattern  $A(I_j)$  with  $p_j$ -block structure and  $\max\{q, p_1, \dots, p_q\} = k$  holds.

Observe, there is an essential difference between the terms *k-block structure* and *k-block pattern*. A *k-block pattern* has exactly  $k$  homogeneous blocks whereas the number of blocks in a pattern with *k-block structure* can exceed  $k$  due to the recursive definition.

**Proposition 10.2** A pattern  $A(I)$  has *k-block structure* with  $k \leq 3$  if and only if it has guillotine structure.

For the proof see Exercise 10.2.

**Definition 10.5** A pattern  $A(I)$  of pallet  $P(L, W)$  has *G4-structure* if  $A(I)$  has  $k$ -block structure with  $k \leq 4$ .

The notation *G4-structure* can be interpreted as *guillotine or 4-block structure* or as *generalized 4-block pattern*.

### 10.2.3 The Basic Recursion

The G4 heuristic consists of a recursion which solves the PLP restricted to patterns with G4-structure, i.e., it computes a pattern with maximal number of placed items among all patterns with G4-structure.

Due to the definition of the G4-structure, a G4-pattern consists of either two disjunctive G4-patterns (if a guillotine cut exists) or, otherwise, of four disjunctive G4-patterns. Hence, to obtain a maximal G4-pattern, maximal G4-patterns for smaller pallets (rectangles) are needed.

Let  $n(L', W')$  denote the (total) number of placed  $V$ - and  $H$ -items in a maximal G4-pattern for a pallet with length  $L'$  and width  $W'$  where  $L', W' \in \mathbb{Z}_+, L' \leq L$ , and  $W' \leq W$ . Obviously, we have

$$n(L', W') = 0 \quad \text{if } \min\{L', W'\} < w \text{ or } \max\{L', W'\} < \ell,$$

and

$$n(\ell, w) = n(w, \ell) = 1,$$

giving initial values for the following recursion. We denote the resulting numbers of placed items, when a vertical or horizontal guillotine cut is present, by

$$n_V(L', W', a) := n(a, W') + n(L' - a, W'), \quad a \in \{1, \dots, L' - 1\}, \quad \text{and}$$

$$n_H(L', W', b) := n(L', b) + n(L', W' - b), \quad b \in \{1, \dots, W' - 1\},$$

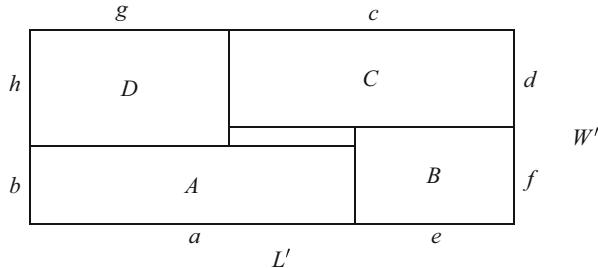
respectively. Using the notations defined in Fig. 10.5 with

$$a, b, c, d \in \mathbb{N}, \quad e := L' - a, \quad f := W' - d, \quad g := L' - c, \quad h := W' - b,$$

we obtain

$$\begin{aligned} n(L', W') = \max & \left\{ \max_{1 \leq a \leq L'/2} n_V(L', W', a), \max_{1 \leq b \leq W'/2} n_H(L', W', b), \right. \\ & \left. \max_{1 \leq a < L'} \max_{1 \leq b < W'} \max_{L' - a < c < L'} \max_{1 \leq d < W' - b} \{n(a, b) + n(e, f) + n(c, d) + n(g, h)\} \right\}, \end{aligned}$$

$$W' = w, \dots, W, \quad L' = w, \dots, L \quad \text{with} \quad \max\{L', W'\} \geq \ell.$$



**Fig. 10.5** Notations used in the G4-recursion

Then, the basic version of the G4 heuristic consists in applying the formula for  $n(L', W')$  for all  $L' \in \mathbb{Z}$  and  $W' \in \mathbb{Z}$  with  $0 < W' \leq W$  and  $0 < L' \leq L$ . However, this naive approach possesses a number of opportunities to save computational effort which we will consider in the next subsection.

Now we concentrate on PLP instances of type I which are characterized by  $1 \leq L/W \leq 2$ ,  $1 \leq \ell/w \leq 4$ , and  $6 \leq LW/(lw) \leq 50$ , and which are of particular interest for practical purposes.

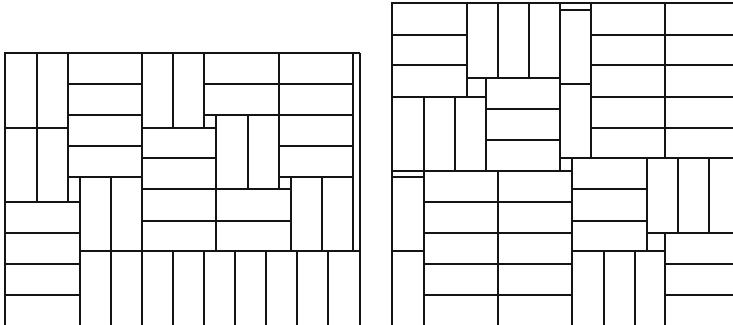
**Theorem 10.3** *The G4 heuristic computes an optimal pattern for each PLP instance of type I.*

*Proof* First of all, we show that the G4 heuristic yields patterns not worse than that obtainable by the previously mentioned heuristics. As shown in Exercise 10.3, any  $k$ -block pattern with  $k \leq 5$  has G4-structure. Furthermore, 7-block patterns generated by the *7-block heuristics*, (introduced in [7, 8]) have G4-structure, too. The *9-block heuristic*, proposed in [8], also constructs patterns with G4-structure. Moreover, the *diagonal heuristics*, proposed in [8, 15], generate patterns with G4-structure as well.

In [16] some further heuristics are proposed, the *angle heuristic*, the *recursive angle heuristic*, the *recurrence heuristic*, and the *complex block heuristic*. All these heuristics generate patterns with G4-structure. The essential difference in comparison to general G4-patterns consists in the fact that the other heuristics only allow homogeneous patterns in some or all blocks.

Summarizing, we can state that any pattern obtained by one of the considered heuristics possesses the G4-structure. Hence, if a best G4-pattern for a PLP instance is computed, then a pattern is obtained at least as good as obtainable by the other heuristics.

In [16] all equivalence classes of PLP instances of type I are examined. For each of the 8274 instances belonging to COVER I but 4, at least one of the heuristics mentioned above yields an optimal pattern. Since the G4 heuristic is at least as good as the other ones which are considered in [16], the investigation



**Fig. 10.6** Optimal patterns of the instances  $(57, 44, 12, 5)$  and  $(56, 52, 12, 5)$

of four representatives remains. The G4 heuristic also computes optimal patterns for these four instances [18]. These patterns are shown in Figs. 10.1 and 10.6, respectively. ■

### 10.2.4 Improvements

Initially, we can state that it is not necessary to compute G4-patterns with  $b+d > W'$  since they are reflections of the case  $f+h > W$  (cf. Fig. 10.5). According to the concept of *normalized* patterns and of (reduced) sets of potential allocation points (Sect. 2.7), the computational effort can be reduced essentially, in general. To this end, let  $S(L)$  and  $S(W)$  denote the reduced sets of potential allocation points in  $L$ - and  $W$ -direction, respectively. Moreover, the application of appropriate upper bounds for  $n(L, W)$  can be meaningful. If during the calculation of  $n(L, W)$  it can be recognized that the optimal value is already found (since equal to an upper bound), then the recursion can early be terminated. The following upper bounds (Sect. 10.1.3) are applied:

1. the area bound  $ub_a(L, W) := \lfloor (LW)/(lw) \rfloor$ ,
2. the Barnes bound  $ub_B(L, W)$ ,
3. the minimal area bound  $ub_e(L, W)$  of Dowsland,
4. the LP bound  $ub_{LP}(L, W)$  of Isermann.

Since their computation requires different effort their application is processed in the presented sequence if still necessary.

In order to compute  $n(L, W)$  all necessary optimal values  $n(L', W')$  have to be obtained before. As an initialization, we set

$$n(w, W') := \lfloor W'/\ell \rfloor, \quad W' \in S(W), \quad n(L', w) := \lfloor L'/\ell \rfloor, \quad L' \in S(L).$$

**G4 heuristic: computation of  $n(L', W')$** 

- (1) Symmetry: If  $W' < L'$  and  $L' \leq W$ , then set  $n(L', W') := n(W', L')$ , stop (i.e.,  $n(L', W')$  is obtained).
- (2) Compute the best homogeneous pattern (let  $n_1$  denote the number of items in that pattern) and the area bound  $u := ub_a(L', W')$ . If  $n_1 = u$ , then stop.
- (3) Comparison with solutions for smaller pallets:  
Set  $n_2 := \max\{n(L' - 1, W'), n(L', W' - 1), n_1\}$ . If  $n_2 = u$ , stop.
- (4) Applying the Barnes bound:  
Set  $u := \min\{u, ub_B(L', W')\}$ . If  $n_2 = u$ , then stop.
- (5) Guillotine structure: Compute  

$$n_3 := \max \left\{ n_2, \max_{a \in S_0(L'/2)} n_V(L', W', a), \max_{b \in S_0(W'/2)} n_H(L', W', b) \right\}.$$
If  $n_3 = u$ , then stop.
- (6) Computation of further bounds:  
Set  $u := \min\{u, ub_e(L', W')\}$ . If  $n_3 = u$ , then stop.  
Set  $u := \min\{u, ub_{LP}(L', W')\}$ . If  $n_3 = u$ , then stop.
- (7) 4-block structure: Calculate the maximal number  $n_4$  of items for a pattern with 4-block structure.

**Fig. 10.7** The G4 heuristic for the PLP

Then the strategy given in Fig. 10.7 is applied to compute  $n(L', W')$  for a pallet  $P(L', W')$  with  $L' \in S(L)$ ,  $W' \in S(W)$ ,  $L' > w$ ,  $W' > w$ , and  $\max\{L', W'\} \geq \ell$  where  $W'$  varies first for fixed  $L'$ . Moreover, in the algorithm a set  $S_0(L)$  is used which is defined as  $S_0(L) := \{r : r = \ell p + wq, r \leq L, p, q \in \mathbb{Z}_+\}$ .

Note that some additional tests to *stop* are possible during the computation of  $n_3$  and  $n_4$ . Since the calculation of  $n_4$  corresponds to the variation of four parameters, several possibilities arise to stop the four loops (for details, see [18]).

Nelißen [16] investigated instances of type II as well which differ from that of type I only by the condition  $51 \leq LW/(\ell w) < 101$ . The corresponding set COVER II of representatives (of the equivalence classes) has cardinality  $\text{card}(\text{COVER II}) = 41.831$ . For all instances of COVER II, but 206, at least one of the heuristics considered in [16] finds an optimal pattern. Remarkably, the G4 heuristic solves further 167 of these 206 instances. Moreover, the gap between the optimal value (or the best upper bound if unknown) and the G4-value is 1 for die remaining 39 instances.

To give an impression of the effects of applying various bounds, we quote some results from [18] in Table 10.1. The G4 heuristic is compared to two variants in which either only the LP bound  $ub_{LP}$  of Isermann or additionally the equivalence bound  $ub_e$  are not used.

The usefulness of both bounds  $ub_{LP}$  and  $ub_e$  also appears in Table 10.2 where results for instances with  $W=200$ ,  $\ell=21$ ,  $w=19$ , and the  $L$ -values  $L=200, 250, \dots, 450$  are given.

**Table 10.1** Comparison of computational times for the 206 instances of COVER II

|                                     | Average value | Maximal value |
|-------------------------------------|---------------|---------------|
| G4 heuristic                        | 0.22          | 0.55          |
| Without bound $ub_{LP}$             | 0.12          | 0.33          |
| Without bounds $ub_{LP}$ and $ub_e$ | 0.25          | 1.43          |

**Table 10.2** Comparison of variants in dependence on  $L$ 

| $L =$                               | 200  | 250  | 300  | 350  | 400   | 450   |
|-------------------------------------|------|------|------|------|-------|-------|
| G4-value                            | 100  | 125  | 149  | 175  | 200   | 225   |
| G4 heuristic                        | 0.22 | 0.28 | 1.15 | 2.20 | 3.40  | 20.54 |
| Without bound $ub_{LP}$             | 0.11 | 0.22 | 0.88 | 1.97 | 3.24  | 21.42 |
| Without bounds $ub_{LP}$ and $ub_e$ | 0.49 | 2.52 | 6.65 | 9.17 | 33.84 | 47.13 |

A small improvement of the G4 heuristic is proposed in [14] considering patterns with 5-block structure. Patterns with 5-block structure can be computed online: <http://familiamartins.com/plp/index.htm>.

## 10.3 The Guillotine Pallet Loading Problem

Within this section we consider a particular PLP where the guillotine condition has to be met, and we describe a polynomial-time algorithm for it, proposed in [20].

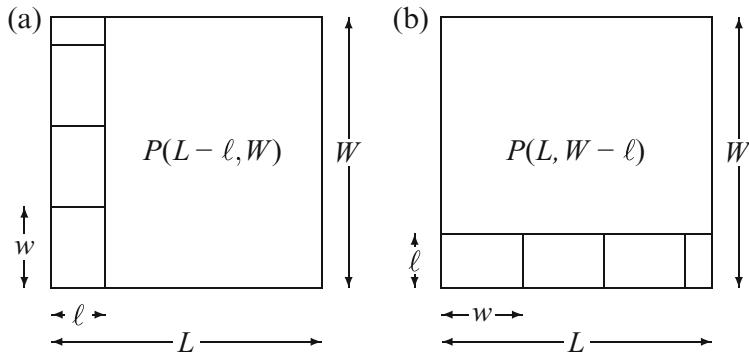
### 10.3.1 Basic Model and Algorithm

The *Guillotine Pallet Loading Problem* (GPLP) asks for a guillotine pattern with maximal number of placed rectangular items of size  $\ell \times w$  or  $w \times \ell$  for a given pallet  $P(L, W)$ . Again, we assume  $L, W, \ell, w \in \mathbb{N}$  and  $\gcd(\ell, w) = 1$ . In difference to the previous section, here we assume that  $\ell < w$  holds.

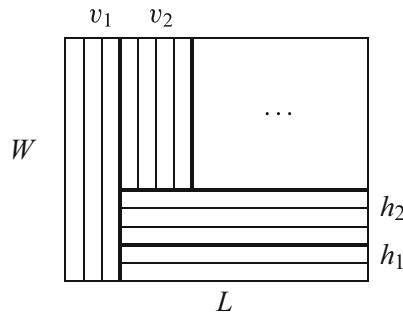
For pallet (rectangle)  $P(L, W)$  we define an  $\ell$ -strip which either is a vertical (Fig. 10.8a) or a horizontal (Fig. 10.8b) strip of width  $\ell$ . An  $\ell$ -strip of a rectangle  $P(L, W)$  has to fulfill the following conditions:

1. The length  $\mathcal{L}$  of an  $\ell$ -strips is equal to  $W$  or  $L$ .
2. An  $\ell$ -strip contains  $\lfloor \mathcal{L}/w \rfloor$  items  $\ell \times w$ .
3. An  $\ell$ -strip is placed *bottom-left justified*, i.e., its allocation point is the lower-left corner of  $P(L, W)$ .

A (packing) pattern is called  $\ell$ -pattern for  $P(L, W)$  if it consists of a single  $\ell$ -strip, or if it contains a vertical or horizontal  $\ell$ -strip and an  $\ell$ -pattern for  $P(L - \ell, W)$  or for  $P(L, W - \ell)$ , respectively, according to Fig. 10.8. It is easy to see that



**Fig. 10.8** Two types of  $\ell$ -strips. (a) vertical  $\ell$ -strip. (b) horizontal  $\ell$ -strip



**Fig. 10.9** Principal shape of a block- $\ell$ -pattern

each guillotine pattern can be transformed into an  $\ell$ -pattern by translations and reflections. Therefore, only  $\ell$ -patterns have to be considered to obtain an optimal pattern for the GPLP. Note that such kind of *normalization* is not possible if more than two (different) non-rotatable items are available. Similar to the  $\ell$ -strips, we can define  $w$ -strips and  $w$ -patterns (see Exercise 10.5).

Each  $\ell$ -pattern induces a partition of  $P(L, W)$  in a sequence of vertical and horizontal  $\ell$ -strips. We describe this partition by a sequence of binary variables

$$\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_t \quad \text{where} \quad t = n + m \quad \text{with} \quad n := \lfloor L/\ell \rfloor, m := \lfloor W/\ell \rfloor.$$

Variable  $\tilde{x}_i$  gets value 1 if the  $i$ th  $\ell$ -strip is vertical, and 0, if it is horizontal. Note that empty  $\ell$ -strips are possible.

An  $\ell$ -pattern is a particular case of a *block- $\ell$ -pattern* where a block consists of a number of parallel  $\ell$ -strips (Fig. 10.9).

A block- $\ell$ -pattern is defined by the number  $r$  of vertical homogeneous blocks and the number  $s$  of horizontal homogeneous blocks, non-negative block-sizes  $v_1, \dots, v_r$  and  $h_1, \dots, h_s$ , which fulfill  $\sum_{i=1}^r v_i = n$  and  $\sum_{i=1}^s h_i = m$ , and a

sequence  $x_1, \dots, x_{r+s}$  of binary variables. The *block-size*  $v_i$  is the number of vertical  $\ell$ -strips within the  $i$ th vertical block, and  $h_i$  is the number of horizontal  $\ell$ -strips in a horizontal block. The binary variable  $x_k$  has value 1 if the  $k$ th block is vertical, 0 otherwise. In the following we identify a block with its block-size  $v_i$  or  $h_i$ .

The 0/1 problem to find an optimal sequence of vertical and horizontal blocks for given block-sizes  $v_1, \dots, v_r, h_1, \dots, h_s$ , which aims to maximize the total number of placed items, is called *guillotine block PLP*. Its solution is described next.

Let us consider the waste of an  $\ell$ -pattern. For a vertical  $\ell$ -strip which is placed on top of  $y \in \mathbb{Z}_+, y \leq m$ , horizontal  $\ell$ -strips, the unused area is equal to  $\ell \cdot E_0(y)$  where

$$E_0(y) := (W - \ell y) \bmod w, \quad y \in \mathbb{Z}_+, \quad y \leq m. \quad (10.2)$$

Analogously, the waste of a horizontal  $\ell$ -strip which is placed right to  $y \in \mathbb{Z}_+, y \leq n$ , vertical  $\ell$ -strips is equal to  $\ell \cdot E_1(y)$  where

$$E_1(y) := (L - \ell y) \bmod w, \quad y \in \mathbb{Z}_+, \quad y \leq n.$$

*Example 10.1* Consider the instance  $(L, W, w, \ell) = (40, 25, 7, 3)$  of the PLP. Then we have  $n = \lfloor L/\ell \rfloor = 13$  and  $m = \lfloor W/\ell \rfloor = 8$ . Furthermore, it holds:

|          |   |   |   |   |   |   |   |   |   |   |    |    |    |    |
|----------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|
| $y$      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| $E_0(y)$ | 4 | 1 | 5 | 2 | 6 | 3 | 0 | 4 | 1 |   |    |    |    |    |
| $E_1(y)$ | 5 | 2 | 6 | 3 | 0 | 4 | 1 | 5 | 2 | 6 | 3  | 0  | 4  | 1  |

We consider now a model of the *block- $\ell$ -pattern problem*. The block-sizes  $v_1, \dots, v_r$  and  $h_1, \dots, h_s$  are given. Each block- $\ell$ -pattern has exactly  $n$  vertical and  $m$  horizontal  $\ell$ -strips as well as  $r$  vertical and  $s$  horizontal blocks. For  $x \in \mathbb{B}^{r+s}$ , the condition  $\sum_{i=1}^{r+s} x_i = r$  ensures that exactly  $r$  vertical blocks occur within the pattern. The numbers

$$\varrho(i) := \sum_{j=1}^i x_j \quad \text{and} \quad \eta(i) := \sum_{j=1}^i (1 - x_j)$$

indicate how many vertical and horizontal blocks are present up to the  $i$ th block. Then the total waste for a given  $x$  can be characterized by

$$M(x) = \sum_{i=1}^{r+s} \left\{ x_i v_{\varrho(i)} E_0 \left( \sum_{t=1}^{\eta(i)} h_t \right) + (1 - x_i) h_{\eta(i)} E_1 \left( \sum_{t=1}^{\varrho(i)} v_t \right) \right\}.$$

More precisely, the block- $\ell$ -pattern problem with  $r$  vertical and  $s$  horizontal predefined block sizes can be stated as follows:

$$S_r(x) := S_0 + \ell \cdot \min \left\{ M(x) : \sum_{i=1}^{r+s} x_i = r, x_i \in \{0, 1\}, i = 1, \dots, r+s \right\} \quad (10.3)$$

where  $S_0 = (L \bmod \ell)(W \bmod \ell)$ . In the following, we describe the waste only by means of function  $M(x)$ . It is easy to see, the block- $\ell$ -pattern problem can be solved using *dynamic programming*. To this end, let  $(i, j)$  denote the set of all  $\ell$ -block sequences with  $i$  vertical blocks  $v_1, \dots, v_i$  and  $j$  horizontal blocks  $h_1, \dots, h_j$ . Moreover,  $H(i, j)$  denotes the minimal waste [with respect to  $M(x)$ ] for the state  $(i, j)$ . Then, with  $H(0, 0) := 0$ ,  $H(1, 0) := v_1 E_0(0)$  and  $H(0, 1) := h_1 E_1(0)$ , recursion

$$H(i, j) = \min \left\{ H(i-1, j) + v_i E_0 \left( \sum_{t=1}^j h_t \right), H(i, j-1) + h_j E_1 \left( \sum_{t=1}^i v_t \right) \right\} \quad (10.4)$$

for  $i = 1, 2, \dots, r$  and  $j = 1, 2, \dots, s$  can be applied to obtain  $H(r, s)$ . It remains to identify appropriate block-sizes  $v_i$  and  $h_j$ .

### 10.3.2 Definition of Subproblems

As we will see, the GPLP can be partitioned into three independent subproblems. Belonging to the functions

$$E_0(y) = (W - \ell y) \bmod w, \quad E_1(y) = (L - \ell y) \bmod w, \quad y \in \mathbb{Z}_+,$$

we define sets of global minimum points

$$A_0 := \{ \sigma : E_0(\sigma) = \min_{0 \leq y \leq m} E_0(y) \}, \quad A_1 := \{ \sigma : E_1(\sigma) = \min_{0 \leq y \leq n} E_1(y) \}.$$

**Theorem 10.4** *Let  $\sigma_0 \in A_0$ ,  $\sigma_1 \in A_1$ ,  $k := \sigma_0 + \sigma_1$ . Then there always exists an optimal  $\ell$ -pattern with the property that the first  $k$   $\ell$ -strips contain  $\sigma_0$  horizontal and  $\sigma_1$  vertical  $\ell$ -strips.*

*Proof* Let  $x$  be an optimal  $\ell$ -pattern and assume that  $x$  does not fulfill this property. Furthermore, let  $\varrho(k) < \sigma_1$ ,  $\eta(k) > \sigma_0$ , i.e., there are more than  $\sigma_0$  zeros within the first  $k$  components of  $x$ . Let  $a$  and  $b$  be two indices defined by

$$x_a = 0 \text{ and } \eta(a) = \sigma_0, \quad x_b = 1 \text{ and } \varrho(b) = \sigma_1,$$

then we have  $a < k < b$ . We define an  $\ell$ -pattern  $x'$  according to

$$x'_i = \begin{cases} x_i & \text{for } i = 1, \dots, a, b+1, \dots, n+m, \\ 1 & \text{for } i = a+1, \dots, k, \\ 0 & \text{for } i = k+1, \dots, b. \end{cases}$$

Then,

$$\begin{aligned} M(x) - M(x') &= \sum_{i=a+1}^b [x_i E_0(\eta(i)) + (1-x_i) E_1(\varrho(i))] \\ &\quad - (\sigma_1 - \varrho(a)) E_0(\sigma_0) - (b - \sigma_1 - \sigma_0) E_1(\sigma_1) \\ &= \sum_{i=a+1}^b [x_i \underbrace{(E_0(\eta(i)) - E_0(\sigma_0))}_{\geq 0} + (1-x_i) \underbrace{(E_1(\varrho(i)) - E_1(\sigma_1))}_{\geq 0}] \\ &\geq 0. \end{aligned}$$

That means,  $x'$  represents an optimal pattern as well. The proof for the case  $\varrho(k) > \sigma_1$ ,  $\eta(k) < \sigma_0$  is similar. ■

Furthermore, let particular minimum points  $\sigma_0^-, \sigma_0^+, \sigma_1^-, \sigma_1^+$ , be defined by

$$\begin{aligned} \sigma_0^- &:= \min\{\sigma : \sigma \in A_0\}, & \sigma_0^+ &:= \max\{\sigma : \sigma \in A_0\}, \\ \sigma_1^- &:= \min\{\sigma : \sigma \in A_1\}, & \sigma_1^+ &:= \max\{\sigma : \sigma \in A_1\}. \end{aligned}$$

Then Theorem 10.4 holds in particular for  $\sigma_0 = \sigma_0^-$  and  $\sigma_1 = \sigma_1^-$ .

*Example 10.2 (Continuation of Example 10.1)* Because of  $A_0 = \{6\}$  we have  $\sigma_0^- = \sigma_0^+ = 6$  and, because of  $A_1 = \{4, 11\}$ ,  $\sigma_1^- = 4$  and  $\sigma_1^+ = 11$  hold. ■

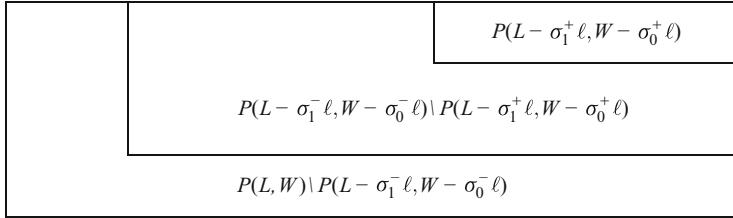
**Theorem 10.5** Let  $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_{n+m})^\top$  be an optimal  $\ell$ -pattern which meets the property of Theorem 10.4 with  $k = \sigma_0^- + \sigma_1^-$ . Furthermore, let  $r = \sigma_0^+ + \sigma_1^+$ . Then the  $\ell$ -patterns

$$x' = (\tilde{x}_1, \dots, \tilde{x}_k, \underbrace{1, 1, \dots, 1}_{\sigma_1^+ - \sigma_1^-}, \underbrace{0, 0, \dots, 0}_{\sigma_0^+ - \sigma_0^-}, \tilde{x}_{r+1}, \dots, \tilde{x}_{n+m})^\top$$

and

$$x^* = (\tilde{x}_1, \dots, \tilde{x}_k, \underbrace{0, 0, \dots, 0}_{\sigma_0^+ - \sigma_0^-}, \underbrace{1, 1, \dots, 1}_{\sigma_1^+ - \sigma_1^-}, \tilde{x}_{r+1}, \dots, \tilde{x}_{n+m})^\top$$

are optimal  $\ell$ -patterns, too.



**Fig. 10.10** Partition of  $L \times W$  in three subregions

*Proof* From Theorem 10.4 we know that an optimal  $\ell$ -pattern exists which has  $\sigma_1^+ - \sigma_1^-$  vertical and  $\sigma_0^+ - \sigma_0^-$  horizontal  $\ell$ -strips as the  $(k + 1)$ st till  $r$ th  $\ell$ -strips. The remaining proof to show the optimality of  $x'$  is analogous to the proof of Theorem 10.4 if  $a := k$ ,  $\sigma_1 := \sigma_1^+$ ,  $b := \sigma_0^+ + \sigma_1^+$ , and  $\ell$ -pattern  $x'$  is considered. Hence,  $\varrho(a) = \sigma_1^-$ . The optimality of  $x^*$  can be shown in a similar way. ■

Due to Theorems 10.4 and 10.5, we obtain

**Corollary 10.6** *An optimal  $\ell$ -pattern  $x$  of the GPLP can be composed as follows:*

$$\tilde{x} = (x_1^1, x_2^1, \dots, x_{k_1}^1, x_1^2, x_2^2, \dots, x_{k_2}^2, x_1^3, x_2^3, \dots, x_{k_3}^3)^\top$$

where  $x^1 = (x_1^1, x_2^1, \dots, x_{k_1}^1)^\top$  with  $k_1 = \sigma_0^- + \sigma_1^-$ ,  $x^2 = (x_1^2, x_2^2, \dots, x_{k_2}^2)^\top$  with  $k_2 = \sigma_0^+ - \sigma_0^- + \sigma_1^+ - \sigma_1^-$ , and  $x^3 = (x_1^3, x_2^3, \dots, x_{k_3}^3)^\top$  with  $k_3 = m - \sigma_0^+ + n - \sigma_1^+$  are optimal  $\ell$ -patterns of the following subproblems:

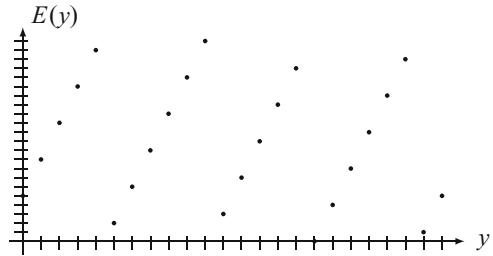
1. Compute  $x^1$  as an optimal  $\ell$ -pattern for the region  $P(L, W) \setminus P(L - \sigma_1^- \ell, W - \sigma_0^- \ell)$  which has  $\sigma_0^-$  horizontal and  $\sigma_1^-$  vertical  $\ell$ -strips.
2. Choose  $x^2$  as an optimal  $\ell$ -pattern of the region  $P(L - \sigma_1^- \ell, W - \sigma_0^- \ell) \setminus P(L - \sigma_1^+ \ell, W - \sigma_0^+ \ell)$  which has  $\sigma_0^+ - \sigma_0^-$  horizontal and  $\sigma_1^+ - \sigma_1^-$  vertical  $\ell$ -strips according to Theorem 10.5.
3. Compute  $x^3$  as an optimal  $\ell$ -pattern of the region  $P(L - \sigma_1^+ \ell, W - \sigma_0^+ \ell)$  which has  $m - \sigma_0^+$  horizontal and  $n - \sigma_1^+$  vertical  $\ell$ -strips.

Each subproblem can be solved independently of each other (Fig. 10.10).

### 10.3.3 Properties of $E_0$ and $E_1$

Let  $E(y) = (\alpha_0 + \alpha y) \bmod \beta$ ,  $y \in \mathbb{Z}_+$ . We assume that  $\alpha_0, \alpha, \beta \in \mathbb{Z}_+$ ,  $\gcd(\alpha, \beta) = 1$ ,  $\alpha_0 < \beta$ ,  $\alpha < \beta$ , and  $\alpha \neq 0$  hold. Figure 10.11 shows the example  $E(y) = (5 + 4y) \bmod 23$ . Moreover, let  $\tilde{y}_0, \dots, \tilde{y}_r$  represent the sequence of minimum points of  $E$  according to

**Fig. 10.11** Modulo function  
 $E(y) = (5 + 4y) \bmod 23$



1.  $\tilde{y}_0 = 0; \tilde{y}_r = \sigma^-$  where  $\sigma^- = \min \{ \sigma : E(\sigma) = \min \{ E(y) : y \in \mathbb{Z}_+ \} \}$ ,
2.  $E(y) > E(\tilde{y}_i)$  for  $y = 0, 1, \dots, \tilde{y}_i - 1, \quad i = 1, \dots, r,$
3.  $E(y) > E(\tilde{y}_i)$  for  $y = \tilde{y}_i + 1, \dots, \tilde{y}_{i+1} - 1, \quad i = 1, \dots, r - 1.$

Hence,  $\tilde{y}_i$  indicates the global minimum point of  $E(y)$ ,  $y \in [0, \tilde{y}_{i+1}) \cap \mathbb{Z}$ , and we have  $E(\tilde{y}_0) > \dots > E(\tilde{y}_r) = 0$ . Furthermore, we define a subsequence  $y_1^*, \dots, y_{r^*}^*$  of  $\tilde{y}_1, \dots, \tilde{y}_r$  such that all elements  $\tilde{y}_i$  with

$$E(\tilde{y}_i) = E(\tilde{y}_{i-1}) + \frac{E(\tilde{y}_{i+1}) - E(\tilde{y}_{i-1})}{\tilde{y}_{i+1} - \tilde{y}_{i-1}} (\tilde{y}_i - \tilde{y}_{i-1})$$

are removed. Below we show that

$$r^* \leq \log_2 \beta$$

holds and that the values  $y_1^*, \dots, y_{r^*}^*$  can be computed in polynomial time  $O(\Theta \log_2 \beta)$  where  $\Theta$  denotes the necessary (binary) length of the input data.

Obviously, function  $E(y)$ ,  $y \in \mathbb{Z}_+$ , is periodic with length  $\beta$  of a period and it is piece-wise linear on  $\mathbb{Z}_+$ . We call function  $E(y)$  discrete increasing, if there does not exist any  $y^*$  with  $E(y^*) > E(y^* + 1) > E(y^* + 2)$  (Fig. 10.11).  $E(y)$  is discrete decreasing, if there is no  $y^*$  with  $E(y^*) < E(y^* + 1) < E(y^* + 2)$ .

In case of  $\alpha < \beta/2$ , function  $E(y)$  is discrete increasing and the linear parts have gradient (ascent)  $\alpha$ . If  $\alpha > \beta/2$ , then function  $E(y)$  is discrete decreasing and, because of  $E(y) = (\alpha_0 + \alpha y) \bmod \beta = (\alpha_0 - (\beta - \alpha)y) \bmod \beta$ , the linear parts of it have gradient  $-(\beta - \alpha)$ . Due to the definition of the modulo-function,  $E(y) \geq 0$ ,  $y \in \mathbb{Z}_+$ , holds. Because of  $\gcd(\alpha, \beta) = 1$ , we obtain  $\min\{E(y) : y \in \mathbb{Z}_+, 0 \leq y \leq \beta\} = 0$ . Let

$$E^0(y) := (\alpha_0^0 + \alpha^0 y) \bmod \beta^0 \equiv E(y) = (\alpha_0 + \alpha y) \bmod \beta.$$

Function  $E^0(y)$  possesses the following set of local minimum points

$$Y^1 = \{y : y \in \mathbb{Z}_+, E(y) < \min\{\alpha^0, \beta^0 - \alpha^0\}\} =: \{z_0, z_1, \dots\}.$$

The values  $E^0(y)$  on set  $Y^1$  define a new function  $E^1(z)$ ,  $z \in \mathbb{Z}_+$ , which is a modulo-function as well as shown in [19]. It can be computed as follows:

$$E^1(y) := (\alpha_0^1 + \alpha^1 y) \bmod \beta^1$$

where

$$\begin{aligned}\alpha_0^1 &= E(z_0), \\ \beta^1 &= \min\{\alpha^0, \beta^0 - \alpha^0\}, \\ \alpha^1 &= \begin{cases} \alpha^0 - \beta^0 \bmod \alpha^0, & \text{if } \alpha^0 \leq \beta^0/2, \\ \beta^0 \bmod (\beta^0 - \alpha^0), & \text{if } \alpha^0 > \beta^0/2. \end{cases}\end{aligned}$$

Therefore,  $\beta^1 \leq \beta^0/2$  holds. In this manner, we define a sequence of functions

$$E^k(y) = (\alpha_0^k + \alpha^k y) \bmod \beta^k, \quad k = 0, 1, 2, \dots$$

Because of

$$\beta^k \leq \beta^{k-1}/2, \quad k = 1, 2, \dots$$

this process ends at value  $k = k^*$  with  $\beta^{k^*} = 1$  and, obviously,

$$k^* \leq \log_2 \beta$$

holds. For  $k \in \{0, 1, \dots, k^* - 1\}$ , set

$$Y^{k+1} := \{y : y \in \mathbb{Z}_+, E(y) < \min\{\alpha^k, \beta^k - \alpha^k\}\}$$

contains all local minimum points of  $E(y)$  having a function value smaller than  $\min\{\alpha^k, \beta^k - \alpha^k\} = \beta^{k+1}$  and we have

$$\{y_1^*, \dots, y_{r^*}^*\} \subseteq Y^* := \{y : y = \min\{t : t \in Y^k\}, k = 1, \dots, k^*\}.$$

This is true since the presumption that  $y_s^*$  is not the first element in a set  $Y^k$  leads to a contradiction to the definition of  $y_s^*$ . Consequently,  $r^*$  depends polynomially on  $\log_2 \beta$ .

Next, we consider function  $E(y)$  on the finite set  $\{0, 1, 2, \dots, \zeta\}$  for a given integer  $\zeta > 0$ . Moreover, in the case  $\zeta < \min\{y : y \in \mathbb{Z}_+, E(y) = 0\}$ , we define  $\sigma^- := \min\{\sigma : E(\sigma) = \min\{E(y) : 0 \leq y \leq \zeta\}\}$  and  $k^*$  is determined by

$$Y^{k^*} \cap \{0, 1, \dots, \zeta\} = \{\sigma^-\}.$$

To investigate optimal block- $\ell$ -patterns, we consider set  $Y^*$  and define  $y_k^* := \min\{t : t \in Y^k\}$ ,  $k = 1, \dots, k^*$ . In the way mentioned above, the values  $k^*$ ,  $y_1^*, \dots, y_{k^*}^*$  can be computed in polynomial time for a given function  $E(y) = (\alpha_0 + \alpha y) \bmod \beta$ ,  $y \in \{0, 1, \dots, \zeta\}$ . A detailed algorithm, called PARTITION, is described in [20].

### 10.3.4 Algorithm for Subproblem 1

Let  $y_0, \dots, y_r$  denote the sequence of minimum points of  $E_1$  with

- 1)  $y_0 = 0$ ,  $y_r = \sigma_1^-$ ,
- 2)  $E_1(y) > E_1(y_i)$  for  $y = 0, 1, \dots, y_i - 1$ ,  $i = 1, \dots, r$ ,
- 3)  $E_1(y) > E_1(y_i)$  for  $y = y_i + 1, \dots, y_{i+1} - 1$ ,  $i = 1, \dots, r - 1$ .

Furthermore, let  $z_0, \dots, z_s$  be the sequence of minimum points of  $E_0$ .

**Theorem 10.7** *There exists an optimal block- $\ell$ -pattern for subproblem 1 with vertical block-values*

$$v_1 := y_1, \quad v_i := y_i - y_{i-1}, \quad i = 2, \dots, r,$$

and horizontal block-values

$$h_1 := z_1, \quad h_j := z_j - z_{j-1}, \quad j = 2, \dots, s.$$

*Proof* Let  $x = (x_1, \dots, x_d)^\top$  with  $d := \sigma_0^- + \sigma_1^-$  be an optimal  $\ell$ -pattern of subproblem 1. We perform the proof by a backward induction. For the general case, we suppose that for some index  $k$  with  $k < d$  the subsequence  $x_{k+1}, \dots, x_d$  of  $x$  forms a block- $\ell$ -pattern with vertical blocks of size  $v_{k_1}, \dots, v_r$  and horizontal blocks of size  $h_{k_0}, \dots, h_s$ . Thus,  $k = d - \sum_{t=k_1}^r v_t - \sum_{t=k_0}^s h_t$  and  $\varrho(k) = y_{k_1-1}$ ,  $\eta(k) = z_{k_0-1}$  hold. Let  $x_k = 1$ . Now we construct a new optimal  $\ell$ -pattern  $x'$  such that additionally

$$x'_t = 1 \quad \text{for } t = k - v_{k_1-1} + 1, \dots, k,$$

holds. Let index  $b$  be determined by

$$x_b = 1 \quad \text{and} \quad \varrho(b) = y_{k_1-2} + 1.$$

Then  $x'$  is defined as

$$x'_i := \begin{cases} x_i & \text{for } i = 1, \dots, b-1, k+1, \dots, d, \\ 0 & \text{for } i = b, \dots, k-v_{k_1-1}, \\ 1 & \text{for } i = k-v_{k_1-1} + 1, \dots, k. \end{cases}$$

Thus,

$$\begin{aligned} M(x) - M(x') &= \sum_{i=b}^k (x_i E_0(\eta(i)) + (1-x_i)E_1(\varrho(i))) \\ &\quad -(k-b-v_{k_1-1}+1)E_1(\varrho(b-1)) - v_{k_1-1}E_0(\eta(k-v_{k_1-1})) \\ &= \sum_{i=b}^k (x_i \underbrace{(E_0(\eta(i)) - E_0(z_{k_0-1}))}_{\geq 0} + (1-x_i) \underbrace{(E_1(\varrho(i)) - E_1(y_{k_1-2}))}_{\geq 0}) \\ &\geq 0. \end{aligned}$$

That means,  $x'$  is also an optimal  $\ell$ -pattern.

The second case with  $x_k = 0$  can be considered in a similar way. Let the index  $b$  be defined by  $x_b = 0$  and  $\eta(b) = z_{k_0-2} + 1$ , and let  $x'$  be defined by

$$x'_i := \begin{cases} x_i & \text{for } i = 1, \dots, b-1, k+1, \dots, d, \\ 1 & \text{for } i = b, \dots, k-h_{k_0-1}, \\ 0 & \text{for } i = k-h_{k_0-1} + 1, \dots, k. \end{cases}$$

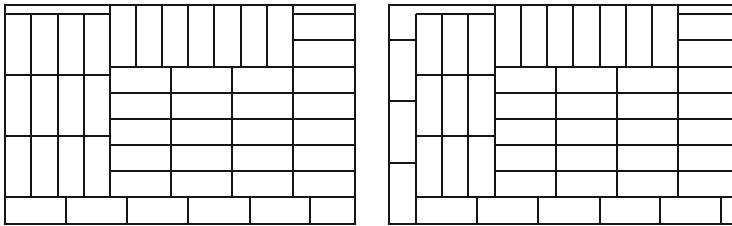
Then,

$$\begin{aligned} M(x) - M(x') &= \sum_{i=b}^k (x_i E_0(\eta(i)) + (1-x_i)E_1(\varrho(i))) \\ &\quad -(k-b-h_{k_0-1}+1)E_0(\eta(b-1)) - h_{k_0-1}E_1(\varrho(k-h_{k_0-1})) \\ &= \sum_{i=b}^k (x_i \underbrace{(E_0(\eta(i)) - E_0(z_{k_0-2}))}_{\geq 0} + (1-x_i) \underbrace{(E_1(\varrho(i)) - E_1(y_{k_1-1}))}_{\geq 0}) \\ &\geq 0. \end{aligned}$$

Hence, in both cases, we have found optimal  $\ell$ -patterns which contain a further block of desired size. ■

**Theorem 10.8** *There exists an optimal block- $\ell$ -pattern of subproblem 1 with vertical blocks of size  $v_1^*, \dots, v_{r^*}^*$  and horizontal blocks of size  $h_1^*, \dots, h_{s^*}^*$  where  $v_1^*, \dots, v_{r^*}^*$  and  $h_1^*, \dots, h_{s^*}^*$  give the block-sizes which can be computed according to the approach described above (with algorithm PARTITION) for  $E_1(y)$ ,  $y \in \{0, \dots, \sigma_1^-\}$  and  $E_0(y)$ ,  $y \in \{0, 1, \dots, \sigma_0^-\}$ .*

A proof can be found in [20]. Clearly, subproblem 3 is similar to subproblem 1.



**Fig. 10.12** Optimal patterns of instance (40,25,7,3) for the GPLP

*Example 10.3 (Continuation of Example 10.2)* For  $E_0$  and  $E_1$  we have  $r = s = 2$  and  $v_1 = 1$ ,  $v_2 = 3$ ,  $h_1 = 1$ ,  $h_2 = 5$ . Applying recursion (10.4), we obtain the following table for  $H$ :

| $i \setminus j$ | 0  | 1 | 2  |
|-----------------|----|---|----|
| 0               | 0  | 5 | 30 |
| 1               | 4  | 6 | 16 |
| 2               | 16 | 9 | 9  |

Optimal sequences for the vertical and horizontal blocks are  $x^* = (1, 0, 1, 0)^\top$  and  $x^* = (0, 1, 1, 0)^\top$ . Therefore, 37 items are packed. Since seven items are placed in subproblem 2 and two items in subproblem 3, altogether 46 items are contained in an optimal guillotine pattern. In Fig. 10.12 both resulting patterns are shown. A pattern with 47 items which, consequently, does not possess the guillotine property, is drawn in Fig. 10.1.

Due to Theorems 10.4 and 10.5, an optimal pattern of the GPLP can be obtained by solving three subproblems. According to the concept of block- $\ell$ -patterns, it is proposed in Theorem 10.8 that an optimal block- $\ell$ -pattern for subproblem 1 with  $O(\log_2 w)$  blocks exists. That optimal pattern can be computed in polynomial time  $O(\Theta \log_2 w)$  with  $\Theta = \log_2 L$  using a dynamic programming approach. A solution of subproblem 2 is obtainable in constant time (Theorem 10.5). Obviously, an optimal pattern for subproblem 3 can be received in polynomial time similar to subproblem 1. Consequently, we have

**Theorem 10.9** *The Guillotine Pallet Loading Problem belongs to the class of problems solvable in polynomial time. There exists an algorithm with complexity  $O(\Theta \log_2^2 L)$ .*

## 10.4 The Pallet Loading Problem with Several Item Types

Besides the (classical) two-dimensional pallet loading problem where rectangular items of a single type have to be placed, the general case with several (many) types of items is of high interest. By means of a recursion formula presented below patterns with up to four item types can be computed with justifiable efforts.

More precisely, we consider the following problem. Given a (rectangular) pallet of length  $L$  and width  $W$ , we aim to pack (rectangular) items of types  $i \in I$  with  $I = \{1, \dots, m\}$  such that the area utilization is maximal and some restrictions (to be specified later on) are met. The items of type  $i$  are characterized by their lengths  $\ell_i$  and width  $w_i$ . Rotation of items by  $90^\circ$  is permitted.

In analogy to the G4 heuristic (Sect. 10.2), we use the *4-block structure*. Potential dissections of the pallet into four rectangular parts and corresponding notations are depicted in Fig. 10.13. We suppose that items of the same type only are packed within a block, and a pattern for a certain block is obtained by the G4 heuristic. Due to the 4-block structure of the generated patterns, we call this approach the *M4 heuristic*.

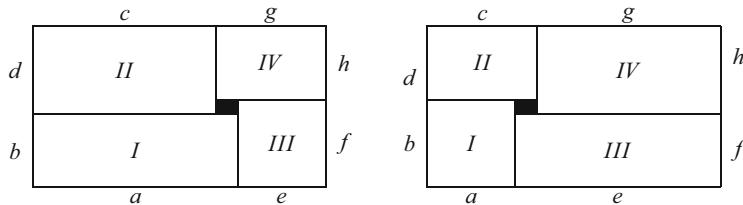
As Fig. 10.13 shows, due to symmetry reasons, only one of the two different dissections has to be considered to obtain a best possible pattern. We choose that one which is determined by  $a + g \geq L$  and  $b + h \leq W$ .

For a subrectangle  $p \times q$  of the pallet let  $v_i(p, q)$  denote the area occupied by items of type  $i \in I$  according to the G4 heuristic. Furthermore, let

$$\bar{v}(p, q) := \max_{i \in I} v_i(p, q)$$

denote the maximal value for a block of length  $p$  and width  $q$ . When applying the concept of reduced sets of potential allocation points, i.e., using  $\tilde{S}(\ell, L)$ ,  $\tilde{S}(w, W)$ , and  $p_S(x) := \max\{r \in S : r \leq x\}$ , cf. Sect. 2.7, we obtain the basic recursion

$$v(L, W) := \max_{a \in \tilde{S}_a} \max_{b \in \tilde{S}_b} \left\{ \bar{v}(a, b) + \max_{c \in \tilde{S}_c} \left\{ \bar{v}(c, d) + \max_{f \in \tilde{S}_f} \{\bar{v}(e, f) + \bar{v}(g, h)\} \right\} \right\} \quad (10.5)$$



**Fig. 10.13** 4-block-dissections of a pallet

**Table 10.3** Input data of Example 10.4

|        | Pallet | Type 1 | Type 2 | Type 3 | Type 4 | Type 5 | Type 6 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| Length | 1250   | 143    | 261    | 295    | 295    | 257    | 200    |
| Width  | 800    | 108    | 135    | 198    | 131    | 108    | 145    |

where

$$\begin{aligned} d &:= p_{\tilde{S}(w,W)}(W-b), & e &:= p_{\tilde{S}(\ell,L)}(L-a), \\ g &:= p_{\tilde{S}(\ell,L)}(L-c), & h &:= p_{\tilde{S}(w,W)}(W-f), \\ \bar{S}_a &:= \{s \in \tilde{S}(\ell,L) : s \geq L/2\}, & \bar{S}_b &:= \tilde{S}(w,W) \setminus \{0\}, \\ \bar{S}_c &:= \{s \in \tilde{S}(\ell,L) : s \leq a\}, & \bar{S}_f &:= \{s \in \tilde{S}(w,W) : s \geq b\}. \end{aligned}$$

Condition  $a \geq L/2$  in  $\bar{S}_a$  results from  $a + g \geq L$  and  $a \geq g$ . The definition of  $\bar{S}_f$  is based on the observation that any pattern with 4-block structure and  $f < b$  is dominated by a pattern where  $f \geq b$  holds for block III.

In order to easily regard additional restrictions, we consider a modification of the basic recursion (10.5):

$$v(L, W) := \max_{a \in \bar{S}_a} \max_{b \in \bar{S}_b} \left( \max_{i_1 \in I_1} \left\{ v_{i_1}(a, b) + \max_{c \in \bar{S}_c} \max_{i_2 \in I_2} \left\{ v_{i_2}(c, d) + \max_{f \in \bar{S}_f} \max_{i_3 \in I_3} \left\{ v_{i_3}(e, f) + \max_{i_4 \in I_4} v_{i_4}(g, h) \right\} \right\} \right\} \right)$$

where  $I_k \subseteq I$ ,  $k = 1, \dots, 4$ . Due to an appropriate choice of  $I_k$  various opportunities to handle particular situations arise. Without going into details, which are described in [17], we present some possibilities to regard additional demands by means of an example.

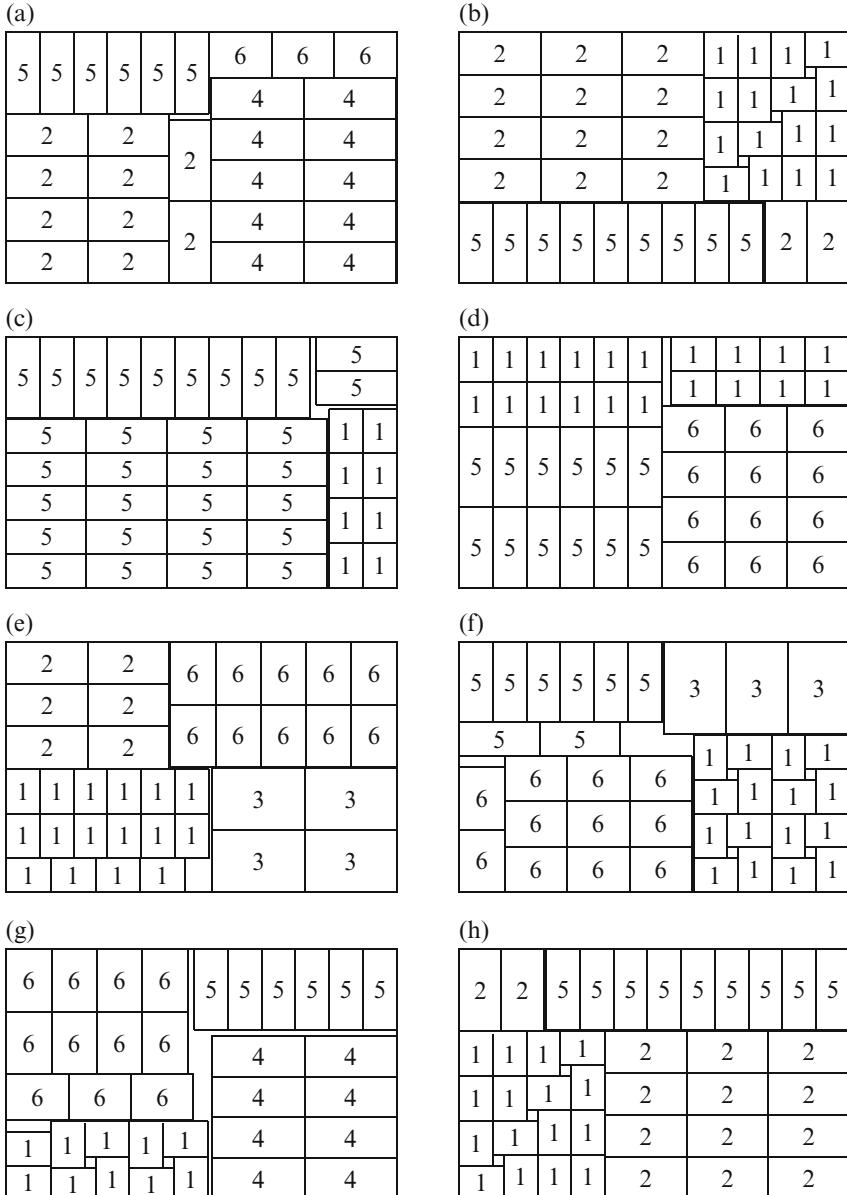
*Example 10.4* We consider an instance with input data given in Table 10.3. In Fig. 10.14a an area maximal pattern is shown. Note that  $10^{-4}v$  yields the percentage of utilization of the pallet area since  $L \cdot W = 10^6$ .

The patterns in (b) and (c) are obtained if the number  $t$  of different piece types is limited to  $t = 3$  and  $t = 2$ , respectively.

In Fig. 10.14d a pattern is shown resulting if items of the same type have to be placed in blocks II and IV.

The pattern in Fig. 10.14e is optimal with respect to limited availabilities of items with  $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$  where  $\bar{u}_i$  gives the maximum number items of type  $i$  are allowed to be packed. Since the upper bound for item types 2 and 4 is not met in variant (a), a smaller area utilization results.

In Fig. 10.14f additionally a minimum number of placed items of a certain type is demanded. If, moreover, eight items of type 4 have to be packed, then the pattern in Fig. 10.14g results.



**Fig. 10.14** Patterns of Example 10.4 regarding various additional restrictions. (a)  $v = 992, 336$ . (b)  $v = 990, 198$ ,  $t = 3$ . (c)  $v = 983, 988$ ,  $t = 2$ . (d)  $v = 989, 952$ , same type in II and IV. (e)  $v = 982, 154$ ,  $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$ . (f)  $v = 963, 382$ ,  $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$ ,  $\underline{u} = (0, 0, 0, 0, 0, 11)^\top$ . (g)  $v = 949, 136$ ,  $\bar{u} = (20, 9, 5, 8, 11, 11)^\top$ ,  $\underline{u} = (0, 0, 0, 8, 0, 11)^\top$ . (h)  $v = 990, 198$ , with sequencing condition

To obtain the pattern in Fig. 10.14h a sequencing condition is regarded. Within a block with higher number only items of such a type are allowed whose index is not smaller than that of the piece type in the previous block. ■

## 10.5 The Multi Pallet Loading Problem

In the previous sections we considered the (two-dimensional) problem of maximizing the area utilization for a single pallet. In terms of the manufacturer's PLP we aimed to maximize the number of (identical) items which can be packed on a pallet. For practical purposes its counterpart, how many pallets are needed to pack a given supply, is also of high interest. Moreover, besides the common packing constraints frequently additional restrictions have to be taken into account when constructing (three-dimensional) patterns.

As already mentioned above, the *Manufacturer's Pallet Loading Problem* [10] consists in finding a pattern with maximal number of identical (rotatable) cuboids (boxes). On the other hand, the problem to find the minimum number of (identical) pallets needed to pack a given assortment of various boxes is called the *Distributor's Pallet Loading Problem* in [4].

In the following, we consider the *Multi Pallet Loading Problem* (MPLP) as formulated in [21], i.e., the Distributor's Pallet Loading Problem.

### 10.5.1 Problem Statement

The (identical) pallets are characterized by their length  $L$ , width  $W$ , and height  $H$  where  $H$  indicates a maximum value up to which the items (their top surface) can be packed. Furthermore, let  $V = LWH$  denote the volume assigned to a pallet. Clearly, the total volume of the packed items must not exceed  $V$ . Let  $m$  be the number of different box types to be packed, and let  $I = \{1, \dots, m\}$ . Length, width, height, volume, weight, and supply of a box of type  $i$ ,  $i \in I$ , are denoted by  $\ell_i, w_i, h_i, v_i, p_i$ , and  $b_i$ , respectively. As usual, we always assume integer input data of an MPLP instance  $E$ .

Moreover, (orthogonal) rotations for some or all item types are permitted, i.e., the boxes have to be placed axis-parallel with their edges. The task is to pack all  $b_i$  boxes for all  $i \in I$  on a minimal number  $k^*(E)$  of pallets and to find corresponding (packing) patterns. A pattern is *feasible* if all of its boxes are packed axis-parallel, non-overlapping, and completely within the pallet volume. Probably, further restrictions have to be taken into account.

Restrictions meaningful in practical applications could be:

- *weight*: The total weight of boxes packed on a pallet does not exceed a given limit.

- *placement/location*: Since items of different types can have different density, frequently restrictions of form *boxes of type i are not allowed to be packed on boxes of type j*, have to be regarded. As the case may be, larger (heavier) boxes have to be placed on the bottom of the pallet, and not so stable boxes should be placed on top.
- *distribution*: If the supply  $b_i$  of item type  $i$  is as large such that a single pallet can be filled sufficiently good with only those items, then such an uniform pattern has to be preferred. Patterns of this kind can be computed in advance. In case supply  $b_i$  is not too large, then all items of this type should be placed on the same pallet. But, if the distribution of items of a single type on several pallets leads to a reduction of the total number of pallets needed, then such a splitting could be permitted.
- *connectivity*: To save time for the loading and unloading process very often patterns are looked for where all items of the same type (which have to be packed on the same pallet) have to be placed immediately one after another without interruption by another piece type.
- *stability*: In dependence on available supporting tools to stabilize a packed pallet, certain stability aspects have to be considered during the computation of a pattern, e.g. a sufficiently large supporting base area.

The aspects mentioned here lead in general to so-called *soft constraints*. If possible, they should be fulfilled, but certain exceptions will be accepted if a somewhat weakened constraint is met.

### 10.5.2 Solution Strategy

Since the MPLP belongs to the NP-hard problems as well, a technique like *branch-and-bound* has to be applied for solving the MPLP to optimality. On the other hand, very often a time limit should not be exceeded, in particular in practical applications. Therefore, efficient heuristics are of importance.

The solution strategy proposed below is motivated by the following observation. If the MPLP is solved sequentially, i.e., successively a (near-)optimal pattern for a single pallet is computed with respect to the not yet placed items, then two difficulties arise.

- The computational effort to obtain an optimal pattern is considerable, in particular for larger assortments.
- The degree of volume utilization decreases from pattern to pattern because of the reduced available assortment.

In the b&b based approach proposed below, we try to avoid these drawbacks. Moreover, a possibility to regard a time limit is incorporated. Initially, a first feasible solution is constructed by fast heuristics. Then attempts are made to improve this solution provided the lower bound does not stop the computations and computer

time is still available. The solution strategy consists of a hierarchical structure. In the *MPLP algorithm* which searches for an optimal solution of the MPLP, various algorithms are applied to solve subproblems.

### 10.5.3 The MPLP Algorithm

The MPLP algorithm (Fig. 10.15) to (approximately) solve the MPLP works as follows: For the given instance  $E$  initially a lower and upper bound,  $\underline{k}$  and  $\bar{k}$ , respectively, for the optimal value  $k^* = k^*(E)$  are computed. ( $\bar{k}$  is the number of pallets of a feasible solution). Then, if  $\underline{k} < \bar{k}$ , the whole assortment is partitioned into  $k = \bar{k} - 1$  subassortments using the *Distributing Procedure* (Sect. 10.5.4) where some of the additional restrictions are regarded (weight and placement). For each subassortment (or for combinations of subassortments) a pattern is computed by the *Loading Procedure* (Sect. 10.5.5). If all items are packed, an improved solution is found. Otherwise, further attempts are made using an *Alternative Loading Procedure* (Sect. 10.5.6) to obtain a solution with  $k$  pallets.

The computation of the upper bound  $\bar{k}$  can be done, for example, using a greedy heuristic, and a lower bound  $\underline{k}$  can easily be obtained by means of the *volume bound*  $\lceil V / \sum_{i \in I} v_i b_i \rceil$ , or, better, the LP bound described in Sect. 11.5 for the *Multi Container Loading Problem*. Since the distribution problem itself is an NP-hard problem, the time limit has to be regarded as well during its solution in step (3). Further appropriate stopping rules can be used (cf. [21]).

Using the Distributing Procedure nearly equally-sized subassortments are generated with respect to volume and weight but without considering the geometrical form of the items. Therefore, in general, it is not guaranteed that a feasible pattern

#### MPLP algorithm

- (1) Compute a lower bound  $\underline{k}$  and an upper bound  $\bar{k}$  for the minimal number of pallets needed to pack the whole assortment. Set  $k := \bar{k}$ .
- (2) Set  $k := k - 1$ . If  $k < \underline{k}$ , then stop – an optimal solution is found.
- (3) Partition the whole assortment into  $k$  subassortments using the *Distributing Procedure*. If this is not possible, then stop.
- (4) Choose  $p$  ( $1 \leq p < k$ ) of the subassortments to define an instance  $E'$  of the (three-dimensional) PLP with several item types. Compute a sufficiently good packing pattern applying the *Loading Procedure*.  
Combine all not yet packed items of  $E'$  with a further subassortment to define a next instance of the PLP with several piece types. Compute a sufficiently good pattern with the *Loading Procedure*, etc.  
If all items are placed, then go to (2). Otherwise, apply an *Alternative Loading Procedure* to compute a solution with  $k$  pallets.

**Fig. 10.15** The MPLP algorithm

for a subassortment exists. Applying supplementary considerations, in particular for large item types (e.g. items with  $\ell_i > L/2$  or  $w_i > W/2$  or  $h_i > H/2$ ) the number of non-successful attempts can be reduced. A further opportunity to act against this problematic aspect consists in combining  $p > 1$  subassortments. In this way, the probability to quickly find a good pattern increases. Since the goal in step (4) is to find a solution with  $k$  patterns, one has an appropriate criterion to accept a pattern. If the (absolute) volume utilization is (e.g.) not smaller than  $1/k$ -times the total volume of the assortment, the pattern can be accepted.

#### 10.5.4 The Distributing Procedure

The *Distributing Procedure* (Fig. 10.16) realizes an algorithm to partition (distribute) a whole assortment in  $k$  subassortments. To this end, the volume, weight, and distribution conditions are considered. Moreover, to obtain a nearly uniform distribution of the heavy and the light items on the  $k$  pallets and to regard placement constraints, we initially partition index set  $I$  of item types into  $\bar{q}$  disjunctive subsets  $I_q, q = 1, \dots, \bar{q}$  (e.g.  $\bar{q} = 5$ ). Index set  $I_1$  represents the heavy piece types and those which have to be placed on the bottom of the pallet, whereas  $I_{\bar{q}}$  represents the light piece types and those to be packed on top of the pallet.

##### Distributing Procedure

- (1) Set  $a_{ij} := 0$  for all  $i$  and  $j$ ,  $q := 0$ .
- (2) Set  $q := q + 1$ . If  $q > \bar{q}$ , stop – a partition is found.  
Compute  $\bar{V}_q = \frac{1}{k} \sum_{r=1}^q \sum_{i \in I_r} v_i b_i$  and  $\Delta_q := \Delta_s \cdot (V - \bar{V}_q)$ . Set  $\bar{I}_q := I_q$ .
- (3) If  $\bar{I}_q = \emptyset$ , then go to (2).  
Determine  $i_0 \in \bar{I}_q$  with  $v_{i_0} b_{i_0} \geq v_i b_i$  for all  $i \in \bar{I}_q$ .
- (4) If there does not exist any pallet  $j$  with  $V_j + v_{i_0} b_{i_0} \leq \bar{V}_q + \Delta_q$ , then go to (5).  
If a pallet  $j$  with  $\Delta_q/2 \leq V_j + v_{i_0} b_{i_0} - \bar{V}_q \leq \Delta_q$  exists, then assign  $i_0$  to pallet  $j$ , i.e., set  $a_{i_0,j} := b_{i_0}$ ,  $\bar{I}_q := \bar{I}_q \setminus \{i_0\}$ , and go to (3). Otherwise, assign  $i_0$  to a pallet  $j_0$  with  $V_{j_0} \leq V_j$  for all  $j$ , i.e., set  $a_{i_0,j_0} := b_{i_0}$ ,  $\bar{I}_q := \bar{I}_q \setminus \{i_0\}$ , and go to (3).
- (5) Assign all items of type  $i_0$  to two pallets.  
More precisely: choose  $j_1, j_2$  and  $\alpha$  with  $\alpha < b_{i_0}$  such that  
 $\Delta_q/2 \leq V_{j_1} + v_{i_0} \alpha - \bar{V}_q \leq \Delta_q$  and  $V_{j_2} + v_{i_0} (b_{i_0} - \alpha) \leq \bar{V}_q + \Delta_q$  hold.  
If this is possible, then assign  $i_0$  to pallets  $j_1$  and  $j_2$ , i.e., set  $a_{i_0,j_1} := \alpha$ ,  $a_{i_0,j_2} := b_{i_0} - \alpha$ ,  $\bar{I}_q := \bar{I}_q \setminus \{i_0\}$ , and go to (3). If  $|\{r : |I_r| > 0\}| = 1$ , stop – no partition could be found. Otherwise, determine  $q'$  with  $I_{q'} \neq \emptyset$  such that  $|q - q'|$  is minimal, set  $I_q := I_q \cup I_{q'}$ ,  $I_{q'} := \emptyset$ ,  $a_{ij} := 0$  for  $i \in I_q$ ,  $\bar{I}_q := I_q$ ,  $q := q - 1$ , and go to (2).

**Fig. 10.16** Distributing Procedure for the MPLP algorithm

If the Distributing Procedure computes successfully  $k$  subassortments, then they can be described in form of an  $(m, k)$ -matrix  $A = (a_{ij})$ ,  $i \in I, j = 1, \dots, k$ , where  $a_{ij}$  indicates how many items of type  $i$  are assigned to pallet  $j$ . Parameter  $\Delta_s$  with  $\Delta_s \in (0, 1)$  is used to control the distribution of item types on the pallets. Initially, the items of  $I_1$  are assigned to the  $k$  pallets, then those of  $I_2$ , etc. At the same time, the assigned volume  $V_j := \sum_{i \in I} v_i a_{ij}$  should not exceed the average volume  $\bar{V}_q := (\sum_{r=1}^q \sum_{i \in I_r} v_i b_i)/k$  by more than  $\Delta_q := \Delta_s \cdot (V - \bar{V}_q)$  for  $q = 1, \dots, \bar{q}$ .

Note that in the case if the distribution of items of a certain type on two pallets (step (4)) is not possible, then this can be caused by an inappropriate definition of the index sets  $I_q$ ,  $q = 1, \dots, \bar{q}$ . In a similar way, the distribution of weight on the pallets can be controlled.

If the Distributing Procedure does not find a feasible distribution, then alternative attempts can be performed to find a partition into  $k$  subassortments, e.g. by means of *exchange heuristics*, or by a successive construction of patterns using the Loading Procedure.

### 10.5.5 The Loading Procedure

The described approach leads either to a problem to find a pattern which places all items of a subassortment on a single pallet ( $p = 1$ ), or to a problem to find a pattern with sufficiently high volume utilization ( $p > 1$ ). In this way, the computation of a (volume) optimal pattern is avoided.

To decide whether the volume utilization is sufficiently good, we use the parameter  $\Delta_v$  with  $\Delta_v \in (0, 1]$ . With  $V = LWH$  and  $\bar{V}_k := \sum_{i \in I} v_i b_i/k$  a pattern  $a \in \mathbb{Z}_+^m$  is said to be *sufficiently good* if the condition  $\sum_{i \in I} v_i a_i \geq \bar{V}_k + \Delta_v \cdot (V - \bar{V}_k)$  is fulfilled.

Furthermore, to construct a pattern for a single pallet, we use a restricted branching scheme with LIFO strategy. To this end, the number of subproblems is kept (very) small in each stage of the search process, i.e., an exhaustive search is avoided.

Moreover, we try to use complete layer patterns. If there exists a piece type  $i$  with sufficiently large number of items still to be packed, and if packing of layer patterns is possible, then such layer pattern is preferred (provided it has good area utilization). This approach is motivated by practical experience. Moreover, layer patterns can be computed in advance.

To find an appropriate layer pattern with identical items the G4 heuristic is applied (Sect. 10.2). If there does not exist any piece type with sufficiently large number of items to fill a layer, then the M4 heuristic (Sect. 10.4) is used which computes optimal patterns with 4-block structure and up to four different piece types. In addition, the M4 heuristic provides the opportunity to regard further demands such as, for example, lower and upper bounds on the number of items of a certain type to be placed within the pattern.

### Loading Procedure

- (1) Compute those piece types (depending on the current height of the already packed items) which can be placed next regarding the placement constraints. Choose packing strategy  $r = 1$ .
- (2) Compute  $\varrho$  sufficiently good layer patterns for strategy  $r$  ( $\varrho \leq \bar{\varrho}$ ) and sort them according to decreasing quality. If no suitable layer pattern could be found, then go to (4).
- (3) If all patterns for packing strategy  $r$  are already considered, then go to (4). Choose the next not yet considered layer pattern and place the items accordingly. If in this way, an improved pattern for the pallet is obtained, then save it. If the volume utilization is sufficiently good and no further items can be packed, then stop. Otherwise, if some items still have to be packed, then compute the new packing height and return to (1).
- (4) If necessary, reproduce the previous packing stage. If all packing strategies are already considered (i.e.  $r = 4$ ), then go to (4). Otherwise, increase  $r$  and go to (2).

**Fig. 10.17** Loading Procedure for the MPLP algorithm

A pattern with 4-block structure and more than one piece type leads, in general, to a non-flat surface of the packing due to the possibly different heights. A simple modification of the M4 heuristic allows the handling of such non-flat surfaces to compute further layer patterns which can be placed above.

In a similar way, piece types with smaller supply can be handled by another modification of the M4 heuristic, called M8 heuristic. In difference to the M4 heuristic, the *M8 heuristic* allows to place items of two types within any of the four blocks. To this end, a block is split into two rectangular subblocks each containing items of a single type only.

Aiming to obtain fast and simple structured patterns, four *packing strategies* (indexed by  $r$ ) are applied within the loading procedure (Fig. 10.17) in the given order:

- $r = 1$  layer-wise packing with identical items (G4 heuristic),
- $r = 2$  layer-wise packing of items of the same height or height combination, at most four piece types (M4 heuristic),
- $r = 3$  non-flat layer-wise packing of items with at most four types (M4 heuristic),
- $r = 4$  non-flat layer-wise packing of items with up to eight types (M8 heuristic).

Obviously, a lower bound  $\underline{k}$  as tight as possible is of high importance. The volume bound

$$\lceil \sum_{i=1}^m b_i \ell_i w_i h_i / (LWH) \rceil$$

provides, in general, only a first information. Stronger bounds will be considered in connection with the Container Loading Problem (Chap. 11).

### 10.5.6 Alternative Loading Procedure

In that case when not all items are packed in step (4) of the MPLP algorithm onto the desired  $k$  pallets, then several possibilities to try to pack the remained items exist.

An *Alternative Loading Procedure* performs attempts to place the not yet packed items on the  $k$  pallets. If this is not successful, then  $k'$  ( $k' < k$ ) sufficiently good (possibly improved) patterns are chosen. The items of the remaining  $k - k'$  pallets define a residual assortment for which a solution with  $k - k'$  patterns has to be found. The following *Alternative Loading Procedures* (ALP) are used in [21]:

**ALP 1** Let  $\bar{V}_k = \sum_{i \in I} v_i b_i / k$ . All patterns with volume utilization not smaller than  $\bar{V}_k + \Delta_1(V - \bar{V}_k)$  are accepted where  $V = LWH$ . The not yet packed items define the residual assortment. Parameter  $\Delta_1$  is appropriately taken from (0,1].

**ALP 2** Starting with the whole assortment, step (4) of the MPLP algorithm is repeated with an increased  $p$ -value and/or increased time limit.

### 10.5.7 Examples

To illustrate some aspects of the Loading Procedure, we consider

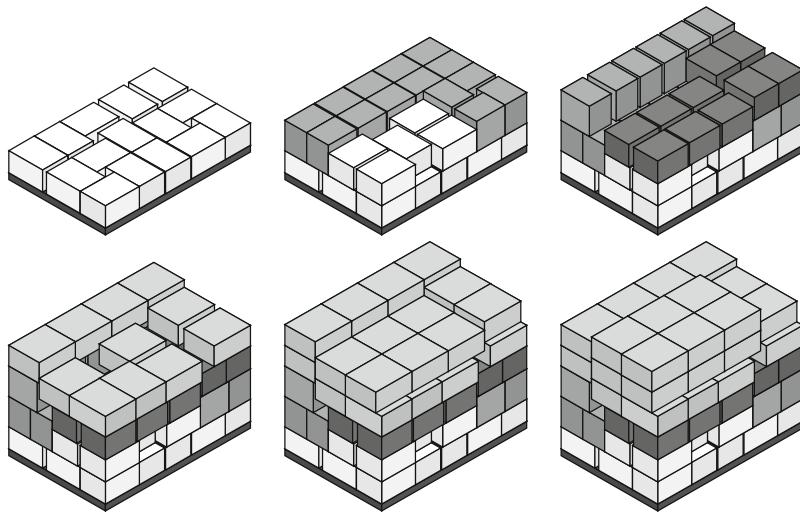
*Example 10.5* In Table 10.4 an assortment of items is given. The items have to be placed according to decreasing weights. 15 items of type 1 are packed within the first layer (Fig. 10.18). This pattern is computed by the G4 heuristic. The remaining five items of type 1 and as many items as possible (14) of type 2 are placed within the second layer. That pattern is obtained by the M4 heuristic.

The remaining six items of type 2 and ten items of type 3 are placed in the third layer. This pattern which is placed on the non-flat surface of the second layer, is obtained by an appropriate modification of the M4 heuristic. Due to some stability restrictions the fourth layer pattern results. For it, and the following two layers, the non-flat surface of the previous layer has to be taken into account. ■

In the next example a Multi Pallet Loading Problem is solved.

**Table 10.4** Input data of Example 10.5

|             | Length | Width | Height | Weight | Supply |
|-------------|--------|-------|--------|--------|--------|
| Item type 1 | 280    | 210   | 160    | 20.0   | 20     |
| Item type 2 | 235    | 180   | 220    | 15.0   | 20     |
| Item type 3 | 250    | 200   | 165    | 7.0    | 10     |
| Item type 4 | 300    | 250   | 150    | 5.5    | 30     |
| Pallet      | 1200   | 800   | 1050   |        |        |



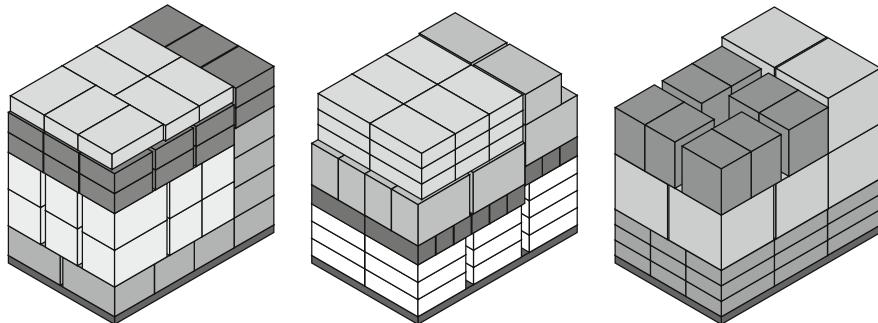
**Fig. 10.18** Layer-wise packing in Example 10.5

**Table 10.5** Input data of Example 10.6 and patterns obtained

| Type                      | Length<br>in [mm] | Width<br>in [mm] | Height<br>in [mm] | Weight<br>in [kg] | Supply | Pattern Vector |        |        |
|---------------------------|-------------------|------------------|-------------------|-------------------|--------|----------------|--------|--------|
|                           |                   |                  |                   |                   |        | Pal. 1         | Pal. 2 | Pal. 3 |
| 1                         | 400               | 266              | 100               | 6.5               | 27     | 0              | 0      | 27     |
| 2                         | 390               | 200              | 270               | 7.2               | 14     | 0              | 14     | 0      |
| 3                         | 390               | 255              | 100               | 3.2               | 25     | 7              | 18     | 0      |
| 4                         | 395               | 395              | 360               | 26.6              | 8      | 0              | 0      | 8      |
| 5                         | 395               | 242              | 240               | 13.5              | 14     | 14             | 0      | 0      |
| 6                         | 400               | 132              | 124               | 3.4               | 18     | 0              | 18     | 0      |
| 7                         | 360               | 399              | 112               | 11.5              | 20     | 0              | 20     | 0      |
| 8                         | 264               | 285              | 130               | 5.6               | 24     | 24             | 0      | 0      |
| 9                         | 390               | 300              | 194               | 14.0              | 14     | 14             | 0      | 0      |
| 10                        | 297               | 226              | 302               | 9.0               | 8      | 0              | 0      | 8      |
| Volume utilization in [%] |                   |                  |                   |                   |        | 93.6           | 90.6   | 89.2   |

*Example 10.6* The size parameters of the pallets are  $L = 1200$  and  $W = 800$ . The packing pattern must not exceed a height of  $H = 1050$ . Table 10.5 contains the input data of the assortment. The volume bound  $\lceil \sum_{i \in I} v_i b_i / V \rceil$  proves that the three patterns shown in Fig. 10.19 define an optimal solution of the MPLP.

Items of types 9, 5, 8, 3 are packed on pallet 1 (from bottom to top), those of types 7, 6, 2, and 3 on pallet 2 where some items of type 2 are rotated, and the items of types 1, 4, and 10 on the third pallet. ■



**Fig. 10.19** Patterns of Example 10.6

## 10.6 Exercises

**Exercise 10.1** Characterize the equivalence of two PLP instances by means of a system of linear inequalities.

**Exercise 10.2** Prove Proposition 10.2: A pattern  $A(I)$  has  $k$ -block structure with  $k \leq 3$  if and only if it has guillotine structure.

**Exercise 10.3** Show that  $k$ -block patterns with  $k \in \{5, 7, 9\}$  and diagonal-block patterns as defined in Figs. 10.3 and 10.4 possess the G4-structure.

**Exercise 10.4** Compute all efficient combinations for the PLP instance  $(40, 25, 7, 3)$  as well as an upper bound for the optimal value and show that it is not larger than 47.

**Exercise 10.5** Show that each  $\ell$ -pattern can be transformed into a  $w$ -pattern, and vice versa.

**Exercise 10.6** Show that the PLP instances  $(3750, 3063, 646, 375)$  and  $(40, 33, 7, 4)$  are equivalent and compute upper bounds for the optimal value.

## 10.7 Solutions

**To Exercise 10.1** We consider the two instances  $(L', W', \ell', w')$  and  $(L, W, \ell, w)$ . Instance  $(L, W, \ell, w)$  is equivalent to instance  $(L', W', \ell', w')$  if and only if the following inequality system is satisfied:

$$p\ell + qw \leq L, \quad p\ell + (q+1)w \geq L + 1, \quad (p, q) \in E(L', \ell', w'),$$

$$r\ell + sw \leq W, \quad r\ell + (s+1)w \geq W + 1, \quad (r, s) \in E(W', \ell', w'),$$

$$(\lfloor L'/\ell' \rfloor + 1)\ell \geq L + 1, \quad (\lfloor W'/\ell' \rfloor + 1)\ell \geq W + 1.$$

The last two conditions ensure that the, in general not efficient, partitions  $(\lfloor L'/\ell' \rfloor, 0)$  and  $(\lfloor W'/\ell' \rfloor, 0)$  are maintained as such ones.

**To Exercise 10.2** The proposition follows directly from the recursive definition of the guillotine structure and that of the 3-block structure.

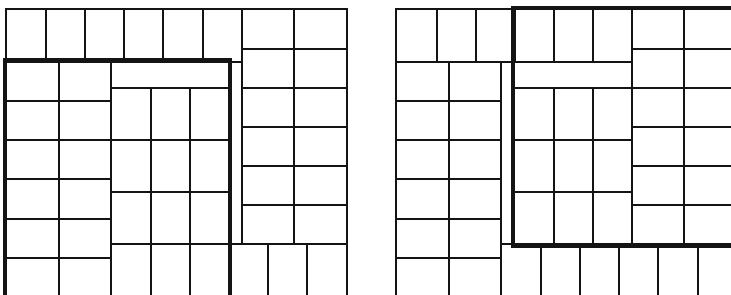
**To Exercise 10.3** Suppose, the 5-block pattern does not have guillotine structure. Then, vertically and horizontally oriented homogeneous patterns occur alternately in the four blocks along the border of the pallet. Moreover, within the central block either a vertically or a horizontally oriented homogeneous pattern is placed, too. But, this block can be combined with another whole and a part of a third block to form a larger non-homogeneous block as shown in Fig. 10.20.

In a similar way, four homogeneous blocks of a 7-block pattern can be pooled together (Fig. 10.21).

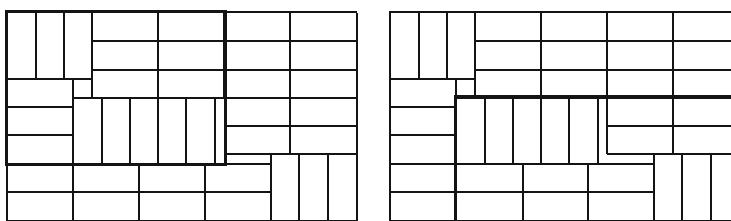
The considered 9-block patterns result from 7-block patterns by adding blocks on two sides. Therefore, using guillotine cuts, we can reduce the 9-block pattern to a 7-block pattern proving the G4-structure. The G4-structure of the diagonal-patterns is obvious.

**To Exercise 10.4** We obtain  $E(40, 7, 3) = \{(5, 1), (4, 4), (3, 6), (2, 8), (1, 11), (0, 13)\}$  and  $E(25, 7, 3) = \{(3, 1), (2, 3), (1, 6), (0, 8)\}$  as well as  $ub_a = ub_B = 47$ .

**To Exercise 10.5** We consider Fig. 10.9 of a partial block- $\ell$  pattern (page 292). If we complete the pattern, then finally horizontally or vertically oriented items are



**Fig. 10.20** G4-structure of a 5-block pattern



**Fig. 10.21** G4-structure of a 7-block pattern

placed on the upper right. Without loss of generality, we suppose that these items are horizontally oriented. If we now move all items to the right as far as possible (*right justified*), then we obtain a  $w$ -strip with  $\lfloor W/\ell \rfloor$  horizontally oriented items. In a recursive way, the reduced pallet of size  $(L-w) \times W$  can be managed analogously.

**To Exercise 10.6** We have

$$\begin{aligned} E(40, 7, 4) &= \{(5, 1), (4, 3), (3, 4), (2, 6), (1, 8), (0, 10)\} = E(3750, 646, 375), \\ E(33, 7, 4) &= \{(4, 1), (3, 3), (2, 4), (1, 6), (0, 8)\} = E(3063, 646, 375), \text{ and} \\ \lfloor 40/7 \rfloor &= 5 = \lfloor 3750/646 \rfloor. \end{aligned}$$

The equivalence of the both instances follows due to Theorem 10.1. Furthermore,  $ub_a = ub_B = 47$  holds.

## References

1. R. Alvarez-Valdes, F. Parreno, J.M. Tamarit, A tabu search algorithm for the pallet loading problem. *OR Spektrum* **27**, 43–61 (2005)
2. F.W. Barnes, Packing the maximum number of  $m \times n$  tiles in a large  $p \times q$  rectangle. *Discrete Math.* **26**, 93–100 (1979)
3. S. Bhattacharya, R. Roy, S. Bhattacharya, An exact depth-first algorithm for the pallet loading problem. *Europ. J. Oper. Res.* **110**(3), 610–625 (1998)
4. E.E. Bischoff, M.S.W. Ratcliff, Issues in the development of approaches to container loading. *OMEGA* **23**(4), 377–390 (1995)
5. E.E. Bischoff, M.S.W. Ratcliff, Loading multiple pallets. *J. Oper. Res. Soc.* **46**, 1322–1336 (1995)
6. K.A. Dowsland, The three-dimensional pallet chart: an analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems. *J. Oper. Res. Soc.* **35**(10), 895–905 (1984)
7. K.A. Dowsland, W.B. Dowsland, A comparative analysis of heuristics for the two-dimensional packing problem, in *Paper for EURO VI Conference* (1983)
8. H. Exeler, *Das homogene Packproblem in der betriebswirtschaftlichen Logistik*, Heidelberg (1988)
9. H. Exeler, Upper bounds for the homogenous case of a two-dimensional packing problem. *Z. Oper. Res.* **35**, 45–58 (1991)
10. T.J. Hodgson, A combined approach to the pallet loading problem. *IIE Trans.* **14**(3), 175–182 (1982)
11. H. Isermann, Ein Planungssystem zur Optimierung der Palettenbeladung mit kongruenten rechteckigen Versandgebinde. *OR Spektrum* **9**, 235–249 (1987)
12. A. Letchford, A. Amaral, Analysis of upper bounds for the pallet loading problem. *Eur. J. Oper. Res.* **132**(3), 582–593 (2001)
13. G.H.A. Martins, R.F. Dell, The minimum size instance of a Pallet Loading Problem equivalence class. *Eur. J. Oper. Res.* **179**(1), 17–26 (2007)
14. R. Morabito, S. Morales, A simple and effective recursive procedure for the manufacturer's pallet loading problem. *J. Oper. Res. Soc.* **49**, 819–828 (1998)
15. G. Naujoks, Neue Heuristiken und Strukturanalysen zum zweidimensionalen homogenen Packproblem. *OR Proc.* **1989**, 257–263 (1990)
16. J. Nelißen, *Neue Ansätze zur Lösung des Palettenbeladungsproblems*. Diss. RWTH Aachen (Verlag Shaker, Aachen, 1994)

17. G. Scheithauer, U. Sommerweiß, 4-block heuristic for the rectangle packing problem. *Eur. J. Oper. Res.* **108**(3), 509–526 (1998)
18. G. Scheithauer, J. Terno, The G4-heuristic for the pallet loading problem. *J. Oper. Res. Soc.* **47**, 511–522 (1996)
19. A.G. Tarnowski, Exact polynomial algorithm for special case of the two-dimensional cutting stock problem: a guillotine pallet loading problem. Report 9205, Belarussian State University (1992)
20. A.G. Tarnowski, J. Terno, G. Scheithauer, A polynomial time algorithm for the guillotine pallet loading problem. *INFOR* **32**, 275–287 (1994)
21. J. Terno, G. Scheithauer, U. Sommerweiß, J. Riehme, An efficient approach for the multi-pallet loading problem. *Eur. J. Oper. Res.* **123**(2), 372–381 (2000)

# Chapter 11

## Container Loading

Container loading and pallet loading problems are similar optimization problems. Their main difference consists in the number of different box types available. In this chapter, we describe an approach differing from that for pallet loading since now, in fact, all available boxes can have another shape. The proposed method is based on a contour concept. Moreover, we address the multi container loading and the (three-dimensional) strip packing problem as well. Furthermore, LP-based bounds are considered, too.

### 11.1 Problem Statements

The denotation *Container Loading Problem* (CLP) means, in general, the following three-dimensional packing problem: given a set of  $m$  different types of cuboids (boxes, pieces) with related availability and value coefficients and a box-shaped container, find a packing pattern of maximal total valuation.

More precisely, let a box (piece) of type  $i \in I := \{1, \dots, m\}$  be defined by its length  $\ell_i$ , width  $w_i$ , height  $h_i$ , supply  $u_i$ , and value  $g_i$ . Moreover, the container is characterized by its length  $L$ , width  $W$ , and height  $H$ . Then, the aim is to find a feasible pattern of a subset of the available pieces such that the total sum of values is maximized. As usual, a pattern is considered to be *feasible* if all packed pieces are placed completely within the container without overlapping each other. Possibly, depending on the real situation, some additional restrictions should be met. However, to be a guillotine pattern is not demanded, in general.

Supplementary requirements on a pattern, in particular resulting from practical purposes, can be rather complex, for example:

- if identical pieces are available, i.e.  $u_i > 1$  for some  $i \in I$ , then possibly, the number of pieces of a certain type  $i$  to be packed within the container can additionally be limited from below and above;
- the rotation of the boxes can be restricted;
- a weight limitation has to be regarded;
- stability and placement conditions can be present.

Naturally, numerous similarities between the CLP and the PLP exist. In principle, solution approaches can be transferred from one to the other problem. Differences result, among others, from the supplementary stability conditions. In the CLP, we have supporting walls which permit to pack in a less stable manner in comparison to the PLP. Another point is due to the loading technology. Loading a truck can require to load from back to front. Sometimes a packability from bottom to top (and not from the side) has to be guaranteed. In the following, we will regard only some of these conditions in our description.

Frequently, the name *Container Loading Problem* is also used for the related optimization problem to minimize the number of (identical) containers needed to stow all given boxes. A generalized problem with various types of containers could also be of interest. Then, an appropriate target function has to be used. This kind of loading problem is usually named as *Multi Container Loading Problem* (MCLP, Sect. 11.3), sometimes also *three-dimensional Bin Packing-Problem* (3BPP). Another interesting problem is the *three-dimensional Strip Packing Problem* (3SPP). Here, all boxes have to be packed into a container with given width and height such that the length needed is minimal (Sect. 11.4).

Within this book, we will not consider other important loading problems such as optimal loading onto ships or aircrafts not having box-shaped space to be filled, or packing of non-rectangular shaped objects. Instead, we refer exemplarily to [20, 29, 30].

## 11.2 The Container Loading Problem

For the classical CLP, i.e., to compute a single pattern with maximal value, we initially present an ILP model. Afterwards, we describe a basic algorithm for the CLP which applies the contour concept and, therefore, enables to incorporate supplementary conditions and to derive appropriate heuristics.

### 11.2.1 Integer Linear Programming Model

For a simpler description, we assume  $u_i = 1$  for all  $i \in I$  within this subsection. Moreover, each box is allowed to be arbitrarily orthogonally rotated. The following model for the CLP is based on thoughts of Fasano [12] and Padberg [27]. To uniform

in the presentation, the following size parameters are introduced: for the container let  $D_X$ ,  $D_Y$ , and  $D_Z$  denote its length, width, and height, i.e.,  $X$ ,  $Y$ , and  $Z$  represent the axes of a Cartesian coordinate system. (For example, in Sect. 11.1 we have  $D_X = L$ ,  $D_Y = W$ , and  $D_Z = H$ .) Moreover, the size parameters of a box  $i$  ( $i \in I$ ) are denoted by  $L_{1i}$ ,  $L_{2i}$ , and  $L_{3i}$ .

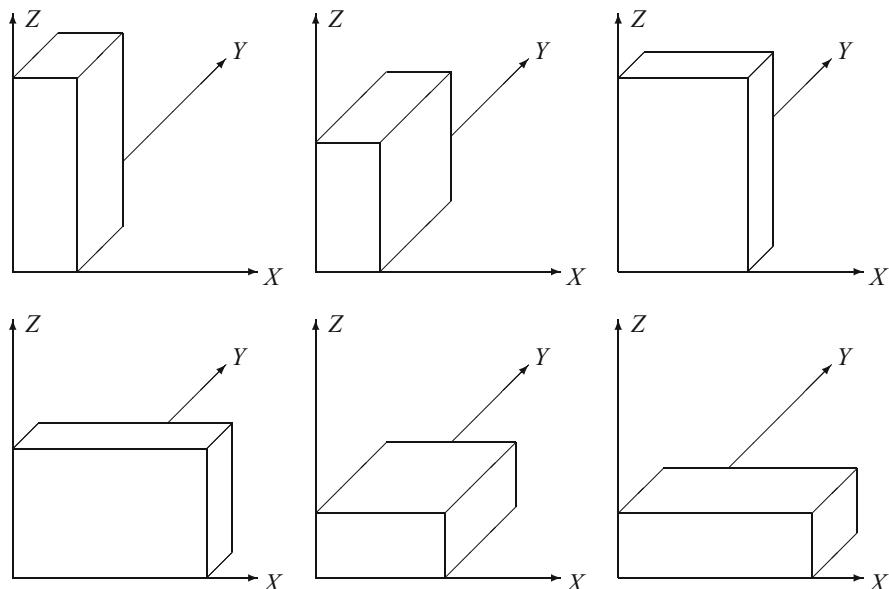
To characterize the orientation of a box  $i$  with respect to the container and the XYZ-coordinate system, we define a binary variable  $\delta_{ai}^A$  according to

$$\delta_{ai}^A := \begin{cases} 1, & \text{if box } i \text{ is placed with axis } a \text{ parallel to axis } A \text{ of the container,} \\ 0, & \text{otherwise,} \end{cases}$$

for each  $i \in I$ , for every axis  $a \in O_p := \{1, 2, 3\}$  of a piece, and for every axis  $A \in O_c := \{X, Y, Z\}$  of the container. The six different orientations are listed in Table 11.1 and depicted in Fig. 11.1.

**Table 11.1** Six possible orientations of a cuboid

|   | $\delta_{1i}^X$ | $\delta_{1i}^Y$ | $\delta_{1i}^Z$ | $\delta_{2i}^X$ | $\delta_{2i}^Y$ | $\delta_{2i}^Z$ | $\delta_{3i}^X$ | $\delta_{3i}^Y$ | $\delta_{3i}^Z$ |
|---|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 1 | 1               | 0               | 0               | 0               | 1               | 0               | 0               | 0               | 1               |
| 2 | 1               | 0               | 0               | 0               | 0               | 1               | 0               | 1               | 0               |
| 3 | 0               | 1               | 0               | 1               | 0               | 0               | 0               | 0               | 1               |
| 4 | 0               | 1               | 0               | 0               | 0               | 1               | 1               | 0               | 0               |
| 5 | 0               | 0               | 1               | 1               | 0               | 0               | 0               | 1               | 0               |
| 6 | 0               | 0               | 1               | 0               | 1               | 0               | 1               | 0               | 0               |



**Fig. 11.1** Six possible orientations of a box

Then, an orthogonal packing of piece  $i$  can be modeled by

$$\sum_{A \in O_c} \delta_{1i}^A \leq 1, \quad \sum_{A \in O_c} \delta_{2i}^A = \sum_{A \in O_c} \delta_{1i}^A, \quad \sum_{a \in O_p} \delta_{ai}^A = \sum_{A \in O_c} \delta_{1i}^A \text{ for all } A \in O_c. \quad (11.1)$$

In case of  $\sum_{A \in O_c} \delta_{1i}^A = 1$ , an edge of piece  $i$  is parallel to one edge of the container.

If  $\sum_{A \in O_c} \delta_{1i}^A = 0$ , then piece  $i$  is not packed. As *allocation point* we always choose the *center point* of the piece and denote it by  $(x_i^X, x_i^Y, x_i^Z)$ . The coordinates  $x_i^X, x_i^Y, x_i^Z$  are considered to be real variables which have to fulfill at least the containment and non-overlapping conditions. Using  $\ell_{ai} := \frac{1}{2}L_{ai}$ ,  $a \in O_p$ , the containment condition for piece  $i$  has the form

$$\sum_{a \in O_p} \ell_{ai} \delta_{ai}^A \leq x_i^A \leq \sum_{a \in O_p} (D_A - \ell_{ai}) \delta_{ai}^A, \quad A \in O_c. \quad (11.2)$$

To model the non-overlapping condition, we define additional binary variables: for each pair of two pieces  $i$  and  $j$  ( $i \neq j$ ) and each direction  $A \in O_c$  let

$$\lambda_{ij}^A = \begin{cases} 1, & \text{piece } i \text{ has to be placed 'before' piece } j \text{ in direction } A, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the non-overlapping condition for  $i, j \in I$ ,  $i < j$ , and  $A \in O_c$  can be described by the following restrictions:

$$\begin{aligned} D_A \lambda_{ji}^A + \sum_{a \in O_p} \ell_{ai} \delta_{ai}^A - \sum_{a \in O_p} (D_A - \ell_{aj}) \delta_{aj}^A &\leq x_i^A - x_j^A \\ &\leq \sum_{a \in O_p} (D_A - \ell_{ai}) \delta_{ai}^A - \sum_{a \in O_p} \ell_{aj} \delta_{aj}^A - D_A \lambda_{ij}^A, \end{aligned} \quad (11.3)$$

$$\sum_{A \in O_c} (\lambda_{ij}^A + \lambda_{ji}^A) \leq \sum_{A \in O_c} \delta_{1i}^A, \quad \sum_{A \in O_c} (\lambda_{ij}^A + \lambda_{ji}^A) \leq \sum_{A \in O_c} \delta_{1j}^A, \quad (11.4)$$

$$\sum_{A \in O_c} \delta_{1i}^A + \sum_{A \in O_c} \delta_{1j}^A \leq 1 + \sum_{A \in O_c} (\lambda_{ij}^A + \lambda_{ji}^A). \quad (11.5)$$

Condition (11.3) ensures a minimum distance between the allocation points, and restrictions (11.4) and (11.5) limit the  $\delta$ - and  $\lambda$ -variables to define a relevant combination. For a better understanding of (11.3), we note that (11.2) implies

$$\sum_{a \in O_p} \ell_{ai} \delta_{ai}^A - \sum_{a \in O_p} (D_A - \ell_{aj}) \delta_{aj}^A \leq x_i^A - x_j^A \leq \sum_{a \in O_p} (D_A - \ell_{ai}) \delta_{ai}^A - \sum_{a \in O_p} \ell_{aj} \delta_{aj}^A$$

(see Example 11.1). Clearly, the appropriate target function for the CLP, which has to be maximized, is as follows:

$$\sum_{i \in I} g_i \left( \sum_{A \in O_c} \delta_{1i}^A \right) \quad (11.6)$$

where  $g_i$  denotes the value coefficient (e.g. the volume) of piece  $i$ . Summarizing, we obtain an ILP model of the CLP defined by objective function (11.6) and restrictions (11.1)–(11.5) for all  $i, j \in I, i < j$ .

An important issue when solving the ILP model with a b&b algorithm consists in the performance (quality) of the *continuous relaxation* which essentially determines the number of subproblems to be investigated, and therefore, the total computational effort. Further investigations which are based on polyhedral theory, in particular for the CLP, can be found in [27]. We desist from going into those details and concentrate on the number of variables and restrictions. Given the total number of pieces  $m = |I|$ , then we have  $3m$  real variables  $x_i^A$  due to the allocation points,  $9m$  binary variables  $\delta_{ai}^A$  to represent the orientation of the pieces and their containment within the container, and  $3m(m - 1)$  binary variables  $\lambda_{ij}^A$  to formulate the non-overlapping conditions. The number of constraints increases quadratically with  $m$  as well.

Moreover, we have to take into account that  $m$  (in this subsection) represents the total number of available pieces, not necessarily the number of different types. Concerning the scenario of Sect. 11.1, we have  $\sum_{i \in I} u_i$  (different) pieces there.

Due of these aspects, the exact solution of the ILP model can become too time-consuming for CLP instances of practical interest. Another problematic issue remains still open: how to incorporate supplementary conditions such as stability. Therefore, we turn to another, more constructive approach, and in particular to heuristics, in the next subsection.

### 11.2.2 The Basic Algorithm

We consider again the CLP as defined in Sect. 11.1, i.e., a container

$$C := \{(x, y, z) \in \mathbb{R}^3 : 0 \leq x \leq L, 0 \leq y \leq W, 0 \leq z \leq H\}$$

and  $m = |I|$  piece types

$$B_i := \{(x, y, z) \in \mathbb{R}^3 : 0 \leq x \leq \ell_i, 0 \leq y \leq w_i, 0 \leq z \leq h_i\}, \quad i \in I,$$

are given, where  $u_i$  boxes of type  $i$  are available. Find a packing pattern, containing at most  $u_i$  pieces of type  $i$ , which has maximal total value. To simplify the

description, we assume that rotation of boxes is not permitted. In the model we use the origin as *reference point* for the pieces and the container, and  $B_i(p_i)$  represents the occupied point set

$$B_i(p_i) := \{(x, y, z) \in \mathbb{R}^3 : x_i \leq x \leq x_i + \ell_i, y_i \leq y \leq y_i + w_i, z_i \leq z \leq z_i + h_i\}$$

if a box of type  $i$  gets allocation point  $p_i = (x_i, y_i, z_i)$ . Obviously, piece  $B_i(p_i)$  is placed completely within the container if

$$0 \leq x_i, x_i + \ell_i \leq L, \quad 0 \leq y_i, y_i + w_i \leq W, \quad 0 \leq z_i, z_i + h_i \leq H$$

hold. The non-overlapping of two placed boxes  $B_i(p_i)$  and  $B_j(p_j)$  ( $i \neq j$ ) can be characterized shortly by  $B_i(p_i) \cap \text{int}(B_j(p_j)) = \emptyset$ . A (finite) sequence  $(\pi_1, p_1), \dots, (\pi_n, p_n)$  with  $n \leq \sum_{i=1}^m u_i$  represents a feasible three-dimensional packing pattern if

- $\pi_k \in I$  for all  $k \in \{1, \dots, n\}$ ,
- $\text{card}(\{k \in \{1, \dots, n\} : \pi_k = i\}) \leq u_i$  for all  $i \in I$ ,
- $B_{\pi_k}(p_k) \subseteq C$  for all  $k = 1, \dots, n$ ,
- $B_{\pi_k}(p_k) \cap \text{int}(B_{\pi_j}(p_j)) = \emptyset$  for  $1 \leq k < j \leq n$ .

For short, we describe a (packing) pattern by

$$\Pi_n := \bigcup_{k=1}^n B_{\pi_k}(p_k).$$

Now we are ready to formulate the principal approach of the basic algorithm. Starting with the empty container, we pack successively further pieces to already composed patterns if possible. Since the number of feasible patterns is, in general, huge (grows exponentially with  $m$ ), we limit the set of patterns which are accepted and used for further adding of pieces as follows. After each construction step at most  $p$  patterns are selected for further computations, and the other generated patterns are discarded. Hereby, the predefined parameter  $p$  has to be chosen carefully since it influences the total computational effort as well as the solution quality. This approach relates, in a certain sense, to the forward variant of *Dynamic Programming*. As criterion for selecting patterns, we apply function  $\beta$  which assigns a non-negative value to a pattern. An example for such a function  $\beta$  will be given below where a *good* pattern gets a larger value in comparison to a *less good* one.

We use the following notations within the basic algorithm:

- $k \dots$  iteration and stage counter,
- $P_k \dots$  set of basic patterns (states) in stage  $k$ ,
- $N_k \dots$  set of new generated patterns (states) in stage  $k$ ,
- $N_k(p) \dots$  set of  $p$  best patterns with respect to  $\beta$  in stage  $k$ ,
- $v^* \dots$  value of the currently best pattern  $\Pi^*$ .

### Basic algorithm for the Container Loading Problem

- $k := 0$ . Initialize  $P_1$  as empty container,  $v^* := 0$ .
- **repeat**  $k := k + 1, N_k := \emptyset$ .
  - For each pattern  $\Pi_j \in P_k$ , for each suitable allocation point  $p_k$ , and for each available piece type  $\pi_k = i$  construct, if possible, the packing pattern  $\Pi_j \cup \{(\pi_k, p_k)\}$  and add it to  $N_k$ .
  - Determine  $N_k(p)$ .
  - Renew  $v^*$  and  $\Pi^*$  if necessary.
  - Set  $P_{k+1} := N_k(p)$ .
- **until**  $P_{k+1} = \emptyset$ .

If some elimination of patterns takes place at least once (since  $|N_k| > p$ ), then the solution found by the basic algorithm is not optimal, in general. In that case, the algorithm constitutes a heuristic. Otherwise, an optimal pattern is found since all possible patterns are considered. Thereby the set of ‘suitable’ allocation points results from the (reduced) set of potential allocation points (see Exercise 11.2).

The basic algorithm as formulated above possesses some weak points which can be overcome by appropriate specification (e.g.. identical patterns can be obtained in several ways). On the other hand, the basic algorithm possesses good opportunities to incorporate supplementary conditions (e.g. stability) by the choice of proper allocation points. We will describe one of these opportunities by means of the contour concept in the next subsection.

### 11.2.3 The Contour Concept

The representation of patterns as the union of allocated pieces, as used in the previous subsection, has some drawbacks, for instance with respect of finding empty space or suitable allocation points to pack another box. For that reason, we use another representation of a pattern. Let

$$\Pi_n = \bigcup_{k=1}^n B_{\pi_k}(p_k) \quad \text{with } p_k = (x_k, y_k, z_k)$$

be a feasible pattern with  $n$  pieces. The projection of the packed pieces defines a partition of the base area  $[0, L] \times [0, W]$  of the container which can easily be extended to a rectangle partition. To each rectangle  $R_j$  of that partition, we assign a height  $e_j$  already occupied by the pattern. The rectangle partition together with

the height values defines the *contour* of the pattern. We describe a contour as follows:

$$K(\Pi_n) := \{(a_j, b_j, c_j, d_j, e_j) : j \in J\} \quad \text{with } a_j < c_j, b_j < d_j, j \in J,$$

where  $[a_j, c_j] \times [b_j, d_j]$ ,  $j \in J$ , represents a rectangle partition of the container base. Hence,  $(a_j, b_j)$  and  $(c_j, d_j)$  are opposite corner points of rectangle  $j$ . Set  $J$  denotes a related index set representing all rectangles of the partition. The *height*  $e_j$  associated with rectangle  $j$  is defined by

$$e_j := \max\{0, z_i + h_i : \exists k \in \{1, \dots, n\} \text{ with } i = \pi_k, (x_i, x_i + \ell_i) \cap (a_j, c_j) \neq \emptyset, (y_i, y_i + w_i) \cap (b_j, d_j) \neq \emptyset\}.$$

Due to the *contour concept* only those packing patterns lying above a previous contour are considered. Contour  $K := \{(R_j, e_j) : j \in J\}$  lies *above* contour  $\widetilde{K} := \{(\widetilde{R}_j, \widetilde{e}_j) : j \in \widetilde{J}\}$ , if  $e_j \geq \widetilde{e}_j$  holds for all  $(x, y) \in R_i \cap \widetilde{R}_j$ .

The packing of a piece above a contour leads to a new contour. Therefore, every pattern can be viewed as a sequence of contours where the successor contour results from the predecessor one by the allocation (packing) of a single piece. As easily can be seen, the representation of a pattern by a sequence of contours is not unique, in general.

Similar to two-dimensional packing problems, the consideration of *normalized* patterns is meaningful. The term *normalized* means in relation to the CLP that every packed piece is placed as near as possible to the origin (above the current contour) so that three of its surfaces are in contact with a wall of the container or another already packed box. As potential allocation points for a further piece only the points  $(a_j, b_j, e_j)$  are considered according to the contour concept. At this point, we assume that the rectangle partition has minimal cardinality to avoid the construction of non-normalized patterns. This assumption can be realized within an implementation of the approach by some additional tests, explained below, whether an allocation point  $(a_j, b_j, e_j)$  ( $j \in J$ ) would lead to a non-normalized pattern or not.

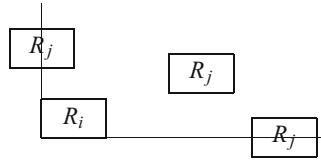
To save computational effort and, in particular, to generate normalized patterns, the structure of the current contour can be exploited. Let

$$\{R_j := (a_j, b_j, c_j, d_j) : j \in J\}$$

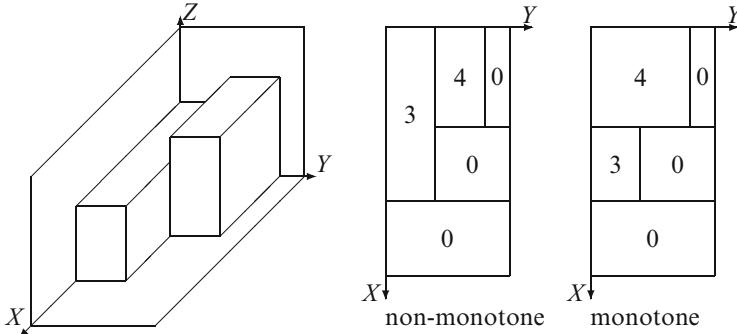
represent a partition of the container base  $[0, L] \times [0, W]$ . We define a relation  $<_R$  between the rectangles of the partition according to

$$R_i <_R R_j \quad :\Leftrightarrow \quad a_i < c_j, b_i < d_j,$$

which is illustrated in Fig. 11.2. The definition implies a partial *predecessor/successor* relation between two rectangles  $R_i$  and  $R_j$ . When placing a box  $k$  with allocation point  $(a_i, b_i, e_i)$ , i.e., directly on top of that box which induces rectangle  $R_i$ , then at most those rectangles  $R_j$  of the contour (besides  $R_i$ ) are changing for



**Fig. 11.2** Relation  $R_i <_R R_j$



**Fig. 11.3** Non-monotone and monotone contour of Example 11.1

which  $R_i <_R R_j$  holds. Moreover, the packing of box  $k$  at  $(a_i, b_i, e_i)$  is feasible only if  $e_j \leq e_i$  for all  $R_j$  with  $R_i <_R R_j$  and  $R_j \cap \{(x, y) : a_i < x < a_i + \ell_k, b_i < y < b_i + w_k\} \neq \emptyset$  hold.

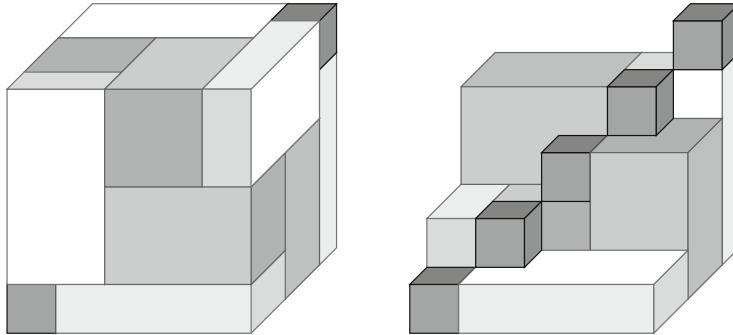
We call a contour  $\{(R_j, e_j) : j \in J\}$  *monotone*, if  $e_i \geq e_j$  holds for all  $R_i$  and  $R_j$  with  $R_i <_R R_j$  ( $i, j \in J$ ). In a straightforward way, we can assign a monotone contour  $K' = \{(R'_j, e'_j) : j \in J'\}$  with  $e'_j \geq e_j$  to a given contour  $K = \{(R_j, e_j) : j \in J\}$ . To this end, for each pair  $R_i = (a_i, b_i, c_i, d_i)$  and  $R_j = (a_j, b_j, c_j, d_j)$  of rectangles with  $R_i <_R R_j$  and  $e_i < e_j$  monotony is enforced as follows:

- if  $c_j \geq c_i$  and  $d_j \geq d_i$ , then set  $e'_i := e_j$ ;
- if  $c_i \leq a_j$  and  $d_j < d_i$ , then we partition  $R_i$  into two subrectangles  $R'_i$  and  $R''_i$  with  $R'_i := (a_i, b_i, c_i, d_j), R''_i := (a_i, d_j, c_i, d_i)$  and set  $e'_i := e_j, e''_i := e_i$ ;
- if  $d_i \leq b_j$  and  $c_j < c_i$ , then we partition  $R_i$  into two subrectangles  $R'_i$  and  $R''_i$  with  $R'_i := (a_i, b_i, c_j, d_i), R''_i := (c_j, b_i, c_i, d_i)$  and set  $e'_i := e_j, e''_i := e_i$ .

Applying repeatedly this procedure, we obtain a monotone contour.

*Example 11.1* Within a container of size  $L \times W \times H = 10 \times 5 \times 6$  two boxes  $B_1 = 7 \times 2 \times 3$  and  $B_2 = 4 \times 2 \times 4$  are placed with allocation point  $(0, 0, 0)$  and  $(0, 2, 0)$ , respectively. Figure 11.3 shows a corresponding non-monotone and a monotone contour. ■

The knowledge of a monotone contour immediately enables to pack a further box  $k$  with allocation point  $(a_i, b_i, e_i)$ , if feasible, i.e., if  $a_i + \ell_i \leq L, b_i + w_i \leq W$ , and  $e_i + h_i \leq H$  hold. Note that an algorithm which always generates a monotone



**Fig. 11.4** Counterexample for an exclusive application of monotone contours

contour after placing a next box, possibly cannot find an optimal pattern. To see this, we consider Example 11.2 (Fig. 11.4) which was kindly provided by J. Rietz.

*Example 11.2* Consider a cube of edge length 5. The task is to pack six boxes of size  $1 \times 2 \times 4$ , six of size  $2 \times 2 \times 3$ , and five unit cubes. Orthogonal rotation is permitted. A packing of all pieces is shown in Fig. 11.4. The partial packing pattern on the right side of Fig. 11.4 proves that forming a related monotone contour would enclose some unused volume (waste) so that an optimal pattern no longer can be constructed. ■

A possible alternative to use monotone contours consists in considering contours being monotone in a single direction, for instance the  $x$ -direction. A contour  $\{(a_j, b_j, c_j, d_j, e_j) : j \in J\}$  is said to be *monotone in  $x$ -direction*, if

$$e_j \leq e_i \quad \text{for all } j \in J \text{ with } b_j < d_i, \quad d_j > b_i$$

is fulfilled. Let us consider again the packing of a box  $k$  with allocation point  $(a_i, b_i, e_i)$ . We suppose that  $a_i + \ell_k \leq L$ ,  $b_i + w_k \leq W$ ,  $e_i + h_k \leq H$ , and  $b_i + w_k > d_i$  hold.

When using contours being monotone in  $x$ -direction, the test of feasibility of a packing is initially reduced to find that rectangle  $R_j$  which is uniquely determined by  $b_j = d_i$  and  $a_j \leq a_i < c_j$ . If  $e_j > e_i$ , then the packing is infeasible. Otherwise, if  $e_j \leq e_i$  and if  $b_i + w_k > d_j$ , then the nearest neighboring rectangle of  $R_j$  has to be examined, etc.

**Theorem 11.1** *For each pattern containing  $n$  pieces there exists a sequence  $K_0, K_1, \dots, K_n$  of contours each monotone in  $x$ -direction, and a sequence  $\pi_1, \dots, \pi_n$  of indices of the packed pieces such that contour  $K_j$  results from  $K_{j-1}$  by placing piece  $B_{\pi_j}$  above  $K_{j-1}$ .*

In difference to the two-dimensional packing problem where an optimal (packing) pattern can be obtained as a sequence of normalized patterns, it is still an open problem whether a similar assertion holds true for the three-dimensional case

as well. Note that removing the unit cubes in Example 11.2 does not provide a counterexample for this problem.

A sequential choice of potential allocation points enables in an easy way to construct a branching scheme. However, the multitude of patterns drastically limits the applicability of such an approach, even if appropriate bounds are available.

To define a non-negative rating function  $\beta$  for non-empty patterns  $\Pi_n$ , we formulate two demands which  $\beta$  should satisfy. Let  $f(\Pi_n)$  denote the sum of value coefficients of packed pieces and let  $v(\Pi_n)$  indicate the volume enclosed by the contour.

- If  $v(s) = v(t)$  and  $f(s) > f(t)$ , then  $\beta(s) > \beta(t) > 0$  should hold.
- If  $v(s) > v(t)$  and  $\frac{f(s)}{v(s)} = \frac{f(t)}{v(t)}$ , then  $\beta(s) > \beta(t) > 0$  should hold.

A simple set of rating functions is given by

$$\beta(s) = \frac{f(s) + \alpha}{v(s) + \gamma} \quad \text{with } \alpha \leq 0, \gamma > 0 \text{ and } f(s) + \alpha > 0 \quad \forall s \in S,$$

where  $S$  represents the set of all non-empty patterns (see Exercise 11.3).

Note that in order to handle practical requirements, we can restrict the set of possible allocation points. For example, in order to fill the container from back to front (the back wall is defined by  $x = 0$ ), the following rule can be applied:

- Only those allocation points  $(a_j, b_j, e_j)$  having minimal  $a_j$ -value are taken into account as allocation point of a further piece.

#### 11.2.4 Further Heuristics

The general approach to compute good patterns, described in the previous section, allows an adaptation to various particular scenarios. For instance, if all pieces of a certain type should be placed immediately adjacent or if a sufficiently large supporting base area has to be guaranteed, then in both cases a restricted choice of allocation points or box types to be placed next can be used.

If relatively few different piece types are given, then the CLP instance is said to be a *nearly* or *weakly homogeneous* packing problem. In case the number of piece types is rather large and the supply per type small, then the CLP instance represents a *strongly heterogeneous* problem [13].

The pallet loading problem (PLP), considered in Chap. 10, is closely related to the CLP but in the former problem stability aspects have more importance. A heuristic approach for the CLP, which is adapted from one for the PLP with several piece types, is presented in [4]. A comprehensive review of supplementary practical restrictions for the CLP can be found in [3].

The heuristics for solving the CLP can roughly be classified as follows. Most of them are so-called *wall building algorithms*, i.e., initially the first packed pieces are placed forming a *wall*, in fact considering a two-dimensional packing problem. Frequently, instead of wall the term *layer* is used. After filling the first layer, another layer pattern is constructed and placed in front of the first one, etc. Mostly, the layers are parallel to  $\{0\} \times [0, W] \times [0, H]$ .

Within the heuristic of George and Robinson [15] each layer is filled using horizontal strip patterns (in fact, one-dimensional). Thereby, the strip patterns are obtained by a greedy approach.

Further heuristics of wall building type are applied, among many others, in [2, 14]. An improved version of the algorithm of George and Robinson is proposed in [28]. The improvement results on the one hand by applying a restricted branching (several values of wall thickness are considered), and on the other hand, by considering horizontal and vertical strip patterns to fill a wall. Moreover, the strip patterns are obtained by solving knapsack problems.

A further class of heuristics collects these ones which initially form *towers* or *stacks* of maximum height  $H$  by putting pieces on top of each other. After that a two-dimensional packing problem is solved with respect to the container base area  $L \times W$ . Such an approach is, for instance, used in [16, 17]. The same construction principle is applied in the genetic algorithm presented in [13].

Within the *Tabu Search* algorithm in [6] the pieces are placed block-wise. In this way the stability of the generated pattern is assured by means of sufficiently large supporting area. A specific heuristic which regards the weight distribution of the packed pieces is proposed in [9].

Another heuristic which works similarly to a greedy heuristic can be found in [26]. There some decisions are randomized. Heuristics which exclusively generate guillotine patterns are considered, for instance, in [25].

### 11.3 Multi Container Loading and Three-Dimensional Bin Packing

Several optimization or decision problems are hidden behind the notion *Multi Container Loading Problem* (MCLP).

Considering the problem of minimizing material usage, we aim to find packing patterns for a single type of containers such that all boxes are stowed and the number of containers needed is minimal. This kind of problem is frequently named the *three-dimensional bin packing problem* (3BPP). The 3BPP is a natural extension of the 2BPP. Instead of rectangles and a rectangular bin, in the 3BPP boxes have to be packed into the minimal number of identical bins.

If various types of containers are available (possibly each with limited number), then the task consists in identifying a subset of the containers and corresponding packing patterns such that all pieces are packed and some target function is minimized.

For both problems it is assumed, in general, that sufficiently many containers are available to pack all boxes. In the opposite case, one has to accept that some boxes are not stowed. In the latter case, we have a problem to maximize volume utilization or to maximize an appropriate target function.

Due to the analogy to the Multi Pallet Loading Problem we abstain from a detailed description of solution approaches and refer to Sect. 10.5. However, a mathematical model of a relaxation of the MCLP with identical containers as well as a model for various container types are considered in Sect. 11.5 and in Exercise 11.5.

Note that the solution concept in [5], which is particularly designed for the MCLP, is similar to that described in Sect. 10.5 for the MPLP.

Due to the NP-hardness of the 3BPP, most of the related work is concerned with heuristic approaches. For instance, an application of *Tabu Search* metaheuristics on the 3BPP is described in [22]. Another heuristic for the 3BPP is proposed in [11] which combines algorithms for the 2SPP and 2BPP. The non-oriented case of the 3BPP is considered in [24]. Within the greedy heuristic proposed in [10] attention is concentrated, in particular, on the physical stability of patterns.

Note that some combinatorial lower bounds for the 3BPP are proposed in [8, 21]. Moreover, lower bounds and a branch-and-bound concept for the 3BPP are investigated in [23]. An integer linear programming approach is presented in [18].

## 11.4 Three-Dimensional Strip Packing

In difference to the CLP, in the *three-dimensional Strip Packing Problem* (3SPP), one of the size parameters of the container is variable and has to be minimized while packing all given boxes. Here we choose the length of the container to be variable.

Such a three-dimensional packing problem, as well as the related three-dimensional cutting problem, occurs in various fields. Within metallurgy, a frequently arising problem consists in optimally cutting cuboids from a larger block whose cross section is, in general, fixed but not its length. Similar problems can be found when cutting wood or cellular material. Applications arise as well in transport facilities.

A comparison of algorithms for solving the 3SPP can be found in [2]. More detailed concepts for developing adapted heuristics are discussed in [3]. Lower bounds and a branch-and-bound algorithm are presented in [23]. The b&b concept in [28] is used in [7] in two variants to solve the CLP. In the first variant a container of minimal length is searched directly, whereas in the second variant a sequence of CLP with decreasing container length is considered. The presented results of exhaustive tests prove the performance of both approaches.

## 11.5 LP Bounds

In this section we discuss upper bounds for the CLP as well as lower bounds for the MCLP.

### 11.5.1 *The Bar Relaxation of the CLP*

Besides the upper bound resulting from the LP relaxation of the ILP model (11.1)–(11.6) of Sect. 11.2 (page 321), further bounds are available.

We consider again a container with length  $L$ , width  $W$ , and height  $H$  having volume  $V = LWH$ . Furthermore, let  $m$  types of boxes (pieces) be given. In the following, we index the box types with  $t$  ( $t \in T$ ) where  $T := \{1, \dots, m\}$ . Moreover, we denote the volume of a piece of type  $t$  by  $v_t$  and its value coefficient with  $g_t$ ,  $t \in T$ . At most  $u_t$  pieces of type  $t$  are available to be packed. In case of maximizing volume utilization we have  $g_t = v_t$ . Moreover, we assume all input data to be integer.

Without loss of generality, we suppose that non-negative integer combinations of piece dimensions exist for the length  $L$ , the width  $W$ , and the height  $H$  such that no reduction of size parameters is possible.

Obviously, the maximal value of the CLP can be bounded above by the optimal value of the knapsack problem

$$ub_{KP} := \max \left\{ \sum_{t \in T} g_t x_t : \sum_{t \in T} v_t x_t \leq V, x_t \leq u_t, x_t \in \mathbb{Z}_+ \forall t \in T \right\}$$

where  $x_t$  indicates how many pieces of type  $t$  belong to the KP solution. Indeed,  $ub_{KP}$  provides an upper bound since the knapsack problem is a relaxation of the CLP which regards the box volume but not the geometrical form.

An improved bound can be obtained as follows: to regard different orientations of the boxes we introduce index sets

$$I_t := \{n_{t-1} + 1, \dots, n_t\}, \quad n_0 := 0, \quad t \in T,$$

in such a way that for each permitted orientation of piece type  $t$  exactly one index  $i \in I_t$  exists. Furthermore, let

$$I := \{1, \dots, n\} \quad \text{with } n := n_m.$$

We denote the length, width, and height of a piece  $i \in I$  with fixed orientation by  $\ell_i$ ,  $w_i$ , and  $h_i$ , respectively. Hence, piece  $i$  is of type  $t$  if  $i \in I_t$  holds.

To model different tasks, we additionally regard limitations how often pieces of a certain type have to be packed or are allowed to be packed. The set  $T^{\equiv} \subseteq T$  represents those piece types  $t$  for which *exactly*  $u_t$  pieces have to be placed. Furthermore, we denote the index sets of piece types where *at least*  $\underline{u}_t$ ,  $t \in T^{\geq}$ , and *at most*  $\bar{u}_t$ ,  $t \in T^{\leq}$ , pieces have to be packed by  $T^{\geq}$  and  $T^{\leq}$ , respectively. In particular, we do not assume that  $T^{\leq} \cap T^{\geq} = \emptyset$  holds. Note that in case of a too large choice of  $u_t$ ,  $t \in T^{\equiv}$ , or of  $\underline{u}_t$ ,  $t \in T^{\geq}$ , the solvability can be lost.

To keep the following linear programming problem as small as possible, we assume that

$$T^{\equiv} \cap (T^{\geq} \cup T^{\leq}) = \emptyset \quad \text{and} \quad \underline{u}_t < \bar{u}_t \text{ for } t \in T^{\geq} \cap T^{\leq}$$

hold. Moreover, we define

$$T^o := T \setminus (T^{\equiv} \cup T^{\geq} \cup T^{\leq})$$

and  $t_i$ ,  $i = 1, \dots, 4$  to be the cardinality of  $T^{\equiv}$ ,  $T^{\geq}$ ,  $T^{\leq}$ , and  $T^o$ , respectively. Without loss of generality, we can renumber the piece types such that

$$T^{\equiv} = \{1, \dots, t_1\}, \quad T^{\geq} = \{t_1 + 1, \dots, t_1 + t_2\}, \quad T^{\leq} = \{t_0 + 1, \dots, t_0 + t_3\}$$

hold where  $t_0 := m - t_4 - t_3$ . Hence, we have  $T^{\geq} \cap T^{\leq} = \{t_0 + 1, \dots, t_1 + t_2\}$ .

The approach presented below is a generalization of a relaxation applied in [19] for the (two-dimensional) *Pallet Loading Problem*. The basic idea of the bar relaxation uses a partition of the container

$$C = \{(x, y, z) : 0 \leq x \leq L, 0 \leq y \leq W, 0 \leq z \leq H\}$$

in  $W \cdot H$  bars or sticks of length  $L$  and cross section  $1 \times 1$ , i.e.,

$$C = \bigcup_{j=1}^W \bigcup_{k=1}^H \{(x, y, z) : 0 \leq x \leq L, j-1 \leq y \leq j, k-1 \leq z \leq k\}.$$

These bars are called *L-bars*. A corresponding schema is already shown in Fig. 5.2 on page 135. Considering a (three-dimensional) packing pattern of  $C$ , we obtain, due to the partition, a set of  $W \cdot H$  *L-patterns* in length direction. In a similar way, we define *W-bars* and *H-bars*. Because of the  $1 \times 1$  cross section a patterns of a bar can be seen as a *one-dimensional* one. If we neglect the real position of the bar patterns, then we obtain a relaxation of the CLP.

To formulate a mathematical model precisely, let  $A = (A_1, \dots, A_n)^T \in \mathbb{Z}_+^n$  denote the *packing vector* of a feasible (three-dimensional) packing pattern. Entry  $A_i$  of  $A$  indicates how many pieces with fixed orientation  $i$  are packed.

Each  $L$ -pattern can be characterized by a vector  $a^p = (a_{1p}, \dots, a_{np})^\top \in \mathbb{Z}_+^n$  where index  $p$  differentiates the  $L$ -patterns. Thus,  $a_{ip}$  counts how many parts of piece type  $i$  (having volume  $\ell_i$ ) are contained in the  $p$ th  $L$ -pattern. Notice, an  $L$ -pattern  $a^p$  is feasible only if

$$\sum_{i \in I} \ell_i a_{ip} \leq L, \quad \sum_{i \in I_t} a_{ip} \leq u_t, \quad t \in T^=, \quad \sum_{i \in I_t} a_{ip} \leq \bar{u}_t, \quad t \in T^{\leq}. \quad (11.7)$$

The first condition ensures that the container length  $L$  is not exceeded. The two other inequalities result from the maximum number pieces of a certain type can be packed regarding all the permitted orientations. Let  $P$  denote the set of all feasible  $L$ -patterns.

In the same way, we define  $W$ -patterns  $b^q = (b_{1q}, \dots, b_{nq})^\top$ ,  $q \in Q$ , and  $H$ -patterns  $c^r = (c_{1r}, \dots, c_{nr})^\top$ ,  $r \in R$ .

*Example 11.3* Given a container of size  $L = 25$ ,  $W = 10$ ,  $H = 15$ , and two (non-rotatable) piece types of size  $\ell_1 = 10$ ,  $w_1 = 10$ ,  $h_1 = 7$ , and  $\ell_2 = 15$ ,  $w_2 = 7$ ,  $h_2 = 5$ . Consider the packing vector  $A = (2, 3)^\top$  assigned to a packing pattern which is uniquely determined by the allocation points  $(0, 0, 0)$  and  $(0, 0, 7)$  for the two pieces of type 1, and  $(10, 0, 0)$ ,  $(10, 0, 5)$ ,  $(10, 0, 10)$  for the three boxes of type 2 (Fig. 11.5). Then we obtain the following bar patterns:

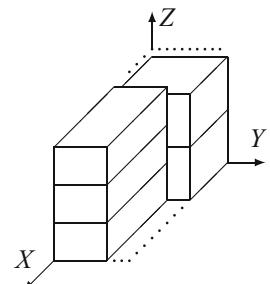
- $L$ -patterns:  $a^1 = (1, 1)^\top$  (98 times),  $a^2 = (1, 0)^\top$  (42),  $a^3 = (0, 1)^\top$  (7),
- $W$ -patterns:  $b^1 = (1, 0)^\top$  (140),  $b^2 = (0, 1)^\top$  (225),
- $H$ -patterns:  $c^1 = (2, 0)^\top$  (100),  $c^2 = (0, 3)^\top$  (105).

We define integer variables  $x_p$ ,  $y_q$ , and  $z_r$  assigned to the  $L$ -,  $W$ -, and  $H$ -patterns which indicate how often the related pattern is used. Then the next equality is obvious:

$$A_i = \frac{1}{w_i h_i} \sum_{p \in P} a_{ip} x_p = \frac{1}{\ell_i h_i} \sum_{q \in Q} b_{iq} y_q = \frac{1}{\ell_i w_i} \sum_{r \in R} c_{ir} z_r. \quad (11.8)$$

To finally obtain the bar relaxation for the CLP, we consider the following

**Fig. 11.5** Packing pattern of Example 11.3



### Basic model of the CLP

$$\eta = \max \sum_{t \in T} g_t \sum_{i \in I_t} A_i \quad \text{s.t.} \quad (11.9a)$$

$A = (A_1, \dots, A_n)^\top \in \mathbb{Z}_+^n$  represents a feasible 3D packing pattern, (11.9b)

$$\sum_{i \in I_t} A_i = u_t, \quad t \in T^=, \quad \sum_{i \in I_t} A_i \geq \underline{u}_t, \quad t \in T^{\geq}, \quad \sum_{i \in I_t} A_i \leq \bar{u}_t, \quad t \in T^{\leq}. \quad (11.9c)$$

If we replace the packing vector  $A$  according to (11.8) and concretize condition (11.9b), then we obtain the model of the *bar relaxation* for the CLP where  $\gamma_t := g_t/v_t$ ,  $t \in T$ :

### Bar relaxation of the CLP

$$\eta = \max \sum_{p \in P} \sum_{t \in T} \gamma_t \sum_{i \in I_t} \ell_i a_{ip} x_p \quad \text{s.t.} \quad (11.10a)$$

$$\sum_{p \in P} \ell_i a_{ip} x_p - \sum_{q \in Q} w_i b_{iq} y_q = 0, \quad i \in I, \quad (11.10b)$$

$$\sum_{p \in P} \ell_i a_{ip} x_p - \sum_{r \in R} h_i c_{ir} z_r = 0, \quad i \in I, \quad (11.10c)$$

$$\sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p = u_t v_t, \quad t \in T^=, \quad (11.10d)$$

$$\sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \geq \underline{u}_t v_t, \quad t \in T^{\geq}, \quad (11.10e)$$

$$\sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \leq \bar{u}_t v_t, \quad t \in T^{\leq}, \quad (11.10f)$$

$$\sum_{p \in P} x_p \leq WH, \quad \sum_{q \in Q} y_q \leq LH, \quad \sum_{r \in R} z_r \leq LW, \quad (11.10g)$$

$$x_p \geq 0, \quad p \in P, \quad y_q \geq 0, \quad q \in Q, \quad z_r \geq 0, \quad r \in R. \quad (11.10h)$$

According to the objective function (11.10a) the total value of the packed pieces is maximized. Restrictions (11.10d)–(11.10f) ensure that the limitations are met how often pieces of a certain type may or have to be packed. Condition (11.9b), which is formulated only verbally, is replaced by restrictions (11.10b), (11.10c), and (11.10g). Obviously, (11.10g) guarantees that not too many  $L$ -,  $W$ -, and  $H$ -patterns are used. Finally, conditions (11.10b) and (11.10c) ensure that for each orientation  $i$  of piece type  $t$  the same volume in  $L$ -,  $W$ -, and  $H$ -direction is packed (cf. Sect. 5.3). In difference to (11.9b) the feasibility of an  $L$ -,  $W$ -, or  $H$ -pattern can easily be defined by linear restrictions as in (11.7).

Since the real position of the bar patterns and the integer condition are neglected, the resulting linear programming problem (11.10) constitutes a relaxation for the CLP.

The above LP model of the bar relaxation possesses, in general, a huge number of variables. For that reason we apply the column generation technique (Sect. 4.3.3) to solve the LP problem. For short, let

$$n_1 := n, \quad n_2 := 2n_1, \quad n_3 := n_2 + t_1, \quad n_4 := n_3 + t_2, \quad n_5 := n_4 + t_3, \quad n_6 := n_5 + 3 \quad (11.11)$$

denote the cumulative number of restrictions in model (11.10). We denote the vector of simplex-multipliers of an optimal basis solution of a restricted master problem, defined by a subset of columns, of model (11.10), by  $d = (d_1, \dots, d_{n_6})^\top$ —Applying *column generation*, we have to solve at most three knapsack problems to verify that the current solution is optimal for the whole problem as well, or to generate at least one *column* to be added to the column pool. Observe, although at least  $2n + 3$  restrictions are present in model (11.10), the number of variables within the column generation problems is always equal to  $n$ . The resulting slave problems are specific knapsack-type problems. For the  $L$ -direction we obtain:

$$\begin{aligned} \chi_L^* &= \max \sum_{t \in T} \gamma_t \sum_{i \in I_t} \ell_i a_i + \sum_{i \in I} d_i \ell_i a_i + \sum_{i \in I} d_{n_1+i} \ell_i a_i + \sum_{t \in T^=} d_{n_2+t} \sum_{i \in I_t} \ell_i a_i \\ &\quad + \sum_{t \in T^{\geq}} d_{n_2+t} \sum_{i \in I_t} \ell_i a_i + \sum_{t \in T^{\leq}} d_{n_4+t-t_0} \sum_{i \in I_t} \ell_i a_i + d_{n_5+1} \\ &= \max \sum_{t \in T^= \cup (T^{\geq} \setminus T^{\leq})} \sum_{i \in I_t} (\gamma_t + d_i + d_{n_1+i} + d_{n_2+t}) \ell_i a_i \\ &\quad + \sum_{t \in T^{\geq} \cap T^{\leq}} \sum_{i \in I_t} (\gamma_t + d_i + d_{n_1+i} + d_{n_2+t} + d_{n_4+t-t_0}) \ell_i a_i \\ &\quad + \sum_{t \in T^{\leq} \setminus (T^{\geq} \setminus T^{\leq})} \sum_{i \in I_t} (\gamma_t + d_i + d_{n_1+i} + d_{n_4+t-t_0}) \ell_i a_i \\ &\quad + \sum_{t \in T^o} \sum_{i \in I_t} (\gamma_t + d_i + d_{n_1+i}) \ell_i a_i + d_{n_5+1} \end{aligned}$$

$$\text{s.t. } \sum_{i \in I} \ell_i a_i \leq L, \quad \sum_{i \in I_t} a_i \leq u_t, \quad t \in T^=, \quad \sum_{i \in I_t} a_i \leq \bar{u}_t, \quad t \in T^{\leq}, \quad a_i \in \mathbb{Z}_+, \quad i \in I.$$

The slave problems for  $W$ - and  $H$ -patterns possess the form

$$\begin{aligned} \chi_W^* &= \max - \sum_{t \in T} \sum_{i \in I_t} d_i w_i b_i + d_{n_5+2} \\ \text{s.t. } & \sum_{i \in I} w_i b_i \leq W, \quad \sum_{i \in I_t} b_i \leq u_t, \quad t \in T^=, \quad \sum_{i \in I_t} b_i \leq \bar{u}_t, \quad t \in T^{\leq}, \quad b_i \in \mathbb{Z}_+, \quad i \in I, \end{aligned}$$

and

$$\begin{aligned} \chi_H^* &= \max - \sum_{t \in T} \sum_{i \in I_t} d_{n+i} h_i c_i + d_{n_5+3} \\ \text{s.t. } & \sum_{i \in I} h_i c_i \leq H, \quad \sum_{i \in I_t} c_i \leq u_t, \quad t \in T^=, \quad \sum_{i \in I_t} c_i \leq \bar{u}_t, \quad t \in T^\leq, \quad c_i \in \mathbb{Z}_+, \quad i \in I, \end{aligned}$$

respectively. Hereby, the integer variables  $a_i, b_i, c_i$  model how often piece  $i$  is contained in the respective pattern ( $i \in \{1, \dots, n\}$ ). If

$$\min \left\{ -\chi_L^*, -\chi_W^*, -\chi_H^*, \min_{t \in T^=} d_{n_2+t}, \min_{t \in T^\leq} d_{n_4+t-t_0}, \min_{i=1,2,3} d_{n_5+i} \right\} \geq 0 \quad (11.12)$$

holds, then the current basis is optimal. It is evident,  $\chi_L^*$ ,  $\chi_W^*$ , and  $\chi_H^*$  are the optimal values of the slave problems whereas the other terms belong to columns of the slack variables. In case condition (11.12) is not fulfilled, at least a new column with negative transformed (reduced) objective coefficient has been found which enters into the improved basis solution. Possibly, several new columns can be inserted into the column pool.

In case of  $T^= \cup T^\geq \neq \emptyset$ , a first feasible basis solution of (11.10) can be obtained by solving an auxiliary problem. To this end, *artificial variables*  $\sigma_i$ ,  $i = 1, \dots, n_4$ , (also known as *auxiliary variables*) are added to the model:

$$\begin{aligned} \eta_0 &= \min - \sum_{p \in P} \left[ 3 \sum_{t \in T^= \cup T^\geq} \sum_{i \in I_t} \ell_i a_{ip} + 2 \sum_{t \in T \setminus (T^= \cup T^\geq)} \sum_{i \in I_t} \ell_i a_{ip} \right] x_p \\ &\quad + \sum_{i \in I} \left[ \sum_{q \in Q} w_i b_{iq} y_q + \sum_{r \in R} h_i c_{ir} z_r \right] + \sum_{t \in T^=} u_t v_t + \sum_{t \in T^\geq} \underline{u}_t v_t \quad \text{s.t.} \end{aligned} \quad (11.13a)$$

$$\sigma_i + \sum_{p \in P} \ell_i a_{ip} x_p - \sum_{q \in Q} w_i b_{iq} y_q = 0, \quad i \in I, \quad (11.13b)$$

$$\sigma_{n_2+t} + \sum_{p \in P} \ell_i a_{ip} x_p - \sum_{r \in R} h_i c_{ir} z_r = 0, \quad i \in I, \quad (11.13c)$$

$$\sigma_{n_2+t} + \sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p = u_t v_t, \quad t \in T^=, \quad (11.13d)$$

$$\sigma_{n_2+t} + \sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \geq \underline{u}_t v_t, \quad t \in T^\geq, \quad (11.13e)$$

$$\sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \leq \bar{u}_t v_t, \quad t \in T^{\leq}, \quad (11.13f)$$

$$\sum_{p \in P} x_p \leq WH, \quad \sum_{q \in Q} y_q \leq LH, \quad \sum_{r \in R} z_r \leq LW, \quad (11.13g)$$

$$\sigma_i \geq 0 \quad \forall i, \quad x_p \geq 0, \quad p \in P, \quad y_q \geq 0, \quad q \in Q, \quad z_r \geq 0, \quad r \in R. \quad (11.13h)$$

According to this method of auxiliary variables (frequently called *phase 1* of the simplex method) the objective function is simply the sum of all additional variables. A feasible basis solution of model (11.10) is found if  $\eta_0^* = 0$  holds and all  $\sigma_i$ -variables became non-basis variables. This problem of finding a first feasible solution can be solved via column generation as well. A first feasible basis solution of model (11.13) is immediately given by the  $\sigma_i$ -variables and the slack variables belonging to (11.13f)–(11.13g).

### 11.5.2 The Bar Relaxation of the MCLP

We consider here a Multi Container Loading Problem where  $u_t$  pieces of box type  $t \in T = \{1, \dots, m\}$  have to be placed within a minimal number of identical containers. The container dimensions are denoted by  $L$ ,  $W$ , and  $H$ . Moreover, we describe all possible (feasible) orientations as in the previous subsection by means of index set  $I = \{1, \dots, n\}$ . The length, width, and height of the (non-rotatable) pieces  $i \in I$  are again denoted by  $\ell_i$ ,  $w_i$ , and  $h_i$ , respectively.

Let index set  $J$  represent the set of all feasible three-dimensional packing patterns  $A^j \in \mathbb{Z}_+^n$ ,  $j \in J$ , for the given container type. Furthermore, we denote the frequency how many containers are filled according to pattern  $j$  by  $\mu_j$ . In this way, we obtain an ILP model of the MCLP:

#### Model of the MCLP

$$\eta = \min \sum_{j \in J} \mu_j \quad \text{s.t.} \quad (11.14a)$$

$$\sum_{j \in J} \sum_{i \in I_t} A_{ij} \mu_j \geq u_t, \quad t \in T, \quad (11.14b)$$

$$\mu_j \geq 0, \quad \mu_j \in \mathbb{Z}_+, \quad j \in J. \quad (11.14c)$$

Using  $\xi_{pj}$ ,  $\psi_{qj}$ , and  $\zeta_{rj}$  to denote the frequency an  $L$ -,  $W$ - or  $H$ -pattern belongs to packing pattern  $A^j$ , respectively, and substituting  $A_{ij}$  in inequality (11.14b) in

analogy to (11.8), we obtain

$$\sum_{j \in J} \sum_{i \in I_t} \frac{1}{w_i h_i} \sum_{p \in P} a_{ip} \xi_{pj} \mu_j = \sum_{p \in P} \sum_{i \in I_t} \frac{1}{w_i h_i} a_{ip} \sum_{j \in J} \xi_{pj} \mu_j \geq u_t, \quad t \in T.$$

Next, we define

$$x_p := \sum_{j \in J} \xi_{pj} \mu_j, \quad y_q := \sum_{j \in J} \psi_{qj} \mu_j, \quad z_r := \sum_{j \in J} \zeta_{rj} \mu_j$$

to count the total number of  $L$ -,  $W$ -, or  $H$ -patterns in a solution. Transforming the objective function appropriately, we obtain the following model of the bar relaxation for the MCLP:

### Bar relaxation of the MCLP

$$\eta = \min \kappa \quad \text{s.t.} \tag{11.15a}$$

$$\sum_{p \in P} \ell_i a_{ip} x_p - \sum_{q \in Q} w_i b_{iq} y_q = 0, \quad i \in I, \tag{11.15b}$$

$$\sum_{p \in P} \ell_i a_{ip} x_p - \sum_{r \in R} h_i c_{ir} z_r = 0, \quad i \in I, \tag{11.15c}$$

$$\sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \geq u_t v_t, \quad t \in T, \tag{11.15d}$$

$$\sum_{p \in P} x_p - W H \kappa \leq 0, \quad \sum_{q \in Q} y_q - L H \kappa \leq 0, \quad \sum_{r \in R} z_r - L W \kappa \leq 0, \tag{11.15e}$$

$$x_p \geq 0, \quad p \in P, \quad y_q \geq 0, \quad q \in Q, \quad z_r \geq 0, \quad r \in R. \tag{11.15f}$$

Obviously, objective function (11.15a) counts the total number  $\kappa$  of containers needed. Conditions (11.15e) ensure that all used  $L$ -,  $W$ -, and  $H$ -patterns fit within the total volume. Restriction (11.15b) enforces that for each orientation  $i$  of piece type  $t$  the same volume in  $L$ - and  $W$ -direction is occupied. Analogously, Eq. (11.15c) guarantees the same volume utilization for each orientation  $i$  of piece type  $t$  in  $L$ - and  $H$ -direction. Inequality (11.15d) realizes the order demands.

It is obvious that model (11.15) constitutes a relaxation of the MCLP. It can be solved using the column generation technique. To this end, we use again the abbreviations defined in (11.11). Because of  $T = T^{\geq}$ , now we have  $n_3 := n_2$ ,  $n_5 := n_4$ .

Let again  $d = (d_1, \dots, d_{n_6})^\top$  denote the simplex-multipliers of an optimal basis solution of a restricted master problem of (11.15). Applying the column generation technique, (Sect. 4.3.3), we have to solve the following knapsack problems to decide whether the current restricted master problem contains an optimal solution of the

whole problem, or not. The slave problem for the  $L$ -patterns is as follows:

$$\begin{aligned}\chi_L^* = \max & \sum_{t \in T} \sum_{i \in I_t} (d_i + d_{n_1+i} + d_{n_2+t}) \ell_i a_i + d_{n_5+1} \\ \text{s.t. } & \sum_{i \in I} \ell_i a_i \leq L, \quad a_i \in \mathbb{Z}_+, \quad i \in I.\end{aligned}$$

For the  $W$ - and  $H$ -patterns the subsequent knapsack problems have to be solved:

$$\chi_W^* = \max d_{n_5+2} - \sum_{t \in T} \sum_{i \in I_t} d_i w_i b_i \quad \text{s.t.} \quad \sum_{i \in I} w_i b_i \leq W, \quad b_i \in \mathbb{Z}_+, \quad i \in I,$$

and

$$\chi_H^* = \max d_{n_5+3} - \sum_{t \in T} \sum_{i \in I_t} d_{n_1+i} h_i c_i \quad \text{s.t.} \quad \sum_{i \in I} h_i c_i \leq H, \quad c_i \in \mathbb{Z}_+, \quad i \in I,$$

respectively. In case of

$$\min \left\{ -\chi_L^*, -\chi_W^*, -\chi_H^*, \min_{t \in T} d_{n_2+t}, -\min_{i=1,2,3} d_{n_5+i} \right\} \geq 0,$$

the current basis is optimal with respect to the whole problem and not only for the restricted master problem.

Some modification of the MCLP results when a supply  $\kappa > 1$  of identical containers is available. In this case the total value of pieces packed into the  $\kappa$  containers has to be maximized. The resulting problem is considered in Exercise 11.4.

Lower bounds, assigned to the MCLP with identical containers, are proposed in [8, 23] which are based on combinatorial case-by-case analysis. There, the MCLP is named *three-dimensional Bin Packing Problem*. A model for the problem with an assortment of different containers is asked for in Exercise 11.5.

### 11.5.3 A Layer Relaxation for the Container Loading Problem

Considering the bar relaxation, a three-dimensional packing pattern is, in fact, replaced by one-dimensional (bar) patterns. Similarly, a reduction of the considered dimension from three to two is possible, too. The *layer relaxation* considered below can be applied for the CLP as well as for the MCLP.

Each three-dimensional packing pattern of a container can easily be described by a sequence (set) of two-dimensional *layer patterns*. Let  $A = (A_1, \dots, A_n)^\top \in \mathbb{Z}_+^n$  be a feasible packing vector for the container  $C = \{(x, y, z) : 0 \leq x \leq L, 0 \leq y \leq W, 0 \leq z \leq H\}$ . Then there exists a sequence of piece indices  $\pi_1, \dots, \pi_k$  with

$\pi_j \in I$ ,  $k := \sum_{i \in I} A_i$ ,  $\text{card}(\{j : \pi_j = i\}) = A_i$ , and  $k$  corresponding allocation points  $p_j = (x_j, y_j, z_j) \in \mathbb{R}^3$ ,  $j = 1, \dots, k$ , which represent the packing pattern. We assume that index set  $J_i := \{j \in \{1, \dots, k\} : \pi_j = i\}$  represents all pieces of type  $i$  of the packing variant. For each  $p \in \{1, \dots, L\}$ , an  $L$ -layer is defined by

$$S_p^L := \{(x, y, z) : p - 1 < x < p, 0 \leq y \leq W, 0 \leq z \leq H\}.$$

The related  $L$ -layer pattern contains all pieces  $\pi_j$  for which  $B_{\pi_j}(p_j) \cap S_p^L \neq \emptyset$  holds. For each allocated piece  $i = \pi_j$  with  $B_{\pi_j}(p_j) \cap S_p^L \neq \emptyset$  the  $L$ -layer  $S_p^L$  contains  $w_i h_i$  units of volume of pieces  $i$  placed at allocation point  $(p - 1, y_j, z_j)$ .

Since a layer has always a thickness of one unit, in fact a two-dimensional packing pattern results. For the related packing vector  $a^p = (a_{1p}, \dots, a_{np})^\top \in \mathbb{Z}_+^n$ , we have

$$a_{ip} = \text{card}(\{j : \pi_j = i, x_j < p \leq x_j + \ell_i\}).$$

Neglecting the real position of the  $L$ -layers, and some further constraints, we obtain the layer relaxation. Index  $p$  is used to differentiate the  $L$ -layer vectors which are collected in index set  $P$ . In a similar way we define  $W$ -layer patterns  $b^q = (b_{1q}, \dots, b_{nq})^\top$ ,  $q \in Q$ , and  $H$ -layer patterns  $c^r = (c_{1r}, \dots, c_{nr})^\top$ ,  $r \in R$ , associated to the  $W$ - and  $H$ -layers, respectively.

*Example 11.4* Consider again Example 11.3. Within a container with  $L = 25$ ,  $W = 10$ , and  $H = 15$  two pieces of type  $\ell_1 = 10$ ,  $w_1 = 10$ ,  $h_1 = 7$  are placed with allocation points  $(0, 0, 0)$  and  $(0, 0, 7)$  and three pieces of type  $\ell_2 = 15$ ,  $w_2 = 7$ ,  $h_2 = 5$  with allocation points  $(10, 0, 0)$ ,  $(10, 0, 5)$ , and  $(10, 0, 10)$ . Then we obtain the following layer vectors:

- $L$ -layer patterns:  $a^1 = (2, 0)^\top$  (10 times),  $a^2 = (0, 3)^\top$  (15 times),
- $W$ -layer patterns:  $b^1 = (2, 0)^\top$  (3 times),  $b^2 = (2, 3)^\top$  (7 times),
- $H$ -layer patterns:  $c^1 = (1, 1)^\top$  (14 times),  $c^2 = (0, 1)^\top$  (1 times).

If we now assign non-negative integer variables  $x_p$ ,  $y_q$ , and  $z_r$  to the  $L$ -,  $W$ -, and  $H$ -layer patterns, the following equation results:

$$A_i = \frac{1}{\ell_i} \sum_{p \in P} a_{ip} x_p = \frac{1}{w_i} \sum_{q \in Q} b_{iq} y_q = \frac{1}{h_i} \sum_{r \in R} c_{ir} z_r. \quad (11.16)$$

Replacing  $A_i$  according to (11.16), we obtain a model of the *layer relaxation* for the Container Loading Problem:

### Layer relaxation of the CLP

$$\eta = \max \sum_{p \in P} \sum_{t \in T} \gamma_t \sum_{i \in I_t} a_{ip} x_p \quad \text{s.t.} \quad (11.17a)$$

$$\sum_{p \in P} w_i a_{ip} x_p - \sum_{q \in Q} \ell_i b_{iq} y_q = 0, \quad i \in I, \quad (11.17b)$$

$$\sum_{p \in P} h_i a_{ip} x_p - \sum_{r \in R} \ell_i c_{ir} z_r = 0, \quad i \in I, \quad (11.17c)$$

$$\sum_{p \in P} \sum_{i \in I_t} \frac{1}{\ell_i} a_{ip} x_p = u_t, \quad t \in T^=, \quad (11.17d)$$

$$\sum_{p \in P} \sum_{i \in I_t} \frac{1}{\ell_i} a_{ip} x_p \geq \underline{u}_t, \quad t \in T^{\geq}, \quad (11.17e)$$

$$\sum_{p \in P} \sum_{i \in I_t} \frac{1}{\ell_i} a_{ip} x_p \leq \bar{u}_t, \quad t \in T^{\leq}, \quad (11.17f)$$

$$\sum_{p \in P} x_p \leq L, \quad \sum_{q \in Q} y_q \leq W, \quad \sum_{r \in R} z_r \leq H, \quad (11.17g)$$

$$x_p \geq 0, \quad p \in P, \quad y_q \geq 0, \quad q \in Q, \quad z_r \geq 0, \quad r \in R. \quad (11.17h)$$

Due to neglecting in particular the integer condition, model (11.17) is an LP relaxation of the CLP. In principle, it could be solved applying the column generation technique. In the general case, a costly two-dimensional rectangle packing problem has to be solved as slave problem. For that reason, an application of the layer relaxation to the CLP is strongly restricted. Only in that case when a particular structure of the three-dimensional patterns, required in a certain application scenario, leads to a slave problem which is efficiently solvable, then the bound provided by the layer relaxation can be meaningful.

As a possibility to apply the layer relaxation, in [1] it is further relaxed leading to a relaxation similar to the bar relaxation.

## 11.6 Exercises

**Exercise 11.1** Verify that inequalities (11.3)–(11.5) model the non-overlapping of pieces.

**Exercise 11.2** In which way can the elements of (reduced) sets of potential allocation points (Sect. 2.7) be used in each of the three directions to possibly improve the rating of a pattern?

**Exercise 11.3** Show that, if  $f(s) > 0$  and  $v(s) > 0$  for all non-empty patterns  $s$ , then each function of form

$$\beta(s) = \frac{f(s) + \alpha}{v(s) + \gamma} \quad \text{with } \alpha \leq 0, \gamma > 0 \text{ and } f(s) + \alpha > 0 \quad \forall s$$

fulfills the demands on a rating function (Sect. 11.2.3).

**Exercise 11.4** A modification of the CLP results when  $\kappa > 1$  identical containers are available and the total value of the packed pieces has to be maximized. Formulate an appropriate model and an ILP model of the related bar relaxation.

**Exercise 11.5** Let an assortment of sufficiently large containers be given, possibly of different shapes. The task is to find a subset of the containers having minimal total volume such that all  $u_t$  pieces of type  $t \in T$  can be packed. Formulate a corresponding model and a model of the related bar relaxation.

## 11.7 Solutions

**To Exercise 11.1** If  $\sum_{A \in O_c} \delta_{1i}^A = 0$  holds, then piece  $i$  is not allocated, and  $\lambda_{ij}^A = 0$  and  $\lambda_{ji}^A = 0$  follow for all  $A \in O_c = \{X, Y, Z\}$  and  $j \in I, j \neq i$ . The assumptions  $\sum_{A \in O_c} \delta_{1i}^A = 1$  and  $\sum_{A \in O_c} \delta_{1j}^A = 0$  for  $i \neq j$  imply  $\lambda_{ij}^A = 0$  and  $\lambda_{ji}^A = 0$  for all  $A \in O_c$ . Moreover, if  $\sum_{A \in O_c} \delta_{1i}^A = 1$  and  $\sum_{A \in O_c} \delta_{1j}^A = 1$  hold, then  $\sum_{A \in O_c} (\lambda_{ij}^A + \lambda_{ji}^A) = 1$  follows, i.e., both pieces lie in some direction side by side. The minimum distance in this direction is enforced by condition (11.3).

**To Exercise 11.2** Generally, the size parameters of the container should be a non-negative integer combination of the respective piece sizes. Otherwise, a reduction of the container size would be possible.

Within an algorithm based on the contour concept, further reductions of the remaining volume are possible in analogy to the two-dimensional case (Sect. 5.6.1).

**To Exercise 11.3** It is easy to see, if  $v(s) = v(t)$  and  $f(s) > f(t)$ , then we have:

$$\beta(s) = \frac{f(s) + \alpha}{v(s) + \gamma} > \frac{f(t) + \alpha}{v(t) + \gamma} = \beta(t).$$

Moreover, if  $v(s) > v(t)$  and  $\frac{f(s)}{v(s)} = \frac{f(t)}{v(t)}$  hold, then

$$\begin{aligned} (v(s) + \gamma)(v(t) + \gamma)(\beta(s) - \beta(t)) &= (f(s) + \alpha)(v(t) + \gamma) - (f(t) + \alpha)(v(s) + \gamma) \\ &= \gamma(f(s) - f(t)) - \alpha(v(s) - v(t)) = \gamma f(t) \left( \frac{f(s)}{f(t)} - 1 \right) - \alpha v(t) \left( \frac{v(s)}{v(t)} - 1 \right) \\ &= \left( \frac{v(s)}{v(t)} - 1 \right) (\gamma f(t) - \alpha v(t)) > 0 \end{aligned}$$

follows. Hence,  $\beta(s) > \beta(t)$  is satisfied.

**To Exercise 11.4** Using the notations of Sect. 11.5.1 we can easily formulate a model of Kantorovich-type. To this end, let  $a_{ik}$  denote the number of pieces of type  $i$  assigned to container  $k$ . Then a basic model of maximizing the volume utilization is as follows:

$$\begin{aligned} \max & \sum_{k=1}^{\kappa} \sum_{i \in I} g_i a_{ik} \quad \text{s.t.} \\ & \sum_{k=1}^{\kappa} a_{ik} \leq u_i, \quad i \in I. \\ & a^k = (a_{uk})_{i \in I} \in \mathbb{Z}^{|I|} \text{ represents a feasible pattern,} \end{aligned}$$

A model of the related bar relaxation is as follows:

$$\begin{aligned} \eta &= \max \sum_{p \in P} \sum_{t \in T} \sum_{i \in I_t} \frac{g_t}{v_t} \ell_i a_{ip} x_p \quad \text{s.t.} \\ & \sum_{p \in P} \ell_i a_{ip} x_p = \sum_{q \in Q} w_i b_{iq} y_q = \sum_{r \in R} h_i c_{ir} z_r, \quad i \in I, \\ & \sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p = u_t v_t, \quad t \in T^=, \\ & \sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \geq \underline{u}_t v_t, \quad t \in T^\geq, \\ & \sum_{p \in P} \sum_{i \in I_t} \ell_i a_{ip} x_p \leq \bar{u}_t v_t, \quad t \in T^\leq, \\ & \sum_{p \in P} x_p \leq W H \kappa, \quad \sum_{q \in Q} y_q \leq L H \kappa, \quad \sum_{r \in R} z_r \leq L W \kappa, \\ & x_p \geq 0, p \in P, \quad y_q \geq 0, q \in Q, \quad z_r \geq 0, r \in R. \end{aligned}$$

In difference to model (11.10), in this model it is no longer guaranteed that an equation of form (11.10b) or (11.10c) is fulfilled for each container (if  $\kappa > 1$ ).

**To Exercise 11.5** We consider here a MCLP with an unlimited number of available containers of types  $C_k = \{(x, y, z) : 0 \leq x \leq L_k, 0 \leq y \leq W_k, 0 \leq z \leq H_k\}$ ,  $k \in K$ , in which all  $u_t$  pieces of type  $t$  have to be packed for all  $t \in T = \{1, \dots, m\}$ . The goal is to minimize the total container volume. Let  $V_k = L_k W_k H_k$ ,  $k \in K$ . We use again index set  $I = \{1, \dots, n\}$  to represent all possible permitted orientations of the

pieces. The length, width, and height of a (non-rotatable) piece  $i \in I$  are denoted again by  $\ell_i$ ,  $w_i$ , and  $h_i$ .

Index set  $J_k$  represents the set of all feasible three-dimensional packing patterns  $A^{jk} \in \mathbb{Z}_+^n$ ,  $j \in J_k$ , of a container of type  $k$ . Moreover,  $\mu_{jk}$  denotes the frequency containers of type  $k$  are packed according to pattern  $j$ . Hence, for the MCLP with various container types, we obtain the following model:

$$\begin{aligned}\eta &= \min \sum_{k \in K} V_k \sum_{j \in J_k} \mu_{jk} \quad \text{s.t.} \\ &\sum_{k \in K} \sum_{j \in J_k} \sum_{i \in I_t} A_{ijk} \mu_{jk} \geq u_t, \\ t \in T, \quad \mu_{jk} &\geq 0, \quad \mu_{jk} \in \mathbb{Z}_+, \quad j \in J_k, \quad k \in K.\end{aligned}$$

If  $\kappa_k$ ,  $k \in K$ , denotes the number of filled containers of type  $k$  and  $x_{pk}$ , the frequency the  $p$ th  $L$ -pattern of length  $L_k$  ( $p \in P_k$ ) is used (analogously, let  $y_{qk}$ ,  $q \in Q_k$ , and  $z_{rk}$ ,  $r \in R_k$  be defined), then we obtain a model of the bar relaxation applied to the MCLP with different container types:

$$\begin{aligned}\eta &= \min \sum_{k \in K} V_k \kappa_k \quad \text{s.t.} \\ \sum_{p \in P_k} \ell_i a_{ipk} x_{pk} &= \sum_{q \in Q_k} w_i b_{iqk} y_{qk} = \sum_{r \in R_k} h_i c_{irk} z_{rk}, \quad i \in I, \quad k \in K, \\ \sum_{k \in K} \sum_{p \in P_k} \sum_{i \in I_t} \ell_i a_{ipk} x_{pk} &\geq u_t v_t, \quad t \in T, \\ \sum_{p \in P_k} x_{pk} - WH\kappa_k &\leq 0, \quad \sum_{q \in Q_k} y_{qk} - LH\kappa_k \leq 0, \quad \sum_{r \in R_k} z_{rk} - LW\kappa_k \leq 0, \quad k \in K, \\ x_{pk} &\geq 0, \quad p \in P_k, \quad y_{qk} \geq 0, \quad q \in Q_k, \quad z_{rk} \geq 0, \quad r \in R_k.\end{aligned}$$

## References

1. G. Belov, V. Kartak, H. Rohling, G. Scheithauer, One-dimensional relaxations and LP bounds for orthogonal packing. *Int. Trans. Oper. Res.* **16**, 745–766 (2009)
2. E.E. Bischoff, M.D. Marriott, A comparative evaluation of heuristics for container loading. *Eur. J. Oper. Res.* **44**(2), 267–276 (1990)
3. E.E. Bischoff, M.S.W. Ratcliff, Issues in the development of approaches to container loading. *OMEGA* **23**(4), 377–390 (1995)
4. E.E. Bischoff, F. Janetz, M.S.W. Ratcliff, Loading pallets with non-identical items. *Eur. J. Oper. Res.* **84**(3), 681–692 (1995)
5. A. Bortfeldt, Eine Heuristik für Multiple Containerladeprobleme. *Diskussionsbeiträge des Fachbereichs Wirtschaftswissenschaft 257*, FernUniversität Hagen (1998)

6. A. Bortfeldt, H. Gehring, Ein Tabu Search-Verfahren für Containerbeladungsprobleme mit schwach heterogenem Kistenvorrat. *OR Spektrum* **20**(4), 237–250 (1998)
7. A. Bortfeldt, D. Mack, A heuristic for the three-dimensional strip packing problem. *Eur. J. Oper. Res.* **183**(3), 1267–1279 (2007)
8. M.A. Boschetti, New lower bounds for the three-dimensional finite bin packing problem. *Discrete Appl. Math.* **140**(1–3), 241–258 (2004)
9. A.P. Davies, E.E. Bischoff, Weight distribution considerations in container loading. *Eur. J. Oper. Res.* **114**(3), 509–527 (1999)
10. J.L. de Castro Silva, N.Y. Soma, N. Maculan, A greedy search for the three-dimensional bin packing problem: the packing static stability case. *Int. Trans. Oper. Res.* **10**(2), 141–153 (2003)
11. L. Epstein, R. van Stee, This side up!, in *Approximation and Online Algorithms, Second International Workshop, WAOA 2004*, Bergen, Norway, 14–16 September 2004. Revised Selected Papers (2005), pp. 48–60
12. G. Fasano, Cargo analytical integration in space engineering: a three-dimensional packing model, in *Operational Research in Industry*, ed. by S. Gliozzi, E. Johnson, R. Tadei (Springer, Manhatten, 1999)
13. H. Gehring, A. Bortfeldt, A genetic algorithm for solving the container loading problem. *Int. Trans. Oper. Res.* **4**(5–6), 401–418 (1997)
14. H. Gehring, K. Menschner, M. Meyer, A computer-based heuristic for packing pooled shipment containers. *Eur. J. Oper. Res.* **44**, 277–288 (1990)
15. J.A. George, D.F. Robinson, A heuristic for packing boxes into a container. *Comput. Oper. Res.* **7**, 147–156 (1980)
16. P.C. Gilmore, R.E. Gomory, Multistage cutting stock problems of two and more dimensions. *Oper. Res.* **13**, 94–120 (1965)
17. R.W. Haessler, Multimachine roll trim–problems and solutions. *TAPPI* **63**(1), 71–74 (1980)
18. M. Hifi, I. Kacem, S. Nègre, L. Wu, A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes Discrete Math.* **36**, 993–1000 (2010)
19. H. Isermann, Ein Planungssystem zur Optimierung der Palettenbeladung mit kongruenten rechteckigen Versandgebinde. *OR Spektrum* **9**, 235–249 (1987)
20. T. Lengauer, M. Schäfer, Combinatorial optimization techniques for three-dimensional arrangement problems, in *Mathematics - Key Technology for the Future*, ed. by W. Jäger, H.-J. Krebs (Springer, Heidelberg, 2003), pp. 63–73
21. C.-S. Liao, C.-H. Hsu, New lower bounds for the three-dimensional orthogonal bin packing problem. *Eur. J. Oper. Res.* **225**, 244–252 (2013)
22. A. Lodi, S. Martello, D. Vigo, Heuristic algorithms for the three-dimensional bin packing problem. *Eur. J. Oper. Res.* **141**(2), 410–420 (2002)
23. S. Martello, D. Pisinger, D. Vigo, The three-dimensional bin packing problem. *Oper. Res.* **48**, 256–267 (2000)
24. F.K. Miyazawa, Y. Wakabayashi, Three-dimensional packings with rotations. *Comput. Oper. Res.* **36**, 2801–2815 (2009)
25. R. Morabito, M.N. Arenales, An AND/OR graph approach to the container loading problem. *Int. Trans. Oper. Res.* **1**, 59–73 (1994)
26. A. Moura, J.F. Oliveira, A GRASP approach to the container loading problem. *IEEE Intell. Syst.* **2005**, 50–57 (2005)
27. M. Padberg, Packing small boxes into a big box. *Math. Methods Oper. Res.* **52**, 1–21 (2000)
28. D. Pisinger, Heuristics for the container loading problem. *Eur. J. Oper. Res.* **141**(2), 382–392 (2002)
29. Y.G. Stoyan, N.I. Gil, A. Pankratov, G. Scheithauer, Packing non-convex polytopes into a parallelepiped. Preprint MATH-NM-06-2004, Technische Universität Dresden (2004)
30. Y.G. Stoyan, N. Gil, G. Scheithauer, A. Pankratov, I. Magdalina, Packing of convex polytopes into a parallelepiped. *Optimization* **54**(2), 215–235 (2005)

# Chapter 12

## Packing of Polygonal Pieces

Besides problems of optimal packing rectangular-shaped objects, as considered in previous chapters, the optimal arrangement of non-rectangular pieces is frequently of interest. Here we address aspects concerning the mutual position of polygonal pieces and the containment within a polygonal region. Moreover, we discuss the basic principles of heuristic solution approaches.

### 12.1 Problem Statement and Modeling

Based on the concept of  $\Phi$ -functions, which is already introduced in Sect. 1.5, a model for the strip packing problem with polygonal pieces is presented. Moreover, related optimization problems are discussed.

#### 12.1.1 Strip Packing

The optimal placement of polygonal objects (pieces) within a region of minimal size constitutes a problem which arises in various real-world scenarios as, for instance, in textile or leather industry, or in metallurgy.

For the sake of an easy description, we assume in the most cases that rotation of pieces is not permitted. However, on respective places we will also consider the general case.

Let polygonal pieces (polygons)  $P_i \subset \mathbb{R}^2$ ,  $i \in I = \{1, \dots, m\}$ , be given which have to be placed entirely and without overlap within a polygonal region  $R \subset \mathbb{R}^2$  as, for instance, a strip of fixed width  $W$  and minimal height, thus  $R = R(H) = W \times H$ .

More generally, let  $p \in \mathbb{R}^q$  be a vector of  $q$  ( $q \geq 1$ ) parameters which indicate the shape of  $R$ . Clearly, in the scenario considered in this subsection we have  $q = 1$  and  $p$  contains only the height  $H$ , whereas  $q = 2$  and  $p = (W, H)$  if the strip width can be varied, too.

In our considerations, we assume that all polygons represent real objects, that means, piece  $P_i$  ( $i \in I$ ) is bounded, its interior  $\text{int}(P_i)$  is connected, and  $P_i = \text{cl}(\text{int}(P_i))$  is satisfied. ( $\text{cl}(S)$  and  $\text{int}(S)$  denote the *closure* and the *interior* of set  $S$ , respectively.)

In order to model the placement problem, we use  $\Phi$ -functions as defined in Sect. 1.5. If only translations of objects are allowed, then a  $\Phi$ -function of two polygons  $P_i$  and  $P_j$  is a continuous function  $\Phi_{ij} : \mathbb{R}^4 \rightarrow \mathbb{R}$  possessing the following characteristic properties:

$$\Phi_{ij}(u_i, u_j) = \begin{cases} > 0, & \text{if } P_i(u_i) \cap P_j(u_j) = \emptyset, \\ = 0, & \text{if } \text{int}(P_i(u_i)) \cap P_j(u_j) = \emptyset \text{ and } P_i(u_i) \cap P_j(u_j) \neq \emptyset, \\ < 0, & \text{if } \text{int}(P_i(u_i)) \cap P_j(u_j) \neq \emptyset, \end{cases}$$

where  $u_i \in \mathbb{R}^2$  and  $u_j \in \mathbb{R}^2$  denote the respective translation vectors (allocation points). Hence, both polygons,  $P_i(u_i)$  and  $P_j(u_j)$ , do not overlap if and only if  $\Phi_{ij}(u_i, u_j) \geq 0$  holds. Furthermore, we model the *containment condition*  $P_i(u_i) \subseteq R$  likewise by means of a  $\Phi$ -function. To this end, we consider the non-overlapping of piece  $P_i$  and the (unbounded) region

$$P_0(H) := \text{cl}(\mathbb{R}^2 \setminus R(H)) = \mathbb{R}^2 \setminus \text{int}(R(H))$$

being the complementary set of  $\text{int}(R(H))$ . Point set  $P_0(H)$  can be considered as a generalized object so that the concept of  $\Phi$ -functions can be applied to  $P_0$ , too. Without loss of generality, we may assume that  $P_0$  is immovable. Therefore, the non-overlapping of  $P_0(H)$  and  $P_i$  can be modeled by an inequality of form  $\phi_i(u_i, H) := \Phi_i(u_i, 0) \geq 0$  where  $\Phi_i$  denotes a  $\Phi$ -function of  $P_0(H)$  and  $P_i$  ( $i \in I$ ). For short, let

$$u = (u_1, \dots, u_m)^\top \in \mathbb{R}^{2m}$$

represent the vector of all translation vectors. Furthermore, let  $F : \mathbb{R}^{2m+q} \rightarrow \mathbb{R}$  denote an appropriate objective function. Then we can formulate the polygon placement problem as follows:

### **$\Phi$ -functions model of the polygon strip packing problem**

$$F^*(u^*, p^*) = \min F(u, p) \quad \text{s.t.} \tag{12.1}$$

$$u \in \Omega := \{u \in \mathbb{R}^{2m} : \Phi_{ij}(u_i, u_j) \geq 0, \phi_i(u_i, p) \geq 0, i, j \in I, i < j\}. \tag{12.2}$$

Set  $\Omega$  represents the set of all feasible (packing) patterns. It is easy to see,  $\Omega$  is not connected, in general. Moreover, connected components are non-convex, in general. We will observe both facts by means of examples in Sect. 12.2. In the case that polygons have to be placed within a strip of fixed width  $W$  and minimal height  $H$ , i.e., in  $R = R(H) = \{(x, y) : 0 \leq x \leq W, 0 \leq y \leq H\}$ , then simply  $F = F(u, p) := H$  can be used in model (12.1) as appropriate objective function. The principal construction of  $\Phi$ -functions for pairs of polygons is addressed in Sect. 12.2. The more general case of defining  $\Phi$ -functions for objects whose frontier consists of straight-line and arc segments only, is considered, for instance, in [5].

### 12.1.2 Related Problems

Besides the polygon strip packing problem, aiming to minimize material usage as considered in the previous subsection, similar problems concerning the best possible material utilization are of high interest. Within the metalworking industry one of the cutting tasks consists in finding a pattern with maximal number of (identical) polygonal pieces which can be obtained from a given (rectangular) sheet. A comprehensive study of recently applied cutting technologies can be found in [30]. Moreover, the particular case, for instance addressed in [11], where identical rectangular pieces have to be placed, is considered in more detail in Chap. 10.

Due to technological reasons, patterns regular in a certain sense are frequently preferred when cutting non-regular items. Patterns which only have allocation points within a predefined lattice are used, for example, in [26, 29, 38, 39]. Such kind of allocating objects is called *lattice packing*,

Furthermore, in the leather goods industry, among others, the material is not homogeneous and not rectangular, in general. These circumstances enforce appropriate adaptations of the solution approaches developed for simpler scenarios. Cutting problems regarding some quality aspects are addressed in Chap. 9.

In this respect, so-called *Minimum Enclosure Problems* are of interest. For example, find an allocation point  $u^*$  for polygon  $P_j$  such that  $\text{int}(P_i(0)) \cap P_j(u^*) = \emptyset$  holds and the convex hull of  $P_i(0) \cup P_j(u^*)$  has minimal area. Investigations concerning such optimization problems can be found, for instance, in [17, 27]. Rotation of objects is permitted there. The decision problem *Is it possible to place two polygons without overlap into a third polygon?* is addressed, for instance, in [19]. Moreover, a linear programming based method to verify whether a polygon can fit within another, can be found in [18].

The computation of an area-minimal rectangle which encloses a given polygon is subject of Exercise 12.5.

Problems of the form *find an area-maximal axes-parallel rectangle which fits within a given polygon* are of interest as well, for instance addressed in [33]. The particular case of a given convex polygon is considered in Exercise 12.9.

## 12.2 Construction of $\Phi$ -Functions

Without loss of generality, we assume that the polygonal pieces  $P_i$  are given in *normalized position* which is defined by

$$\min \{x : (x, y) \in P_i\} = 0 \quad \text{and} \quad \min \{y : (x, y) \in P_i\} = 0.$$

As *reference point* we take the origin of the eigen-coordinate system for each  $i \in I$ . A translation of  $P_i$  by vector  $u \in \mathbb{R}^2$  and the placement of  $P_i$  with allocation point  $u$  is again denoted by  $P_i(u)$ . Thus, we have

$$P_i(u) := u + P_i := \{v \in \mathbb{R}^2 : v = u + w, w \in P_i\}.$$

### 12.2.1 Convex Polygons

To represent a convex polygon  $P_i$ ,  $i \in I$ , two variants are in use. As a first variant, the intersection of a finite number of half-planes

$$G_{ik} = \{v \in \mathbb{R}^2 : g_{ik}(v) \leq 0\}, \quad k \in K_i = \{1, \dots, n_i\}, \quad (12.3)$$

are exploited where  $g_{ik} : \mathbb{R}^2 \rightarrow \mathbb{R}$  denotes an affine linear function for each  $k \in K_i$  and  $n_i$  is their total number. This leads to a first possibility of representing a polygon:

$$P_i = \{v \in \mathbb{R}^2 : g_{ik}(v) \leq 0, k \in K_i\} = \bigcap_{k \in K_i} G_{ik}. \quad (12.4)$$

In any case, we assume, without loss of generality, that  $n_i$  is the minimal number necessary to represent  $P_i$ . A second variant uses the convex hull of all corner (extreme) points  $v_{ik}$ ,  $k \in K_i$ , of  $P_i$ ,  $i \in I$ . That means, we also have the representation of  $P_i$  in the form

$$P_i = \text{conv} \{v_{ik} : k \in K_i\}. \quad (12.5)$$

Obviously, the number of edges (half-planes) coincides with the number of corner points if using a minimal representation of  $P_i$ . Additionally, we suppose a counter-clockwise numbering of the corner points.

To describe the mutual position of two placed polygons  $P_i(u_i)$  and  $P_j(u_j)$  the difference vector  $u := u_j - u_i$  is essential. For that reason, we firstly consider polygon  $P_i$  to be immovable with  $u_i = 0$ . Polygon  $P_j(u)$  does not overlap polygon  $P_i(0)$ , that means,  $\text{int}(P_i(0)) \cap P_j(u) = \emptyset$  holds, if and only if  $P_i(0)$  and at least one half-plane  $u + G_{jl}$ ,  $l \in K_j$ , or if  $P_j(u)$  and at least one half-plane  $G_{ik}$ ,  $k \in K_i$ , do not overlap. This

aspect is illustrated in Fig. 12.2. Due to the convexity of  $P_i$  and  $P_j$  this happens if

$$\exists l \in K_j : \min_{k \in K_i} g_{jl}(v_{ik} - u) \geq 0 \quad \text{or} \quad \exists k \in K_i : \min_{l \in K_j} g_{ik}(v_{jl} + u) \geq 0$$

holds. These conditions can easily be transformed into

$$\phi_{ij}(u) := \max \left\{ \max_{l \in K_j} \min_{k \in K_i} g_{jl}(v_{ik} - u), \max_{k \in K_i} \min_{l \in K_j} g_{ik}(v_{jl} + u) \right\} \geq 0. \quad (12.6)$$

In this way we obtain a  $\Phi$ -function

$$\Phi_{ij}(u_i, u_j) := \phi_{ij}(u_j - u_i) \quad (12.7)$$

for a pair  $P_i$  and  $P_j$  of convex polygons,  $i \neq j$ ,  $i, j \in I$ .

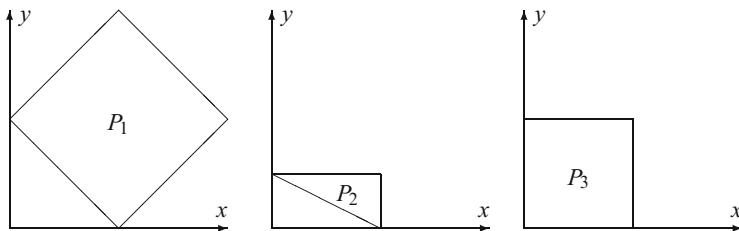
*Example 12.1* With the help of this example, we illustrate some issues. Let three polygons  $P_i$ ,  $i = 1, 2, 3$  be given with  $n_1 = n_3 = 4$ ,  $n_2 = 3$ , and

| $k$ | $g_{1k}$     | $v_{1k}$ | $g_{2k}$      | $v_{2k}$ | $g_{3k}$ | $v_{3k}$ |
|-----|--------------|----------|---------------|----------|----------|----------|
| 1   | $x - y - 2$  | (2, 0)   | $x - 2$       | (2, 0)   | $-y$     | (0, 0)   |
| 2   | $x + y - 6$  | (4, 2)   | $y - 1$       | (2, 1)   | $x - 2$  | (2, 0)   |
| 3   | $-x + y - 2$ | (2, 4)   | $-x - 2y + 2$ | (0, 1)   | $y - 2$  | (2, 2)   |
| 4   | $-x - y + 2$ | (0, 2)   |               |          | $-x$     | (0, 2)   |

These polygons are shown in Fig. 12.1. To model the non-overlapping of the two polygons, all edges and corner points of them have to be regarded as can be seen in Fig. 12.2 for the polygons  $P_1$  and  $P_2$ . ■

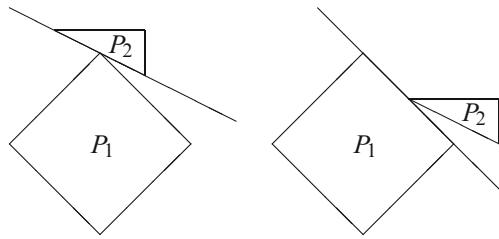
Due to the affine linearity of all  $g_{ik}$  there exist indices  $k_l \in K_i$  for all  $l \in K_j$  and indices  $l_k \in K_j$  for each  $k \in K_i$ , which do not depend on  $u$ , with

$$\min_{k \in K_i} g_{jl}(v_{ik} - u) = g_{jl}(v_{ik_l} - u), \quad \min_{l \in K_j} g_{ik}(v_{jl} + u) = g_{ik}(v_{jl_k} + u),$$



**Fig. 12.1** Polygons considered in Example 12.1

**Fig. 12.2** Mutual positions of two polygons



such that formula (12.6) implies

$$\phi_{ij}(u) = \max \left\{ \max_{l \in K_j} g_{jl}(v_{ik_l} - u), \max_{k \in K_i} g_{ik}(v_{jl_k} + u) \right\} \quad (12.8)$$

being a simpler formulation of  $\phi_{ij}$ . Point set

$$U_{ij} := \{u \in \mathbb{R}^2 : \phi_{ij}(u) \geq 0\}$$

contains all difference vectors  $u = u_j - u_i$  with  $\text{int}(P_i(u_i)) \cap P_j(u_j) = \emptyset$ . The closure of the complement of point set  $U_{ij}$ ,

$$\overline{U}_{ij} := \text{cl}(\mathbb{R}^2 \setminus U_{ij}) = \{u \in \mathbb{R}^2 : \phi_{ij}(u) \leq 0\},$$

is usually called the *no-fit polygon*. Obviously, we have

$$\overline{U}_{ij} = P_i(0) \oplus (-1)P_j(0) = \{w \in \mathbb{R}^2 : w = v - u, v \in P_i(0), u \in P_j(0)\}.$$

This representation is also known as the *Minkowski sum*. In particular, therefore we have

$$\overline{U}_{ij} = \text{conv} \{v_{ik} - v_{jl} : k \in K_i, l \in K_j\}. \quad (12.9)$$

Due to (12.9) the relation

$$\overline{U}_{ij} = -\overline{U}_{ji}$$

follows immediately for  $\overline{U}_{ij}$  by interchanging the two polygons.

In addition, set

$$U_{ij}^0 := \{u \in \mathbb{R}^2 : \phi_{ij}(u) = 0\},$$

i.e., the frontier of set  $U_{ij}$  and of  $\overline{U}_{ij}$ , is frequently named *hodograph*. The points  $u \in U_{ij}^0$  represent exactly those difference vectors  $u_j - u_i$  for which  $P_i(u_i)$  and  $P_j(u_j)$  are in contact but do not overlap.

As can be seen from formulas (12.6) and (12.8), the no-fit polygon of two convex polygons  $P_i$  and  $P_j$  is itself a convex polygon with at most  $|K_i| + |K_j|$  edges and corner points. However, the minimum number of edges is given by  $\max\{|K_i|, |K_j|\}$ .

*Example 12.2 (Example 12.1 Continued)* Because of  $k_1 = 4$ ,  $k_2 = 1$ ,  $k_3 = 3$ ,  $l_1 = 3$ ,  $l_2 = 3$ ,  $l_3 = 1$ , and  $l_4 = 2$ , we have with  $u = (x, y)$ :  
 $\phi_{12}(u) = \max \{\max_{l \in K_2} g_{2l}(v_{1k_l} - u), \max_{k \in K_1} g_{1k}(v_{2l_k} + u)\}$   
 $= \max \{-x - 2, -y - 1, x + 2y - 8, x - y - 3, x + y - 5, -x + y - 4, -x - y - 1\}$ .

Figure 12.3 shows different allocations of two polygons which are meaningful for the construction of the corresponding hodographs. Moreover, the set of all points where the two objects are in contact is drawn, too.

Obviously, the frontier of set  $\overline{U}_{ij}$  is formed by the line which is obtained by the reference point of  $P_j$  when shifting  $P_j$  around  $P_i$  and maintaining the contact of both objects. It coincides with the hodograph of polygons  $P_i$  and  $P_j$ . The three resulting hodographs are shown in Fig. 12.4. ■

**Proposition 12.1** In the particular case where  $P_i$  and  $P_j$  are rectangles with sizes  $\ell_i \times w_i$  and  $\ell_j \times w_j$ , respectively, we have

$$\phi_{ij}(u) = \max_{k \in K_i} \min_{l \in K_j} g_{ik}(v_{jl} + u) = \max_{l \in K_j} \min_{k \in K_i} g_{jl}(v_{ik} - u) \quad (12.10)$$

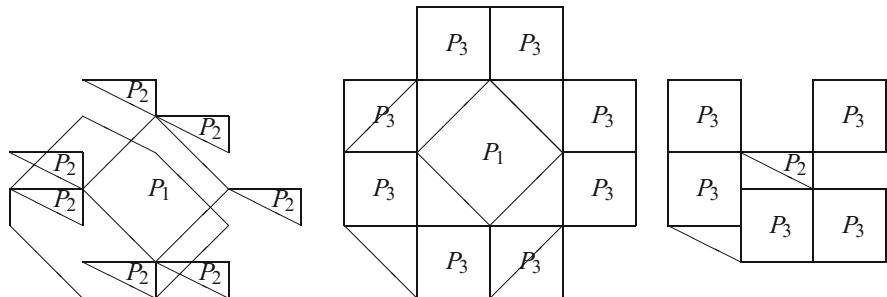


Fig. 12.3 Placements related to Example 12.2

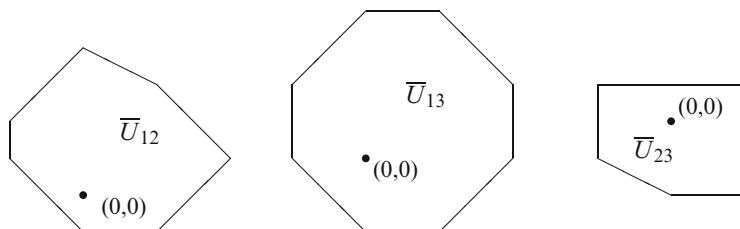


Fig. 12.4 Hodographs related to Example 12.2

for  $P_i(0)$  and  $P_j(u)$  as well as

$$\overline{U}_{ij} = \{(x, y) : -\ell_j \leq x \leq \ell_i, -w_j \leq y \leq w_i\}. \quad (12.11)$$

A proof of this proposition can be found in Exercise 12.8.

To model the *containment* of a polygon  $P_i$  within a convex polygon  $R$ , we assume, without loss of generality, that  $R$  is bounded and is given by

$$R := \{(x, y) : g_{0l}(x, y) \geq 0, l \in K_0\} = \text{conv}\{v_{0l} : l \in K_0\}.$$

In this representation the functions  $g_{0l}, l \in K_0 = \{1, \dots, n\}$ , are again affine linear. The points  $v_{0l}, l \in K_0$ , indicate all corner points of  $R$ . In difference to the description of polygons  $P_i, i \in I$ , we use here  $\geq$ -restrictions. Let

$$G_{0l} := \{(x, y) : g_{0l}(x, y) \geq 0\}$$

denote, in analogy to (12.3), a half-plane limiting  $R$ , i.e.,  $R \cap G_{0l} \subset \{(x, y) : g_{0l}(x, y) = 0\}$  holds,  $l \in K_0$ . Then, we have  $P_i \subseteq R$  for some  $i \in I$  if and only if  $\text{int}(P_i) \cap G_{0l} = \emptyset$  is satisfied for all  $l \in K_0$ . Hence, the containment of  $P_i$  in  $R$  can also be modeled as non-overlapping with convex regions. We define

$$P_0 := \bigcup_{l \in K_0} G_{0l} = \text{cl}(\mathbb{R}^2 \setminus R).$$

Obviously, region  $P_0$  is not convex. Let  $\psi_{il} : \mathbb{R}^2 \rightarrow \mathbb{R}$  denote that function which results as restricted  $\Phi$ -functions of  $P_i$  and  $G_{0l}$  when region  $G_{0l}$  is fixed. Since the functions  $g_{0l}$  are affine linear there exists an index  $\kappa_l \in K_i$ , independent on  $u$ , with

$$\psi_{il}(u) = g_{0l}(v_{i\kappa_l} + u) = \min\{g_{0l}(v_{ik} + u) : k \in K_i\}.$$

Moreover, since  $P_i(u)$  is completely contained in  $R$  if and only if  $\psi_{il}(u) \geq 0$  holds for all  $l \in K_0$ , we define

$$\phi_i(u) := \min_{l \in K_0} \psi_{il}(u) = \min_{l \in K_0} g_{0l}(v_{i\kappa_l} + u)$$

for  $i \in I$ . Hence, set

$$U_i := \{u \in \mathbb{R}^2 : \phi_i(u) \geq 0\}$$

represents the set of all allocation points  $u$  of  $P_i$  such that  $P_i(u) \subseteq R$  holds,  $i \in I$ . Consequently, a non-overlapping placement of two polygons  $P_i(u_i)$  and  $P_j(u_j)$  within  $R$  is given if the conditions

$$u_i \in U_i, \quad u_j \in U_j, \quad u_j - u_i \in \overline{U}_{ij}$$

are fulfilled.

*Example 12.3 (Example 12.2 Continued)* Now we suppose that all polygons have to be placed within a rectangle  $R$  with width  $W = 5$  and height  $H = 6$ . Then, we have  $g_{01}(x, y) = y$ ,  $g_{02}(x, y) = 5 - x$ ,  $g_{03}(x, y) = 6 - y$ ,  $g_{04}(x, y) = x$ , and  $v_{01} = (0, 0)$ ,  $v_{02} = (5, 0)$ ,  $v_{03} = (5, 6)$ ,  $v_{04} = (0, 6)$ . We obtain the (packing) regions  $U_i$ ,  $i \in I$ , as depicted in Fig. 12.5.

To demonstrate a natural placing principle, we consider different packing positions  $u_1$  of  $P_1$  and construct the sets  $(u_1 + U_{12}) \cap U_2$  for  $u_1 = (0, 0)$ ,  $u_1 = (0.5, 1)$ , and  $u_1 = (1, 2)$ , respectively, which are depicted in Fig. 12.6. The resulting sets of allocation points, also called *allocation regions*, for  $P_2$  in dependence on the allocation point  $u_1$  of  $P_1$ , are drawn in Fig. 12.7 for  $u_1 = (0, 0)$ ,  $u_1 = (0.5, 1)$ , and  $u_1 = (1, 2)$ .

Set  $U_{12}$  represents the set of all translation vectors  $u_2 - u_1$  such that  $\text{int}(P_1(u_1)) \cap P_2(u_2) = \emptyset$  holds. The restriction of  $U_{12}$  onto the allocation points  $u_1 \in U_1$ ,  $u_2 \in U_2$ , i.e., onto all allocation possibilities of  $P_1$  and  $P_2$  in  $R$ , yields for fixed  $u_1$  the point set  $(u_1 + U_{12}) \cap U_2$ . For each of these three allocation points  $u_1$ , this set is not connected. Furthermore, isolated points occur and the connected components are not convex, in general. ■

As the following example shows, set  $\Omega$  can possess exponentially many connected components. Hence, a successive investigation of all components of  $\Omega$  as a possible solution strategy is therefore not applicable for practical purposes, in general.

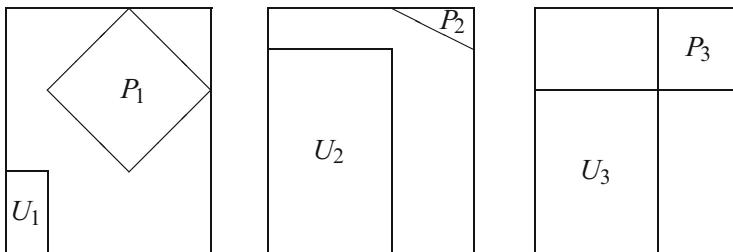


Fig. 12.5 Packing regions  $U_i$ ,  $i \in \{1, 2, 3\}$ , related to Example 12.3

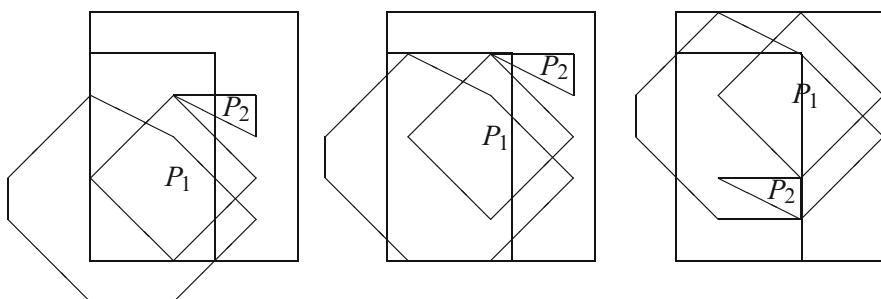
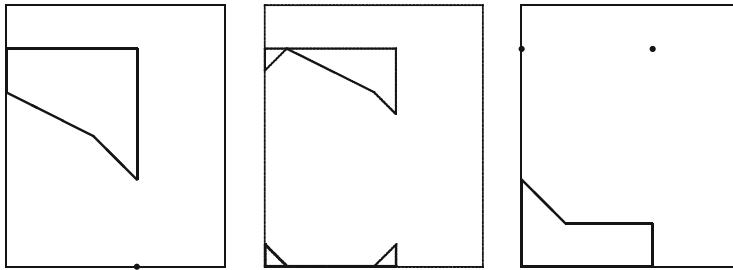
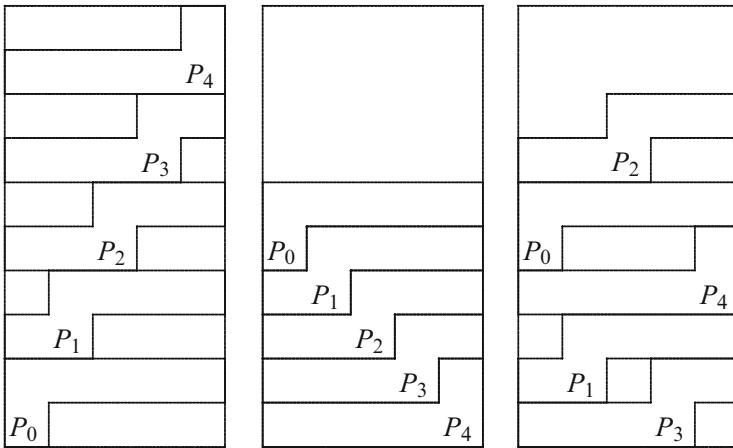


Fig. 12.6 Some patterns belonging to Example 12.3



**Fig. 12.7** Allocation regions  $U_{12}(u_1)$  of Example 12.3 for  $u_1 = (0, 0), (0.5, 1)$ , and  $(1, 2)$



**Fig. 12.8** Patterns of Example 12.4 with  $m = 4$

*Example 12.4* Let  $m \in \mathbb{N}$ . The task is to place polygonal pieces  $P_i$ ,  $i \in I = \{0, \dots, m\}$ , into a given strip of width  $W = m + 1$  and height  $2m + 2$ . The pieces are defined as follows:

$$P_i := \{(x, y) : 0 \leq x \leq i + 1, 0 \leq y \leq 1\} \cup \{(x, y) : i \leq x \leq m + 1, 1 \leq y \leq 2\}.$$

Obviously, the allocation points  $u_i := (0, m - i)$ ,  $i = 0, \dots, m$ , yield a pattern with minimal height  $m + 2$  (Fig. 12.8).

Each of the  $(m+1)!$  different sequences  $\pi \in \Pi(0, \dots, m)$  of the polygons defines a subregion  $\Omega(\pi)$  of  $\Omega$  with  $\Omega(\pi) \cap \Omega(\chi) = \emptyset$  for all permutations  $\chi \in \Pi(0, \dots, m)$  with  $\chi \neq \pi$ . This is true because of

$$\begin{aligned} \Omega(\pi) = & \{(u_0, \dots, u_m)^\top \in \mathbb{R}^{2m+2} : u_{i,1} = 0, i \in I, u_{\pi_0,2} \geq 0, u_{\pi_m,2} \leq 2m, \\ & u_{\pi_i,2} \geq u_{\pi_{i-1},2} + 1 \text{ for } \pi_{i-1} > \pi_i, \quad u_{\pi_i,2} \geq u_{\pi_{i-1},2} + 2, \text{ for } \pi_{i-1} < \pi_i\}. \end{aligned}$$

Each of the subregions is non-empty. As is shown in Exercise 12.3, set  $\Omega$  can have exponentially many components even in the case of convex polygons. ■

### 12.2.2 Non-convex Polygons

Several possibilities are available to represent non-convex polygons. A comprehensive description of appropriate methodologies can be found in [4]. In the following, we will use a representation as union of convex parts.

First of all, we consider the modeling of non-overlapping of two non-convex polygons. To avoid too many indices, let

$$P = \bigcup_{i \in I} P_i \quad \text{with} \quad P_i = \{v \in \mathbb{R}^2 : g_{ik}(v) \leq 0, k \in K_i\}, \quad i \in I, \quad (12.12)$$

and

$$Q = \bigcup_{j \in J} Q_j \quad \text{with} \quad Q_j = \{v \in \mathbb{R}^2 : h_{jl}(v) \leq 0, l \in L_j\}, \quad j \in J,$$

represent two non-convex polygons where  $P_i$ ,  $i \in I$ , and  $Q_j$ ,  $j \in J$ , are assumed to be convex. Furthermore, we define corresponding half-planes

$$G_{ik} = \{v \in \mathbb{R}^2 : g_{ik}(v) \leq 0\}, \quad k \in K_i, \quad H_{jl} = \{v \in \mathbb{R}^2 : h_{jl}(v) \leq 0\}, \quad l \in L_j.$$

Because of

$$P_i = \bigcap_{k \in K_i} G_{ik}, \quad i \in I, \quad Q_j = \bigcap_{l \in L_j} H_{jl}, \quad j \in J,$$

we obtain the following description of the two polygons:

$$P = \bigcup_{i \in I} \bigcap_{k \in K_i} G_{ik} \quad \text{and} \quad Q = \bigcup_{j \in J} \bigcap_{l \in L_j} H_{jl}.$$

Furthermore, let  $v_{ik}$ ,  $k \in K_i$ , and  $w_{jl}$ ,  $l \in L_j$ , denote the corner points of  $P_i$ ,  $i \in I$ , and  $Q_j$ ,  $j \in J$ , respectively, such that

$$P_i = \text{conv} \{v_{ik} : k \in K_i\} \quad \text{and} \quad Q_j = \text{conv} \{w_{jl} : l \in L_j\}$$

hold. Finally, let  $u_p = (x_p, y_p)$  and  $u_q = (x_q, y_q)$  denote the allocation points (translation vectors) of  $P$  and  $Q$ , and  $u = u_q - u_p$  the resulting difference vector. Both polygons  $P(u_p)$  and  $Q(u_q)$  do not overlap if and only if

$$\text{int}(P_i(0)) \cap Q_j(u) = \emptyset \quad \text{for all } i \in I, j \in J$$

holds. According to the characterization in (12.8), this is the case for some  $i \in I$  and some  $j \in J$  if

$$\phi_{ij}(u) = \max \left\{ \max_{l \in L_j} h_{jl}(v_{ik_l} - u), \max_{k \in K_i} g_{ik}(w_{jl_k} + u) \right\} \geq 0$$

is satisfied where the indices  $k_l \in K_i$ ,  $l \in L_j$ , and  $l_k \in L_j$ ,  $k \in K_i$ , are determined by

$$\min_{k \in K_i} h_{jl}(v_{ik} - u) = h_{jl}(v_{ik_l} - u) \quad \text{and} \quad \min_{l \in L_j} g_{ik}(w_{jl} + u) = g_{ik}(w_{jl_k} + u).$$

By means of  $\phi_{ij}$  we define, in analogy to Sect. 12.2.1, a  $\Phi$ -function  $\Phi_{ij}$  for the pair  $P_i$  and  $Q_j$  of convex polygons. In this way, we directly obtain a  $\Phi$ -function  $\Phi_{PQ}$  for the pair  $P$  and  $Q$  of non-convex polygons:

$$\begin{aligned} \Phi_{PQ}(u_p, u_q) &:= \min_{i \in I} \min_{j \in J} \Phi_{ij}(u_p, u_q) = \min_{i \in I} \min_{j \in J} \phi_{ij}(u_q - u_p) \\ &= \min_{i \in I} \min_{j \in J} \max \left\{ \max_{l \in L_j} h_{jl}(v_{ik_l} - u), \max_{k \in K_i} g_{ik}(w_{jl_k} + u) \right\}. \end{aligned}$$

To model the containment of a non-convex polygon  $P$  within another non-convex polygon  $R$ , we use the following representation for  $R$ : let

$$R^* := \text{cl}(\mathbb{R}^2 \setminus R)$$

denote the complement of  $\text{int}(R)$  with respect to  $\mathbb{R}^2$ . Point set  $R^* \subset \mathbb{R}^2$  is also a region having a polygonal frontier and with  $\text{cl}(\text{int}(R^*)) = R^*$ . Now we use a representation of  $R^*$  as union of convex regions:

$$R^* = \bigcup_{j \in J} R_j^* \tag{12.13}$$

where all sets  $R_j^*$  are convex and have polygonal frontiers, and  $J$  denotes an appropriate index set. Since now we have a description of  $R^*$  as for the non-convex polygon  $Q$ , we use the same notations for the restrictions which define  $R_j^*$ . Hence, polygon  $R_j^*$  can be represented in the form

$$R_j^* = \{v \in \mathbb{R}^2 : h_{jl}(v) \leq 0, l \in L_j\}$$

where  $L_j$  is a respective index set. Each function  $h_{jl}$ ,  $l \in L_j$ , defines a corresponding half-plane by

$$H_{jl} = \{v \in \mathbb{R}^2 : h_{jl}(v) \leq 0\}.$$

Consequently, we have

$$R_j^* = \bigcap_{l \in L_j} H_{jl}, \quad j \in J,$$

and, therefore,

$$R^* = \bigcup_{j \in J} \bigcap_{l \in L_j} H_{jl}.$$

Defining half-planes

$$H_{jl}^* := \text{cl}(\mathbb{R}^2 \setminus H_{jl}) = \{v \in \mathbb{R}^2 : h_{jl}(v) \geq 0\}$$

for all  $l \in L_j$  and  $j \in J$ , we obtain the following description for  $R$ :

$$R = \bigcap_{j \in J} \bigcup_{l \in L_j} H_{jl}^*.$$

To simplify the presentation, we assume that  $R$  lies completely within the interior of a compact region which is convex and has polygonal frontier, i.e.  $R \subset \text{int}(S)$ . We denote the corner points of  $R_j^* \cap S$  by  $w_{jl}$ ,  $l \in \tilde{L}_j$ , where  $\tilde{L}_j$  represents an appropriate index set. Obviously, set  $R_j^* \cap S$  is convex and we have

$$R_j^* \cap S = \text{conv} \{w_{jl} : l \in \tilde{L}_j\}.$$

Therefore, we can characterize the containment of  $P$  in  $R$  by means of the non-overlapping of  $P$  and  $R^*$ , and trace it back to the non-overlapping of convex polygons.

Because of  $R_j^* = \bigcap_{l \in L_j} H_{jl}$  with  $H_{jl} = \{v : h_{jl}(v) \leq 0\}$  and  $P_i = \bigcap_{k \in K_i} G_{ik}$  with  $G_{ik} = \{v : g_{ik}(v) \leq 0\}$ , polygon  $P_i(u)$  does not overlap  $R_j^*$  if and only if  $P_i(u)$  does not overlap any half-plane  $H_{jl}^*$  or if  $R_j^*(-u)$  does not overlap any half-plane  $G_{ik}$ . This is the case if

$$\exists l \in L_j : \min_{k \in K_i} h_{jl}(v_{ik} + u) \geq 0 \quad \text{or} \quad \exists k \in K_i : \min_{l \in L_j} g_{ik}(w_{jl} - u) \geq 0$$

holds. This condition is equivalent to

$$\phi_{ij}(u) := \max \left\{ \max_{l \in L_j} \min_{k \in K_i} h_{jl}(v_{ik} + u), \max_{k \in K_i} \min_{l \in L_j} g_{ik}(w_{jl} - u) \right\} \geq 0.$$

Hence,  $P_i(u) \subseteq R(0)$  holds if and only if

$$\phi_i(u) := \min_{j \in J} \max \left\{ \max_{l \in L_j} \min_{k \in K_i} h_{jl}(v_{ik} + u), \max_{k \in K_i} \min_{l \in L_j} g_{ik}(w_{jl} - u) \right\} \geq 0$$

is satisfied and, moreover,  $P(u) \subseteq R(0)$  holds if and only if

$$\phi(u) := \min_{i \in I} \min_{j \in J} \max \left\{ \max_{l \in L_j} \min_{k \in K_i} h_{jl}(v_{ik} + u), \max_{k \in K_i} \min_{l \in L_j} g_{ik}(w_{jl} - u) \right\} \geq 0$$

is fulfilled. Since all  $g_{ik}$  and  $h_{jl}$  are affine linear, the terms

$$\min_{k \in K_i} h_{jl}(v_{ik} + u) \quad \text{and} \quad \min_{l \in L_j} g_{ik}(w_{jl} - u)$$

can be analyzed in advance. Setting

$$\tilde{h}_{ijl}(u) := \min_{k \in K_i} h_{jl}(v_{ik} + u) \quad \text{and} \quad \tilde{g}_{ijk}(u) := \min_{l \in L_j} g_{ik}(w_{jl} - u),$$

we obtain

$$\phi_{PR^*}(u) := \min_{i \in I} \min_{j \in J} \max \left\{ \max_{l \in L_j} \tilde{h}_{ijl}(u), \max_{k \in K_i} \tilde{g}_{ijk}(u) \right\}. \quad (12.14)$$

To uniform the description, we further define

$$f_{ijq}(u) := \begin{cases} \tilde{h}_{ijl}(u), & q = l = 1, \dots, |L_j|, \\ \tilde{g}_{ijk}(u), & q - |L_j| = k = 1, \dots, |K_i|, \end{cases} \quad (12.15)$$

for all  $i \in I$  and all  $j \in J$ . Then formula (12.14) can be rewritten as follows:

$$\phi_{PR^*}(u) = \min_{i \in I} \min_{j \in J} \max_{q \in M_{ij}} f_{ijq}(u) \quad (12.16)$$

where  $M_{ij} = \{1, \dots, |L_j| + |K_i|\}$ .

### 12.2.3 Rotation of Objects

In this subsection we allow an arbitrary rotation of pieces. At first let us consider the particular case of convex polygons. According to (12.3)–(12.5), polygon  $P_i$ ,  $i \in I$ , is defined by

$$P_i = \{v \in \mathbb{R}^2 : g_{ik}(v) \leq 0, k \in K_i\} = \bigcap_{k \in K_i} G_{ik} = \text{conv} \{v_{ik} : k \in K_i\}. \quad (12.17)$$

To characterize the mutual position of two polygons  $P_i$  and  $P_j$  the difference vector  $u := u_j - u_i$  of the allocation points (translation vectors) and the rotation angle

$\theta := \theta_j - \theta_i$  are necessary. The restriction on  $u$  and  $\theta$  can be interpreted as follows: polygon  $P_i$  is considered to be in normalized position and only polygon  $P_j$  is rotatable and movable.

We describe the rotation by angle  $\theta$  by means of a *rotation matrix*  $D(\theta)$  and its inverse  $D^{-1}(\theta)$  which are defined as follows:

$$D(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}, \quad D^{-1}(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}.$$

Polygon

$$P_j(u, \theta) := \{v \in \mathbb{R}^2 : v = u + D(\theta)w, w \in P_j\}$$

does not overlap polygon  $P_i(0, 0)$ , i.e.,  $\text{int}(P_i(0, 0)) \cap P_j(u, \theta) = \emptyset$  holds, if and only if  $P_i(0, -\theta)$  and at least a half-plane  $u + G_{jl}$ ,  $l \in K_j$ , or  $P_j(u, \theta)$  and at least a half-plane  $G_{ik}$ ,  $k \in K_i$ , do not overlap. Because of the convexity of  $P_i$  and  $P_j$  this is the case when

$$\exists l \in K_j : \min_{k \in K_i} g_{jl}(D^{-1}(\theta)v_{ik} - u) \geq 0 \quad \text{or} \quad \exists k \in K_i : \min_{l \in K_j} g_{ik}(D(\theta)v_{jl} + u) \geq 0$$

holds. This condition can be transformed into

$$\phi_{ij}(u, \theta) := \max \left\{ \max_{l \in K_j} \min_{k \in K_i} g_{jl}(D^{-1}(\theta)v_{ik} - u), \max_{k \in K_i} \min_{l \in K_j} g_{ik}(D(\theta)v_{jl} + u) \right\} \geq 0.$$

In this way, we obtain a  $\Phi$ -function for the pair of objects  $P_i$  and  $P_j$ ,  $i, j \in I$ ,  $i \neq j$ , setting

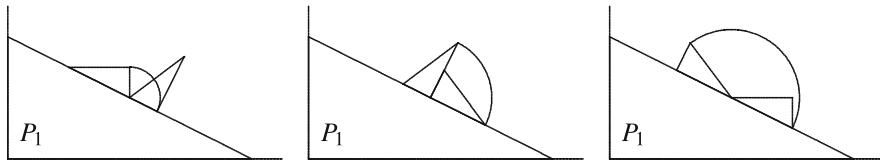
$$\Phi_{ij}(u_i, u_j, \theta_i, \theta_j) := \phi_{ij}(u_j - u_i, \theta_j - \theta_i)$$

in analogy to (12.7). A simplification of function  $\phi_{ij}$  as in (12.8) is possible in a similar way.

Let  $\alpha_{ik} \in [0, 2\pi)$  denote the angle of the gradient (i.e., of the normal vector)  $\nabla g_{ik}$  of  $g_{ik}$ , and let  $\alpha_{jl} \in [0, 2\pi)$  be that of  $\nabla g_{jl}$ . Suppose, corner point  $v_{jl}$  of polygon  $P_j$ , rotated by angle  $\theta$ , yields the minimum  $\min_{l \in K_j} g_{ik}(D(\theta)v_{jl} + u)$ , then there exist two extreme positions such that one of the edges, being in contact with  $v_{jl}$ , lies parallel to the edge of polygon  $P_i$  which belongs to  $g_{ik}$ . Hence, the rotation angle  $\theta$  is contained in

$$\theta \in [\alpha_{ik} - \alpha_{jl} + \pi, \alpha_{ik} - \alpha_{jl-1} + \pi]^*.$$

Here,  $[\alpha, \beta]^*$  indicates the transformation of  $[\alpha, \beta]$  with  $\alpha \neq \beta$  into  $[\alpha + 2\pi a, \beta + 2\pi b]$  with appropriate  $a, b \in \mathbb{Z}$  such that  $0 \leq \alpha + 2\pi a \leq \beta + 2\pi b \leq 2\pi$  holds.



**Fig. 12.9** Rotation ranges in Example 12.5

In a similar way, the containment of a rotated polygon within a polygonal convex region can be modeled by means of  $\Phi$ -functions. A translation of the approach to non-convex objects is possible as well.

*Example 12.5* Consider polygons  $P_1$  and  $P_2$  as given in the table.

| $k$ | $g_{1k}$     | $v_{1k}$ | $\alpha_{1k}$ | $g_{2k}$      | $v_{2k}$   | $\alpha_{2k}$  |
|-----|--------------|----------|---------------|---------------|------------|----------------|
| 1   | $-y$         | $(4, 0)$ | $3\pi/2$      | $-x - 2y + 1$ | $(1, 0)$   | $\alpha + \pi$ |
| 2   | $x + 2y - 4$ | $(0, 2)$ | $\alpha$      | $x - 1$       | $(1, 0.5)$ | 0              |
| 3   | $-x$         | $(0, 0)$ | $\pi$         | $y - 0.5$     | $(0, 0.5)$ | $\pi/2$        |

The resulting angle ranges with respect to the straight line  $g_{12}(x, y) = x + 2y - 4 = 0$  are sketched in Fig. 12.9. Taking  $\alpha = \arctan(2)$ , we obtain

the interval  $[\alpha + \pi, 2\pi]$  for  $v_{21}$ ,  
 the interval  $[\alpha + \pi/2, \alpha + \pi]$  for  $v_{22}$ , and  
 the interval  $[0, \alpha + \pi/2]$  for  $v_{23}$ . ■

A comprehensive description how to construct  $\Phi$ -functions for a pair of non-convex polygons which can be rotated is presented in [35]. Among related optimization problems, the rotational polygon containment problem, e.g. considered in [25], aims to maximize the area utilization when non-convex polygons have to be arranged within a rectangular region.

## 12.2.4 Normalized $\Phi$ -Functions, $\rho$ -Dense Patterns

The general definition of a  $\Phi$ -function ( Definition 1.2 in Sect. 1.5) characterizes the mutual position of two objects qualitatively, but not quantitatively. For that reason so-called *normalized  $\Phi$ -functions* are considered, too, [34].

**Definition 12.1** A  $\Phi$ -function of a pair of objects is called *normalized* if its function value is equal to the Euclidean distance of the objects in case of non-overlapping. By means of a normalized  $\Phi$ -function  $\Phi_{ij}^*$  of two polygons  $P_i$  and  $P_j$  the demand that a minimum distance between the polygons has to hold, can easily be modeled. The minimum distance  $\rho$  ( $\rho \geq 0$ ) between  $P_i(u_i)$  and  $P_j(u_j)$  is guaranteed if  $\Phi_{ij}^*(u_i, u_j) \geq \rho$  holds.

The construction of an efficiently computable normalized  $\Phi$ -function is, in general, very costly since, due to definition,  $\Phi_{ij}^*(u_i, u_j)$  is the optimal value of the minimization problem

$$\min \{ \|w - v\| : v \in P_i(u_i), w \in P_j(u_j) \}$$

where  $\|x\|$  denotes the Euclidean norm of  $x$ . Instead of a normalized  $\Phi$ -function a simple approximation can be used whose function values provide lower bounds for the real distance.

We consider two convex polygons  $P_i$  and  $P_j$ , given in the form (12.17), and we assume that all affine linear functions  $g_{ik}$  and  $g_{jl}$  are given in *Hesse normal form*, that means, for an affine linear function

$$g(x, y) = ax + by + c \quad \text{we have} \quad a^2 + b^2 = 1.$$

The Hesse normal form can be assumed without loss of generality since for any affine linear function  $g(x, y) = ax + by + c$  with  $a^2 + b^2 > 0$  the related Hesse normal form results to  $\bar{g}(x, y) = \bar{a}x + \bar{b}y + \bar{c}$  where

$$\bar{a} := \frac{a}{a^2 + b^2}, \quad \bar{b} := \frac{b}{a^2 + b^2}, \quad \bar{c} := \frac{c}{a^2 + b^2}.$$

We denote the Hesse normal form belonging to  $g_{ik}$  by  $\bar{g}_{ik}$ ,  $k \in K_i$ ,  $i \in I$ . Furthermore, according to (12.6) and (12.7), let the functions

$$\bar{\phi}_{ij}(u) := \max \left\{ \max_{l \in K_j} \min_{k \in K_i} \bar{g}_{jl}(v_{ik} - u), \max_{k \in K_i} \min_{l \in K_j} \bar{g}_{ik}(v_{jl} + u) \right\}$$

and

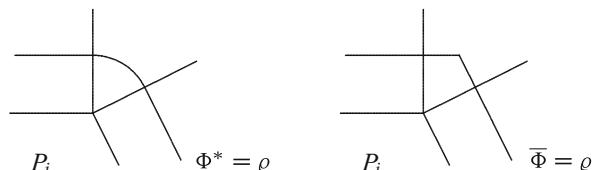
$$\bar{\Phi}_{ij}(u_i, u_j) := \bar{\phi}_{ij}(u_j - u_i)$$

be defined. Then, we have (Exercise 12.7)

$$\bar{\Phi}_{ij}(u_i, u_j) \leq \Phi_{ij}^*(u_i, u_j) \quad \text{for all } u_i, u_j \in \mathbb{R}^2 \text{ with } \bar{\Phi}_{ij}(u_i, u_j) \geq 0. \quad (12.18)$$

However, as becomes clear from Fig. 12.10, often equality holds.

**Fig. 12.10** Normalized  $\Phi$ -function and its approximation



## 12.3 Computation of All Allocation Points

In some circumstances it can be helpful or even necessary to know all possible allocation points. Therefore, we describe a method in the following how to obtain all feasible allocation points for a non-convex polygon  $P$  to fit completely in a region  $R$  having a polygonal frontier.

In analogy to the representation (12.12), let the non-convex polygon  $P$  be given by

$$P = \bigcup_{i \in I} P_i \quad \text{with} \quad P_i = \{v \in \mathbb{R}^2 : g_{ik}(v) \leq 0, k \in K_i\} = \text{conv} \{v_{ik} : k \in K_i\}$$

and region  $R$  as in (12.13). Similar to Sect. 12.2.2, let  $R$  be a subset of a compact convex set  $S$  having a polygonal frontier. Due to

$$R^* := \text{cl}(\mathbb{R}^2 \setminus R) \quad \text{and} \quad R^* = \bigcup_{j \in J} R_j^*,$$

where sets  $R_j^*$  are convex with polygonal frontier, we use the representations

$$R_j^* = \{v \in \mathbb{R}^2 : h_{jl}(v) \leq 0, l \in L_j\} \quad \text{and} \quad R_j^* \cap S = \text{conv} \{w_{jl} : l \in \widetilde{L}_j\}.$$

Set  $U = U(P)$ , containing all translation vectors  $u \in \mathbb{R}^2$  with  $P(u) \subseteq R$ , can be described by

$$U := \{u \in \mathbb{R}^2 : \phi_{PR^*}(u) \geq 0\}$$

because of (12.14) and (12.16). Hence, for a given  $u$  it is easy to verify whether  $P(u) \subseteq R$  holds or not.

The opposite problems *find a single  $u$ , or find all  $u$ , with  $P(u) \subseteq R$* , are much more costly to solve.

### 12.3.1 A Model with Binary Variables

Within this subsection, we describe a method to construct all feasible allocation points for  $P$  to lie in  $R$  which is based on a model possessing only binary variables. According to the structure of  $\phi_{PR^*}$  in (12.16), we define binary variables  $a_{ijq}$ ,  $q \in M_{ij}$ , to identify those functions  $\tilde{h}_{ijl}$  or  $\tilde{g}_{ijk}$  which yield the maximum in

$$\max \left\{ \max_{l \in L_j} \tilde{h}_{ijl}(u), \max_{k \in K_i} \tilde{g}_{ijk}(u) \right\}.$$

In this case,  $a_{ijq}$  gets value 1, otherwise value 0. Hence, any binary solution of the system of inequalities

$$\tilde{h}_{ijl}(u) \geq (a_{ijq} - 1)M, \quad q = l \in L_j,$$

$$\tilde{g}_{ijk}(u) \geq (a_{ijq} - 1)M, \quad q - |L_j| = k \in K_i,$$

$$\sum_{q \in M_{ij}} a_{ijq} \geq 1,$$

fulfills condition  $\phi_{ij}(u) \geq 0$ , and vice versa. Here,  $M$  stands for a sufficiently large constant. If we combine all  $|I| \cdot |J|$  systems of inequalities, then allocation point  $u$  is feasible if there exists a solution of the system of inequalities with binary variables

$$f_{ijq}(u) \geq (a_{ijq} - 1)M, \quad q \in M_{ij}, \quad j \in J, \quad i \in I, \quad (12.19)$$

$$\sum_{q \in M_{ij}} a_{ijq} \geq 1, \quad j \in J, \quad i \in I, \quad (12.20)$$

$$a_{ijq} \in \{0, 1\}, \quad q \in M_{ij}, \quad j \in J, \quad i \in I, \quad u \in \mathbb{R}^2,$$

where function  $f_{ijq}$  is defined in (12.15). Hence, a model with

$$m = \sum_{i \in I} \sum_{j \in J} |M_{ij}| = |I| \sum_{j \in J} |L_j| + |J| \sum_{i \in I} |K_i|$$

binary variables and  $n = m + |I| \cdot |J|$  inequalities is obtained.

If we aim to find all allocation points, then we need to construct all combinations of binary variables which fulfill (12.20) and possess a non-empty set of solutions of (12.19).

### 12.3.2 The Basic Algorithm

Here we describe a simple method to construct all combinations of the  $a_{ijq}$ -variables. In the basic algorithm (Fig. 12.11) one of the  $|M_{ij}|$  inequalities is chosen for each  $i \in I$  and each  $j \in J$ . Hence, altogether

$$\prod_{i \in I} \prod_{j \in J} (|K_i| + |L_j|) \quad (12.21)$$

different possibilities result. Since most of these combinations lead to inconsistent systems of inequalities (12.19), an appropriate sequence of fixing variables has to

**Basic algorithm for computing all allocation points**

- (1) Set  $K := \{(i_1, j_1)\} := \{(1, 1)\}$ ,  $k := 1$ .
- (2) Choose some inequality for  $(i_k, j_k)$ , say  $f_k(u) := f_{i_k, j_k, q}(u) \geq 0$  with  $q \in M_{i_k j_k}$  and define  $V_k := \{u \in \mathbb{R}^2 : f_t(u) \geq 0, t = 1, \dots, k\}$ .  
If  $V_k = \emptyset$  (the system is inconsistent), then go to step (4).
- (3) If  $K = I \times J$ , i.e., we have  $k = |I| \cdot |J|$ , then a set of feasible allocation points is found: go to step (4). Otherwise, set  $k := k + 1$ , extend  $K$  by a further index pair  $(i_k, j_k)$ , and go to step (2).
- (4) If still possible, choose the next inequality for  $(i_k, j_k)$  and go to step (2). Otherwise, remove  $(i_k, j_k)$  from  $K$  and set  $k := k - 1$ . If  $K$  is empty, i.e.,  $k = 0$ , then stop, otherwise repeat step (4).

**Fig. 12.11** Basic algorithm to obtain all allocation points

be applied. To identify inequalities which are relevant for all  $P_i$ , or for most of them, we consider at first those index pairs  $(i, j)$  with small  $|M_{ij}|$ , and at last those sets with large cardinality. Within the basic algorithm, presented in Fig. 12.11, set  $K$  contains index pairs for which an inequality in (12.19) is non-trivial.

Obviously, if  $|L_j| = 1$  for some  $j \in J$ , then a corresponding inequality has to hold in any case. Therefore, the convex hull  $\text{conv}(R)$  of  $R$  leads to inequalities which are fulfilled for each feasible allocation point of  $P$ .

If, in the branching process, several  $l \in L_j$  do not reduce set  $V_{k-1}$  for some stage  $k$ , i.e., if we have  $V_k = V_{k-1}$ , then only one of them has to be considered within the branching tree since the others lead to the same allocation points (if any exists).

In order to obtain disjunctive sets of allocation points, the following branching scheme can be applied in stage  $k$  (for short, let  $(i, j) = (i_k, j_k)$ ):

- 1. subproblem,  $q = 1$ :  $f_{ij1}(u) \geq 0$ , i.e.,  $a_{ij1} := 1$ ,
- 2. subproblem,  $q = 2$ :  $f_{ij1}(u) < 0, f_{ij2}(u) \geq 0$ , i.e.,  $a_{ij1} := 0, a_{ij2} := 1$ ,
- ...
- (last) subproblem,  $q = |K_i| + |L_j|$ :  
 $f_{ij1}(u) < 0, \dots, f_{i,j,q-1}(u) < 0, f_{ijq}(u) \geq 0$ , i.e.,  $a_{ij1} := 0, \dots, a_{ij,q-1} := 0, a_{ijq} := 1$ .

By means of this branching scheme, the total effort can be reduced essentially.

### 12.3.3 The Rectangular Case

In this subsection we assume that all  $P_i$  are axes-parallel rectangles and that all  $R_j^*$  have rectangular shape (not necessarily bounded). With other words, all equations  $g_{ik}(v) = 0$  and  $h_{jl}(v) = 0$  define axes-parallel straight lines. Moreover, we have  $|K_i| = 4$  for all  $i \in I$  and  $1 \leq |L_j| \leq 4$  for all  $j \in J$ .

In this particular situation the definition of  $\phi_{ij}$  for  $R_j^*(0)$  and  $P_i(u)$  can be simplified (see Exercise 12.8) to

$$\phi_{ij}(u) := \max_{l \in L_j} h_{jl}(v_{ik_l} + u) \quad \text{where} \quad h_{jl}(v_{ik_l} + u) := \min_{k \in K_i} h_{jl}(v_{ik} + u).$$

Therefore, we have

$$\phi_i(u) = \min_{j \in J} \phi_{ij}(u) = \min_{j \in J} \max_{l \in L_j} h_{jl}(v_{ik_l} + u)$$

and

$$\phi(u) = \min_{i \in I} \phi_i(u) = \min_{i \in I} \min_{j \in J} \max_{l \in L_j} h_{jl}(v_{ik_l} + u). \quad (12.22)$$

If we use, in analogy to above, functions

$$f_{ijl}(u) := h_{jl}(v_{ik_l} + u) := \min_{k \in K_i} h_{jl}(v_{ik} + u)$$

which are affine linear as well, then we obtain

$$\phi(u) = \min_{i \in I} \min_{j \in J} \max_{l \in L_j} f_{ijl}(u).$$

The definition of  $\phi$  in (12.22) using the  $h_{jl}$ -functions is favorable in comparison to a definition by means of functions  $g_{ik}$  since  $|L_j| \leq |K_i|$  for all  $i \in I$  and  $j \in J$  holds. In comparison to the total number of combinations of binary variables, as given in (12.21), in the current case it is remarkably smaller since in  $h_{jl}$  always exactly two of the four corner points of  $P_i$  yield the minimum.

### 12.3.4 Example

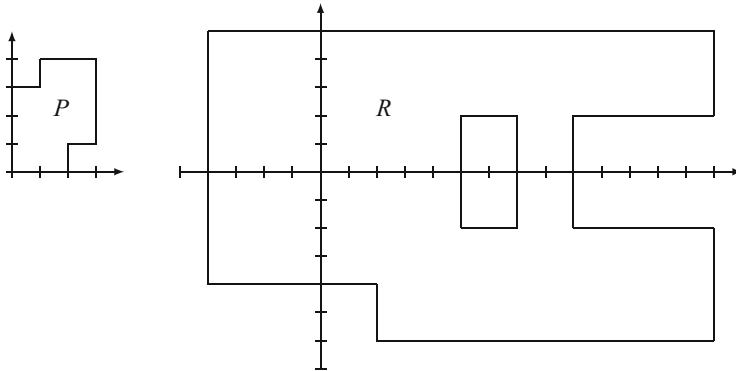
Consider a non-convex polygon  $P = P_1 \cup P_2$  given as union of convex polygons  $P_i = \{(x, y) : g_{ik}(x, y) \leq 0, k \in K_i\}$  with  $K_i = \{1, \dots, 4\}$  for all  $i \in I = \{1, 2\}$ . The related affine linear functions are given in Table 12.1. Region  $R$ , which contains a hole, is given by convex, axes-parallel regions  $R_j^* = \{(x, y) : h_{jl}(x, y) \leq 0, l \in L_j\}$ ,  $j \in J = \{1, \dots, 7\}$ . Functions  $h_{jl}$  are listed in Table 12.2. To simplify the

**Table 12.1** Input data of non-convex polygon  $P$

| $g_{ik}$ | $k = 1$ | $k = 2$ | $k = 3$ | $k = 4$ |
|----------|---------|---------|---------|---------|
| $i = 1$  | $-x$    | $-y$    | $x - 2$ | $y - 3$ |
| $i = 2$  | $1 - x$ | $1 - y$ | $x - 3$ | $y - 4$ |

**Table 12.2** Input data of non-convex regions  $R$ 

| $h_{jl}$ | $j = 1$  | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ | $j = 6$  | $j = 7$  |
|----------|----------|---------|---------|---------|---------|----------|----------|
| $l = 1$  | $14 - x$ | $5 - y$ | $x + 4$ | $y + 6$ | $x - 2$ | $9 - x$  | $5 - x$  |
| $l = 2$  |          |         |         |         | $y + 4$ | $y - 2$  | $y - 2$  |
| $l = 3$  |          |         |         |         |         | $-y - 2$ | $-y - 2$ |
| $l = 4$  |          |         |         |         |         |          | $x - 7$  |

**Fig. 12.12** Polygon  $P$  and region  $R$ **Table 12.3** Functions  $f_{ijl}$  for  $j = 1, \dots, 4$  and  $i \in \{1, 2\}$ 

|         | $j = 1$  | $j = 2$ | $j = 3$ | $j = 4$ |
|---------|----------|---------|---------|---------|
| $i = 1$ | $12 - x$ | $2 - y$ | $x + 4$ | $6 + y$ |
| $i = 2$ | $11 - x$ | $1 - y$ | $x + 5$ | $7 + y$ |

presentation, we assume in the following that the convex sets  $R_j^*$  are sorted according to increasing cardinality of  $L_j$ . Polygon  $P$  and region  $R$  are shown in Fig. 12.12.

In this example, altogether we have  $24^2 = 576$  combinations of binary variables such that for each  $i \in I$  and each  $j \in J$  exactly one has value 1. Corner points of  $P_1$  and  $P_2$  are the following:

$$v_{1k} : (0, 0), (2, 0), (2, 3), (0, 3), \quad v_{2k} : (1, 1), (3, 1), (3, 4), (1, 4).$$

Hereby we obtain functions  $f_{ijl} = \tilde{h}_{ijl}(x, y) = \min_{k \in K_i} h_{jl}(v_{ik} + (x, y))$  for  $j = 1, \dots, 4$ , as given in Table 12.3, where always  $l = 1$  holds. Obviously, four of the eight conditions are dominated and we obtain

$$\begin{aligned} U \subset V_8 &:= \{u \in \mathbb{R}^2 : \tilde{h}_{ij1}(u) \geq 0, i \in \{1, 2\}, j \in \{1, 2, 3, 4\}\} \\ &= \{(x, y) : -4 \leq x \leq 11, -6 \leq y \leq 1\} \\ &= [-4, 11] \times [-6, 1]. \end{aligned}$$

**Table 12.4** Functions  $f_{ijl}$  for  $j \geq 5$  and  $i \in \{1, 2\}$ 

|         | $j = 5$ | $j = 5$ | $j = 6$ | $j = 6$ | $j = 6$  | $j = 7$ | $j = 7$ | $j = 7$  | $j = 7$ |
|---------|---------|---------|---------|---------|----------|---------|---------|----------|---------|
|         | $l = 1$ | $l = 2$ | $l = 1$ | $l = 2$ | $l = 3$  | $l = 1$ | $l = 2$ | $l = 3$  | $l = 4$ |
| $i = 1$ | $x - 2$ | $y + 4$ | $7 - x$ | $y - 2$ | $-y - 5$ | $3 - x$ | $y - 2$ | $-y - 5$ | $x - 7$ |
| $i = 2$ | $x - 1$ | $y + 5$ | $6 - x$ | $y - 1$ | $-y - 6$ | $2 - x$ | $y - 1$ | $-y - 6$ | $x - 6$ |

For  $j \geq 5$ , the respective  $f_{ijl}$ -functions are listed in Table 12.4. Next, we consider  $R_5^*$  and  $R_6^*$  and obtain

$$\begin{aligned} V_9 &= [2, 11] \times [-6, 1] \text{ with } a_{151} = 1 \quad (\text{i.e., } x \geq 2), \\ V_{10} &= [2, 11] \times [-6, 1] \text{ with } a_{251} := 1 \quad (\text{i.e., } x \geq 1), \\ V_{11} &= [2, 7] \times [-6, 1] \text{ with } a_{161} := 1 \quad (\text{i.e., } x \leq 7), \\ V_{12} &= [2, 6] \times [-6, 1] \text{ with } a_{261} := 1 \quad (\text{i.e., } x \leq 6). \end{aligned}$$

If we now choose  $a_{171} = 1$  (i.e.,  $x \leq 3$ ), then

$$V_{13} = [2, 3] \times [-6, 1]$$

results. With  $a_{271} = 1$  (i.e.,  $x \leq 2$ ), we find the first set of allocation points:

$$U_1 := V_{14} = [2, 2] \times [-6, 1].$$

Applying the branching scheme of page 364, we obtain with  $a_{272} := 1$  and  $a_{271} := 0$  the inequalities  $y \geq 1$  and  $x > 2$ , and therefore,

$$U_2 = V_{14} = (2, 3] \times [1, 1].$$

With  $a_{273} := 1$ ,  $a_{272} := 0$ , and  $a_{271} := 0$ , the inequalities  $y \leq -6$ ,  $y < 1$ , and  $x > 2$  have to be considered, and we obtain

$$U_3 = V_{14} = (2, 3] \times [-6, -6].$$

Conditions  $a_{274} := 1$  (i.e.,  $x \geq 6$ ),  $a_{273} := 0$ ,  $a_{272} := 0$ , and  $a_{271} := 0$  lead to the empty set. The case  $a_{172} := 1$ ,  $a_{171} := 0$  (i.e.,  $y \geq 2$ ,  $x > 3$ ) also leads to the empty set. For  $a_{173} := 1$ ,  $a_{172} := 0$ ,  $a_{171} := 0$  (i.e.,  $y \leq -5$ ,  $y < 2$ ,  $x > 3$ ), we obtain

$$V_{13} = (3, 6] \times [-6, -5].$$

The choice of  $a_{271} := 1$  (i.e.,  $x \leq 2$ ) or  $a_{272} := 1$  (i.e.,  $y \geq 1$ ) leads to  $V_{14} = \emptyset$ , i.e., no allocation point can be found for this (particular) combination. In the case  $a_{273} := 1$ ,  $a_{272} := 0$ ,  $a_{271} := 0$  (i.e.,  $y \leq -6$ ,  $x > 2$ ,  $y < 1$ ), we obtain the next set of allocation points:

$$U_4 := V_{14} = (3, 6] \times [-6, -6].$$

According to the proposed branching scheme, because of  $a_{274} := 1$ ,  $a_{273} := 0$ ,  $a_{272} := 0$ ,  $a_{271} := 0$ , the inequalities  $x \geq 6$ ,  $y > -6$ ,  $y < 1$ ,  $x > -1$  become relevant, and therefore,

$$U_5 = V_{14} = [6, 6] \times (-6, -5]$$

results. Using  $a_{174} := 1$ ,  $a_{173} := 0$ ,  $a_{172} := 0$ ,  $a_{171} := 0$  (i.e.,  $x \geq 7$ ,  $y > -5$ ,  $y < 2$ ,  $x > 3$ ),  $V_{13} = \emptyset$  follows. The choice  $a_{262} := 1$ ,  $a_{261} := 0$  (i.e.,  $y \geq 1$ ,  $x > 6$ ) instead of  $a_{261} := 1$  leads to  $y = 1$  and

$$V_{12} = (6, 7] \times [1, 1].$$

Only for  $a_{174} := 1$  (i.e.,  $x \geq 7$ ) a non-empty set results which contains the isolated point  $(7, 1)$ :

$$V_{13} = [7, 7] \times [1, 1].$$

With  $a_{272} := 1$  (i.e.,  $y \geq 1$ ), we find the isolated allocation point

$$U_6 = \{(7, 1)\}.$$

In the case  $a_{274} := 1$  (i.e.,  $x \geq 6$ ,  $y < 1$ ) the empty set results again. The choice  $a_{263} := 1$  (i.e.,  $y \leq -6$ ,  $y < 1$ ,  $x > 6$ ) leads to  $y = -6$  and

$$V_{12} = (6, 7] \times [-6, -6].$$

For  $a_{173} := 1$  (i.e.,  $y \leq -5$ ), and  $a_{273} := 1$  (i.e.,  $y \leq -6$ ), we obtain a further set of allocation points:

$$U_7 = V_{14} = (6, 7] \times [-6, -6].$$

In the case  $a_{162} := 1$  (i.e.,  $y \geq 2$ ,  $x > 7$ ) a contradiction appears. Conditions  $a_{163} := 1$ ,  $a_{162} := 0$ , and  $a_{161} := 0$  (i.e.,  $y \leq -5$ ,  $y < 2$ ,  $x > 7$ ) enforce a reduction of the set of potential allocation points to

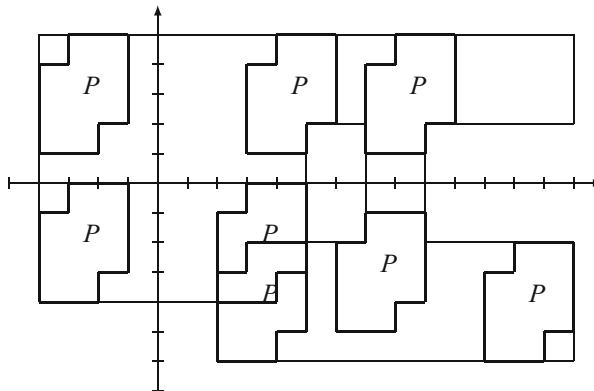
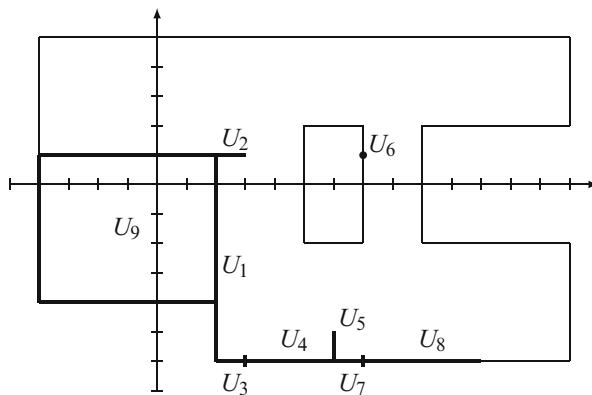
$$V_{11} = (7, 11] \times [-6, -5].$$

With  $a_{263} := 1$ ,  $a_{173} := 1$ , and  $a_{273} := 1$  (i.e.,  $y \leq -6$ ,  $y \leq -5$ ,  $y \leq -6$ ), we obtain

$$U_8 = (7, 11] \times [-6, -6].$$

This process can be continued until all  $24^2 = 576$  possible combinations are analyzed. Finally, we find

$$U_9 = [-4, 2) \times [-4, 1].$$

**Fig. 12.13** Some possible placements**Fig. 12.14** Set of all allocation points

Some of the possible allocations of piece  $P$  within region  $R$  are drawn in Fig. 12.13. The set of all allocation points  $u$  with  $P(u) \subset R$  is depicted in Fig. 12.14.

## 12.4 Algorithms for Strip Packing Problems

As we have seen in the previous section, describing the interaction of polygons by means of a mixed-integer linear program requires a very large number of binary variables which makes an exact solution too costly, in general. Therefore, MILP models are used for local optimization as, for instance, in [16, 37]. A drastic simplification of the MILP model is derived by fixing most of (or even all) binary variables, that means, the mutual position of the objects remains unchanged, only

translations are considered. Such local optimization requires good starting solutions (patterns). For that reason, and because of their general importance, we now turn to heuristic approaches.

To this end, we consider again the problem of packing polygonal pieces (polygons)  $P_i$ ,  $i \in I$ , within a strip  $R$  of given width  $W$  and minimal height  $H$ . To construct a pattern with a small height, i.e., to approximately solve the minimization problem, there exists a diversity of approaches. On the one hand, we can differentiate between deterministic methods and those which incorporate stochastic elements. On the other hand, essential differences are present concerning the packing strategy. There are approaches which place the objects sequentially, and others handle all pieces simultaneously. Within sequential approaches sometimes holes, arising during the packing process, are regarded, but not in others.

In the following we give some overview on heuristic approaches without making a claim of completeness.

### 12.4.1 Sequential Approaches

In sequential heuristics, the pieces are placed one after another according to a particular packing strategy. Thereby, the already placed pieces  $P_j(u_j)$ ,  $j = 1, \dots, i-1$ , are considered to be immovable. For polygon  $P_i$ , to be placed next, a suitable allocation point  $u_i = (x_i, y_i)$  has to be found according to the chosen strategy. Hence, in each step a containment problem has to be solved in which some appropriate target function is used, for instance, the  $y$ -coordinate of the allocation point is minimized.

In many algorithms a dense packing of  $P_i$  on the *contour*  $\kappa_{i-1}$  is realized. The contour (also named *profile* or *skyline*) is defined by

$$\kappa_i(x) := \max \left\{ 0, \max \left\{ y : (x, y) \in \bigcup_{j=1}^i P_j(u_j) \right\} \right\}, \quad x \in [0, W].$$

The contour partially represents the hodograph of  $P_i$  and  $\bigcup_{j=1}^{i-1} P_j(u_j) \cup R^*$ .

Within heuristics having sequential packing strategy the following principles are regarded with varying order:

- choose that piece as piece  $i$  (to be placed next) whose allocation leads to a (new) hodograph with minimal extension in  $y$ -direction;
- place piece  $i$  such that the waste (unusable area) induced by  $i$  is minimal;
- choose a piece with largest area first;
- choose a piece with minimal  $y$ -coordinate of its allocation point.

Heuristics, e.g. the BL algorithm, which apply a predefined sequence of allocating the polygons work, in general, faster than those where in each step the next

piece has to be selected. Therefore, to reduce the expenses for computing the next allocation point, frequently some discretizations are applied. In a first variant, a lattice with a given step-size in  $x$ - and  $y$ -direction is defined in advance. Then, only lattice points are accepted as allocation points. In a second variant, only the  $y$ -direction is discretized (with step-size  $\Delta y$ ), that means, the allocation point is looked for on lines  $y = k \cdot \Delta y$ ,  $k = 0, 1, 2, \dots$ . If a  $y$ -coordinate for a feasible allocation point is found, then, applying some suitable criterion, the  $x$ -coordinate is determined.

In Art's algorithm [3, 7] piece  $P_i$  to be placed next has to be selected and the allocation point  $u_i = (x_i, y_i)$  has to lie on the hodograph of  $P_i$  and  $R^* \cup \bigcup_{j=1}^{i-1} P_j(u_j)$ . In this heuristic, holes are not considered for a possible packing of another subsequent piece. The packing strategy can be seen as shifting the piece in negative  $y$ -direction, starting from a sufficiently high and appropriately chosen horizontal position, until a first contact with an already placed object or the strip border occurs. The area below the contour minus the total area of the already packed pieces determines the waste of the partial packing.

Within the algorithm of Albano and Supuppo (for details we refer to [1, 7]) discrete rotations, induced by a given step-size for the rotation angle, are additionally regarded.

The heuristics proposed by Dowsland and Dowsland [13] also use lattice points as potential allocation points. Moreover, if possible, polygons are placed into holes, i.e., into unused regions below the contour.

A translation technique is proposed in [2]. There, the directions of translation are determined by the current contour.

Another heuristic, particularly designed for placement problems of the textile industry, is presented in [28]. At first the larger pieces are placed and after that the smaller objects are packed into the holes, if possible.

Due to the approach of Heckmann and Lengauer [20] the pieces are packed according to a predefined sequence. In each step, a random orientation (rotation angle) is chosen for the piece to be placed next. Then, for the rotated piece  $i$  and each already placed object (say  $k$ ), and for  $i$  and the strip, pairwise point sets are identified such that  $i$  does not overlap  $k$ . Obviously, any point of the intersection of these sets indicates a feasible allocation point for  $i$ . Among all of these points, an allocation point for  $i$  is chosen according to some appropriate criterion.

### 12.4.2 Improvement Methods

Improvement methods are those which try to improve a known feasible or infeasible pattern. Some of these approaches assume an initial feasible arrangement, whereas other methods allow in the starting pattern some overlap of pieces or a non-complete containment within the strip. Investigations in this respect can be found, for instance, in [13, 22, 36, 37]. Generally, small changes of the mutual positions of the pieces are performed within each step of the heuristic.

Within the so-called *compaction algorithms* [22, 28], the influence of (gravity) forces onto a pattern is simulated and, in this way, a compaction of the pattern or an enlargement of holes, maintaining the strip height, is aimed for.

Improved patterns can also be obtained by applying *Linear Programming* approaches. Based on the  $\Phi$ -functions, *active restrictions* (those which represent the contact of two pieces, or of a piece with the strip border) can be identified. Maintaining these equations, a simultaneous translation of all pieces can be realized and, possibly, an improved pattern can be obtained. Investigations in this respect are presented, for instance, in [36, 37].

### 12.4.3 Metaheuristics

The availability of fast heuristics enables to apply more general principles to search for good (near-optimal) solutions of optimization problems which cannot be handled efficiently with common solution methods. In particular, heuristics which apply a predefined sequence to pack the pieces are used in metaheuristic approaches (Sect. 7.4.1).

The application of *Simulated Annealing* is investigated, for instance, in [10, 12, 23, 24, 31]. In the latter, rotation of objects is allowed. Moreover, a *Tabu Search* algorithm is proposed in [8].

Within the *Sequential Value Correction* method, which is applied in [32] for the strip packing problem, the order of allocating the polygons is determined deterministically. To this end, based on the last generated pattern, pseudo-values for the pieces are computed defining the new sequence in which the pieces are packed anew.

### 12.4.4 Further Aspects

Within the various different strategies, nonlinear programming approaches are also considered as, for instance, in [21]. Moreover, the solution of non-standard packing problems by global optimization is addressed in [15].

Besides an approximation of real objects by polygons, frequently representations of objects by means of straight-line segments and circular arcs are meaningful. A related sequential algorithm can be found, for instance, in [9]. Moreover, the optimal clustering of two non-regular objects is addressed in [6] based on that kind of object representation.

Due to the large plurality of different approaches, in this chapter we focused ourselves on the modeling and basic principles to (heuristically) solve the packing problems. In dependence on the real-world scenario it is to decide which of the methods is most suitable. Naturally, reviewing articles, as for instance [13, 14], are helpful in this respect.

Note that further aspects, besides the choice of the method, are important for an efficient implementation of a solution method. For instance, the representation of the polygons in the computer (e.g. as a double-linked chain of edges) has importance as well as the possibility of a *local* consideration of the hodograph or the respective  $\Phi$ -functions.

## 12.5 Exercises

**Exercise 12.1** Let three convex polygons  $P_i$  be defined by their corner points:  $v_{1k}$ : (0,0), (2,0), (0,2);  $v_{2k}$ : (0,0), (1,0), (0,1);  $v_{3k}$ : (1,0), (1,1), (0,1). Compute a  $\Phi$ -function for each pair of objects and draw the corresponding hodograph or rather the respective no-fit polygon.

**Exercise 12.2** Formulate an algorithm to check a mutual overlap for that case when a polygon  $P_j(u_j)$  is shifted in direction  $d \in \mathbb{R}^2$  and possibly intersects  $P_i(u_i)$ . Which length of translation leads to a first contact?

**Exercise 12.3** Construct an instance with convex polygons for which the region  $\Omega$  of feasible allocation points has an exponential number of connected components.

**Exercise 12.4** Show, by means of an example, that the following presumption is wrong, in general: if all corner points of the polygons have integer coordinates and all angles of their edges are an integer multiple of  $45^\circ$ , then the coordinates of each allocation point are integer in a bottom-left-justified pattern.

**Exercise 12.5** Compute a rectangle with minimal area which envelopes a given polygon.

**Exercise 12.6** Compute the allocation points of all pieces of Example 12.4 (page 353) for an arbitrary sequence if the BL strategy is applied.

**Exercise 12.7** Show the validity (correctness) of formula (12.18) on page 361.

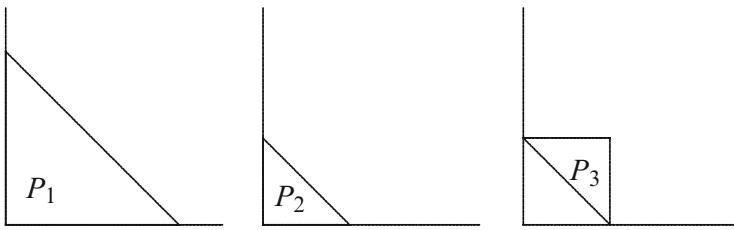
**Exercise 12.8** Show the validity of the simplified form of  $\phi_{ij}$  and  $\overline{U}_{ij}$  in Proposition 12.1 (page 351) for a pair of rectangles.

**Exercise 12.9** Compute an area-maximal axes-parallel rectangle which fits within a given convex polygon.

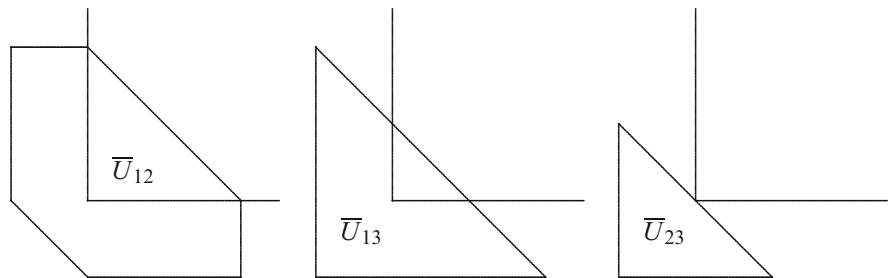
## 12.6 Solutions

**To Exercise 12.1** The three polygons are shown in Fig. 12.15. As  $\phi$ -functions we obtain

$$\begin{aligned}\phi_{12} &= \max\{x + y - 2, -x - 1, -y - 1, -x - y - 1, x - 2, y - 2\}, \\ \phi_{13} &= \max\{x + y - 1, -x - 1, -y - 1\}, \\ \phi_{23} &= \max\{x + y, -x - 1, -y - 1\}.\end{aligned}$$



**Fig. 12.15** Polygons of Exercise 12.1



**Fig. 12.16** No-fit polygons of Exercise 12.1

The no-fit polygons, and thereby the hodographs, are drawn in Fig. 12.16.

**To Exercise 12.2** The simple approach, described below, consists of two steps.

1. At first we verify whether a collision appears at all.

To this end, we compute strips  $S_i$  and  $S_j$  with minimal width parallel to direction  $d$  in which  $P_i(u_i)$  and  $P_j(u_j)$  are completely contained, respectively. Let  $a$  denote a vector perpendicular (orthogonal) to  $d$ . Then

$$S_i = \{u \in \mathbb{R}^2 : \min_{k \in K_i} a^\top (v_{ik} + u_i) \leq a^\top u \leq \max_{k \in K_i} a^\top (v_{ik} + u_i)\},$$

$$S_j = \{u \in \mathbb{R}^2 : \min_{k \in K_j} a^\top (v_{jk} + u_j) \leq a^\top u \leq \max_{k \in K_j} a^\top (v_{jk} + u_j)\}.$$

If  $\text{int}(S_i \cap S_j) = \emptyset$ , then collision does not appear.

2. In the case that a collision can appear, we define  $S := S_i \cap S_j = \{u \in \mathbb{R}^2 : \alpha \leq a^\top u \leq \beta\}$  where

$$\alpha := \max\{\min_{k \in K_i} a^\top (v_{ik} + u_i), \min_{k \in K_j} a^\top (v_{jk} + u_j)\}$$

and

$$\beta := \min\{\max_{k \in K_i} a^\top (v_{ik} + u_i), \max_{k \in K_j} a^\top (v_{jk} + u_j)\}.$$

For each corner point  $v_{ik}$  of  $P_i$  with  $a^\top(v_{ik} + u_i) \in [\alpha, \beta]$  and each corner point  $v_{jk}$  of  $P_j$  with  $a^\top(v_{jk} + u_j) \in [\alpha, \beta]$  the maximal length of translation until collision appears the first time, has to be computed. Then, the resulting minimal value yields the maximal feasible length of translation.

It is advisable for this approach to consider the edges of  $P_i(u_i)$  and  $P_j(u_j)$  in opposite direction. The particular case where the normal vectors of some edge  $g_i$  and some edge  $g_j$  of  $P_i(u_i)$  and  $P_j(u_j)$ , respectively, are orthogonal to  $d$  with different orientation and where additionally the straight lines belonging to  $g_i$  and  $g_j$  coincide, requires some separate investigations.

**To Exercise 12.3** Let  $m \in \mathbb{N}$  and  $W > 0$  be given. We choose identical triangles as pieces  $T_i$ ,  $i = 1, \dots, m$ , which are defined by their corner points  $(0, 0)$ ,  $(0, 1)$  and  $(W, i)$ . Then the argumentation that an exponential number of connected components of  $\Omega$  appears is similar to that in Example 12.4.

**To Exercise 12.4** We show by means of an example: although all input data are integers, there exists an optimal pattern having allocation points with y-coordinate  $(2k)^{-1}$  for  $k = 1, 2, \dots$ . Let convex polygons  $P_i$  be defined by their corner points:

- $P_1$ :  $(0,0), (1,0), (1,1), (0,1)$
- $P_2$ :  $(1,0), (2,1), (1,2), (0,1)$
- $P_3$ :  $(1,0), (2,0), (3,1), (0,1)$
- $P_4$ :  $(0,0), (3,0), (2,1), (1,1)$

We define a sequence of instances by

$$E_k : \quad W = 6k, \quad b^k = (2k+2, 2k, k, k)^\top, \quad k = 1, 2, \dots,$$

where  $b_i^k$  denotes the number of polygons of type  $i$  which have to be placed. The patterns shown in Fig. 12.17 possess the minimal height

$$H_k^* = 2 + \frac{1}{2k}.$$

Note that a dense packing with height 2 in the form

- $(k+1)$  times two polygons of type 1 (on top of each other),
- $k$  times polygons of type 2, and
- $k$  times alternating a polygon of type 3 above a polygon of type 4 followed by a polygon of type 2,

requires a strip width of  $6k+1$  units.

**To Exercise 12.5** Without loss of generality, we can assume that the polygon to be enclosed is convex. Moreover, for computing a rectangle with minimal area, we can further assume that a situation as shown in Fig. 12.18 is present: the rectangle is determined by four corner points of the polygon which are denoted by  $v_1, \dots, v_4$ .

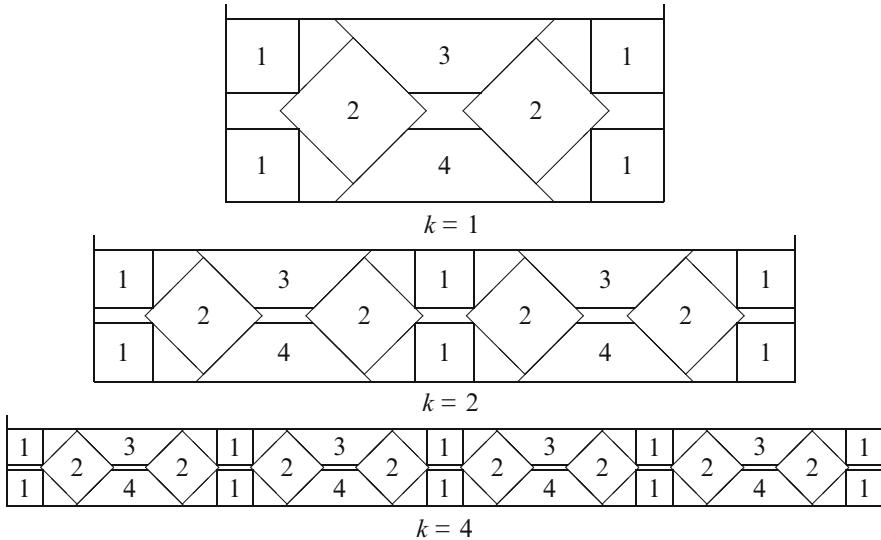


Fig. 12.17 Examples of Exercise 12.4

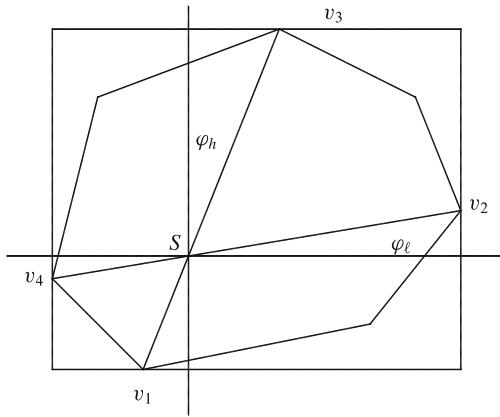


Fig. 12.18 Example-polygon of Exercise 12.5

Then, the polygon can be rotated with rotation center  $S$  where  $S$  is the intersection point of the two straight lines connecting  $v_1$ ,  $v_3$  and  $v_2$ ,  $v_4$ , respectively. The range of the rotation angle  $\varphi$  is limited such that the same four corner points of the polygon remain determining the enveloping axes-parallel rectangle. Therefore, we can assume  $\varphi \in [\alpha, \beta]$  with  $\alpha \leq 0 \leq \beta$  with some appropriate  $\alpha$  and  $\beta$ . Furthermore, let  $\varphi_h$  denote the angle (in mathematical positive sense, i.e., counter-clockwise) between the vertical axis and the straight line defined by  $v_1$  and  $v_3$ . Similarly, let  $\varphi_\ell$  denote that angle between the horizontal axis and the straight line going through

$v_2$  and  $v_4$ . Thus, we have  $\varphi_h < 0$  and  $\varphi_\ell > 0$  in Fig. 12.18. Then, obviously,

$$|\varphi_h| + |\varphi_\ell| \leq \pi/2$$

holds. According to  $d_1 := \|v_3 - v_1\|$  and  $d_2 := \|v_4 - v_2\|$ , we obtain the area  $A(\varphi)$  as product of length and width in dependence on the rotation angle:

$$A(\varphi) = d_1 \cos(\varphi_h + \varphi) \cdot d_2 \cos(\varphi_\ell + \varphi).$$

To show that function  $A(\varphi)$  is piecewise concave, we compute the first derivative:

$$A'(\varphi) = -d_1 d_2 \{\sin(\varphi_h + \varphi) \cos(\varphi_\ell + \varphi) + \cos(\varphi_h + \varphi) \sin(\varphi_\ell + \varphi)\}.$$

Using an appropriate addition theorem, we have

$$A'(\varphi) = -d_1 d_2 \sin(\varphi_h + \varphi_\ell + 2\varphi)$$

and, therefore,

$$A''(\varphi) = -2d_1 d_2 \cos(\varphi_h + \varphi_\ell + 2\varphi).$$

Thus, in case of  $|\varphi_h + \varphi_\ell| < \pi/2$ ,  $A''(\varphi) < 0$  is satisfied for angle  $\varphi = 0$ . Hence, function  $A(\varphi)$  is locally concave. Therefore, local minima appear on the interval limits  $\alpha$  and  $\beta$ . However, these particular angles are determined in such way that at least one edge of the polygon is axes-parallel. In that case, a change of the set of corner points arises which define the enveloping rectangle.

As a consequence we have: an area-minimal enveloping rectangle can be obtained if successively each edge of the polygon is taken as a basic side of an enveloping rectangle.

Note that the case  $|\varphi_h + \varphi_\ell| = \pi/2$  can be considered in an analogous manner. It yields the same result.

**To Exercise 12.6** Let  $\pi = (\pi_0, \dots, \pi_m)$  be a permutation of  $0, \dots, m$ . We define by

$$a_0 := 0, \quad a_i := \begin{cases} a_{i-1}, & \text{if } \pi_i < \pi_{i-1}, \\ a_{i-1} + 1, & \text{if } \pi_i > \pi_{i-1}, \end{cases} \quad i = 1, \dots, m,$$

how often it happens that a pieces  $k$  with  $\pi_k < \pi_{k-1}$  is placed, leading to some additional height requirement, until piece  $\pi_i$  is placed. Then, the allocation points obtained by the BL heuristic are as follows:

$$u_i = (0, i + a_i), \quad i = 0, 1, \dots, m.$$

The corresponding strip height equals  $BL(\pi) = m + 2 + a_m$ .

**To Exercise 12.7** Without loss of generality, we consider the distance of a single point  $u = (x, y)$  to the corner point  $v_{ik} = (x_{ik}, y_{ik})$  of  $P_i$ . Moreover, let

$$g_{ik}(u) = \rho \quad \text{with } \rho > 0,$$

where  $g_{ik}$  is given in Hesse normal form. Therefore,

$$g_{ik}(x, y) = ax + by + c \quad \text{with} \quad a^2 + b^2 = 1$$

holds as well as

$$g_{ik}(v_{ik}) = 0 \quad \text{and} \quad c = -(ax_{ik} + by_{ik}).$$

Applying Schwarz inequality, due to  $\sqrt{a^2 + b^2} = 1$ ,

$$\rho = g_{ik}(u) = ax + by - ax_{ik} - by_{ik} = \begin{pmatrix} x - x_{ik} \\ y - y_{ik} \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} \leq \|u - v_{ik}\|$$

follows. Hence, we have: if  $u$  possesses a distance to  $P_i$  not less than  $\rho$  with respect to  $\overline{\Phi}_{ij}$ , then the Euclidean distance is not smaller than  $\rho$ .

**To Exercise 12.8** We consider the term (relation)  $\phi_{ij}(u) = \max_{k \in K_i} \min_{l \in K_j} g_{ik}(v_{jl} + u)$ . The corner points of rectangles  $P_j$  are  $v_{j1} = (0, 0)$ ,  $v_{j2} = (\ell_j, 0)$ ,  $v_{j3} = (\ell_j, w_j)$ , and  $v_{j4} = (0, w_j)$ . Then, we have:

$$\begin{aligned} g_{i1}(x, y) := x - \ell_i &\Rightarrow \min_{l \in K_j} g_{i1}(v_{jl} + (x, y)) = \min\{x - \ell_i, \ell_j + x - \ell_i\} = x - \ell_i, \\ g_{i2}(x, y) := y - w_i &\Rightarrow \min_{l \in K_j} g_{i2}(v_{jl} + (x, y)) = y - w_i, \\ g_{i3}(x, y) := -x &\Rightarrow \min_{l \in K_j} g_{i3}(v_{jl} + (x, y)) = -\ell_j - x, \\ g_{i4}(x, y) := -y &\Rightarrow \min_{l \in K_j} g_{i4}(v_{jl} + (x, y)) = -y - w_j. \end{aligned}$$

Analogously, we obtain with  $v_{i1} = (0, 0)$ ,  $v_{i2} = (\ell_i, 0)$ ,  $v_{i3} = (\ell_i, w_i)$ , and  $v_{i4} = (0, w_i)$  the following:

$$\begin{aligned} g_{j1}(x, y) := x - \ell_j &\Rightarrow \min_{k \in K_i} g_{j1}(v_{ik} - (x, y)) = \min\{-x - \ell_j, \ell_i - x - \ell_j\} = -x - \ell_j, \\ g_{j2}(x, y) := y - w_j &\Rightarrow \min_{k \in K_i} g_{j2}(v_{ik} - (x, y)) = -y - w_j, \\ g_{j3}(x, y) := -x &\Rightarrow \min_{k \in K_i} g_{j3}(v_{ik} - (x, y)) = x - \ell_i, \\ g_{j4}(x, y) := -y &\Rightarrow \min_{k \in K_i} g_{j4}(v_{ik} - (x, y)) = y - w_i. \end{aligned}$$

In both cases, we obtain the same conditions, summarized in

$$\overline{U}_{ij} = \{(x, y) : -\ell_j \leq x \leq \ell_i, -w_j \leq y \leq w_i\}.$$

**To Exercise 12.9** Let the convex polygon  $P$ ,

$$P := \{(x, y) \in \mathbb{R}^2 : g_i(x, y) = a_i x + b_i y + c_i \leq 0, i \in I = \{1, \dots, m\}\},$$

be given in normalized position, and let  $\text{int}(P) \neq \emptyset$ . We denote the corner points of  $P$  by  $v_i = (x_i, y_i)$  and assume that  $g_i(v_i) = 0$  and  $g_{i+1}(v_i) = 0$  hold for  $i \in I$  where, as a convention,  $g_{m+1} := g_1$ . Because of the normalized position of  $P$ , we have  $P \subseteq [0, X] \times [0, Y]$  with  $X := \max_{i \in I} x_i$  and  $Y := \max_{i \in I} y_i$ .

It is obvious, any axes-parallel rectangle which is completely contained within a polygon cannot have maximal area if it has at most one of its corner points lying on the frontier of the polygon. Let  $(p, q)$  denote the lower left corner and  $\ell$  and  $w$  the length and width of the searched rectangle:  $R = \{(x, y) : p \leq x \leq p + \ell, q \leq y \leq q + w\}$ . Furthermore, we define the *lower (bottom)* and the *upper edge* of  $P$  by

$$\begin{aligned} Y_b &:= \{(x, y) : y = \min\{s : (x, s) \in P\}, x \in [0, X]\}, \\ Y_u &:= \{(x, y) : y = \max\{s : (x, s) \in P\}, x \in [0, X]\}, \end{aligned}$$

and the *left* and *right edge* of  $P$  by

$$\begin{aligned} X_l &:= \{(x, y) : x = \min\{s : (s, y) \in P\}, y \in [0, Y]\}, \\ X_r &:= \{(x, y) : x = \max\{s : (s, y) \in P\}, y \in [0, Y]\}. \end{aligned}$$

Assigned to these sets we use corresponding index sets  $I_b$ ,  $I_u$ ,  $I_l$ , and  $I_r$  indicating those restrictions which define the respective part of the frontier of  $P$ . Then, we have  $I_b = \{i \in I : b_i < 0\}$ ,  $I_u = \{i \in I : b_i > 0\}$ ,  $I_l = \{i \in I : a_i < 0\}$ , and  $I_r = \{i \in I : a_i > 0\}$ . Therefore, the following conditions are necessary for optimality of an axes-parallel rectangle  $R$ :

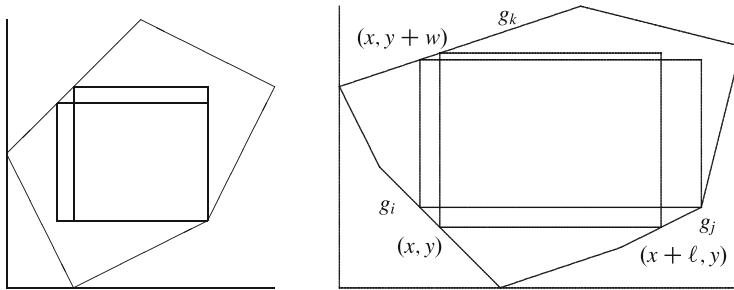
$$\begin{aligned} (p, q) \in Y_b \quad &\vee (p + \ell, q) \in Y_b, \\ (p, q + w) \in Y_u \quad &\vee (p + \ell, q + w) \in Y_u, \\ (p, q) \in X_l \quad &\vee (p, q + w) \in X_l, \\ (p + \ell, q) \in X_r \quad &\vee (p + \ell, q + w) \in X_r. \end{aligned} \tag{12.23}$$

Clearly, if at least one of the conditions is not fulfilled, then  $R$  can be increased in the respective direction.

For the four unknowns  $p$ ,  $q$ ,  $\ell$ , and  $w$ , we obtain from (12.23) at least two but not more than four linearly independent equations.

If (12.23) yields only equations for opposite lying points of  $R$ , then rectangle  $R$  can be moved without changing its size as far as a further equation is fulfilled. Moreover, if conditions (12.23) lead to four linearly independent equality restrictions, then  $R$  is uniquely determined and no optimization of the area is possible. Otherwise, that means, if three equations have to be fulfilled, only one parameter remains to be optimized. Two different situations which hereby can occur are shown in Fig. 12.19

In the following, we describe, by means of an example, the handling of a particular case where three corner points of  $R$  lie on the frontier of  $P$ . Here, we omit to discuss the other cases since that is possible in an analogous manner.



**Fig. 12.19** Example-polygons of Exercise 12.9

We consider a situation as depicted in Fig. 12.19, right part. For each edge  $g_i(x, y) = 0$  of  $P$  with  $\nabla g_i(x, y) = \begin{pmatrix} a \\ b \end{pmatrix} < 0$  (componentwise), that means,  $i \in I_b \cap I_l$ , the interval  $[x_{i-1}, x_i]$  has to be dissected into subintervals  $[\underline{x}, \bar{x}]$  such that for each  $x$  of a certain subinterval the two edges of  $P$  are uniquely determined which are cut by the horizontal and vertical lines containing  $(x, y)$  with  $g_i(x, y) = 0$ . Let these two edges of  $P$  be denoted by  $g_j$  and  $g_k$ .

In case of  $\nabla g_j > 0$  or  $\nabla g_k > 0$ , then the rectangle, determined by the three points, is not completely contained in  $P$ . Therefore, we now assume that  $a_j > 0$ ,  $b_j \leq 0$  and  $a_k \leq 0$ ,  $b_k > 0$  hold. In the considered case, corner point  $(p, q) = (x, y)$  and the both neighboring corner points of  $R$  fulfill the conditions  $(x, y) \in Y_b \cap X_l$ ,  $(x + \ell, y) \in Y_b \cap X_r$  and  $(x, y + w) \in Y_u \cap X_l$ . Length  $\ell$  and width  $w$  of the rectangle which is fixed by the three edges are dependent on  $x \in [\underline{x}, \bar{x}]$ . They can be obtained as solution of the linear system of equations

$$g_i(x, y) = 0, \quad g_j(x + \ell, y) = 0, \quad g_k(x, y + w) = 0,$$

that means, from

$$a_i x + b_i y + c_i = 0, \quad a_j(x + \ell) + b_j y + c_j = 0, \quad a_k x + b_k(y + w) + c_k = 0.$$

We obtain

$$y(x) = -\frac{a_i x + c_i}{b_i}, \quad \ell(x) = -\frac{g_j(x, y(x))}{a_j}, \quad w(x) = -\frac{g_k(x, y(x))}{b_k}$$

and, therefore, area  $A(x)$ :

$$\begin{aligned} A(x) &= \ell(x)w(x) = \left[ \left( \frac{a_i b_j - a_j b_i}{a_j b_i} \right) x + \frac{b_j c_i - b_i c_j}{a_j b_i} \right] \cdot \left[ \left( \frac{a_i b_k - a_k b_i}{b_i b_k} \right) x + \frac{b_k c_i - b_i c_k}{b_i b_k} \right] \\ &=: \frac{1}{a_j b_i^2 b_k} (\alpha x^2 + \beta x + \gamma) \end{aligned}$$

with

$$\begin{aligned}\alpha &= (a_i b_j - a_j b_i)(a_i b_k - a_k b_i), \\ \beta &= (a_i b_j - a_j b_i)(b_k c_i - b_i c_k) + (b_j c_i - b_i c_j)(a_i b_k - a_k b_i), \\ \gamma &= (b_j c_i - b_i c_j)(b_k c_i - b_i c_k).\end{aligned}$$

Due to the necessary optimality condition  $A'(x^*) = 0$ , we obtain a global maximum point  $x^* = -\beta/(2\alpha)$ , since  $\alpha < 0$ , and therefore  $A''(x) < 0$ , holds because of  $a_i < 0, b_i < 0, a_j > 0, b_j \leq 0, a_k \leq 0$ , and  $b_k > 0$ :

$$\alpha = \underbrace{a_j b_i a_i b_k}_{>0} \underbrace{\left( \frac{a_i b_j}{a_j b_i} - 1 \right)}_{<0} \underbrace{\left( 1 - \frac{a_k b_i}{a_i b_k} \right)}_{>0} < 0.$$

If  $x^* \in [\underline{x}, \bar{x}]$ , then  $x^*$  yields maximal area for that case. Otherwise, due to the concavity of  $A(x)$ , the maximum point lies on the limits of  $[\underline{x}, \bar{x}]$ . The complete containment of the obtained rectangle within  $P$  has still to be verified.

Performing respective investigations for all of the different  $[\underline{x}, \bar{x}]$ -intervals and the consideration of the other cases where point  $(x, y)$  does not lie on the frontier of  $P$ , the computation of an area-maximal rectangle can be done by calculating a finite number of function values.

## References

1. A. Albano, G. Sapuppo, Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *IEEE Trans. Syst. Man Cybern.* **10**(5), 242–248 (1980)
2. G. Amaral, J. Bernardo, J. Jorge, Marker-making using automatic placement of irregular shapes for the garment industry. *Comput. Graph.* **14**, 41–46 (1990)
3. R.C. Art, An approach to the two dimensional, irregular cutting stock problem. Technical report, IBM Cambridge Scientific Center Report (1966)
4. J.A. Bennell, J.F. Oliveira, The geometry of nesting problems: a tutorial. *Eur. J. Oper. Res.* **184**, 397–415 (2008)
5. J.A. Bennell, G. Scheithauer, Y. Stoyan, T. Romanova, Tools of mathematical modeling of arbitrary object packing problems. *Ann. OR* **179**, 343–368 (2010)
6. J.A. Bennell, G. Scheithauer, Y. Stoyan, T. Romanova, A. Pankratov, Optimal clustering of a pair of irregular objects. *J. Glob. Optim.* **61**(3), 497–524 (2014)
7. J. Blazewicz, M. Drozdowski, B. Soniewicki, R. Walkowiak, Two-dimensional cutting problem, basic complexity results and algorithms for irregular shapes. *Found. Cont. Eng.* **14**(4), 137–160 (1989)
8. J. Blazewicz, P. Hawryluk, R. Walkowiak, Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Ann. OR* **41**(1–4), 313–325 (1993)
9. E.K. Burke, R.S.R. Hellier, G. Kendall, G. Whitwell, Complete and robust no-fit polygon generation for the irregular stock cutting problem. *Eur. J. Oper. Res.* **179**(1), 27–49 (2007)
10. A. Casotto, F. Romeo, A. Sangiovanni-Vincentelli, A parallel simulated annealing algorithm for the placement of macro-cells. *IEEE Trans. Comput.* **6**, 838–847 (1987)

11. Y. Cui, Dynamic programming algorithms for the optimal cutting of equal rectangles. *Appl. Math. Model.* **29**(11), 1040–1053 (2005)
12. K.A. Dowsland, Some experiments with simulated annealing techniques for packing problems. *Europ. J. Oper. Res.* **68**, 389–399 (1993)
13. K.A. Dowsland, W.B. Dowsland, Solution approaches to irregular nesting problems. *Eur. J. Oper. Res.* **84**, 506–521 (1995)
14. H. Dyckhoff, G. Scheithauer, J. Terno, Cutting and packing, Chap. 22, in *Annotated Bibliographies in Combinatorial Optimization*, ed. by M. Dell'Amico, F. Maffioli, S. Martello (Wiley, Chichester, 1997), pp. 393–412
15. G. Fasano, A global optimization point of view to handle non-standard object packing problems. *J. Glob. Optim.* **55**, 279–299 (2013)
16. M. Fischetti, I. Luzzi, Mixed-integer programming models for nesting problems. *J. Heuristics* **15**(3), 201–226 (2009)
17. R.B. Grinde, T.M. Cavalier, A new algorithm for the minimum-area convex enclosing problem. *Eur. J. Oper. Res.* **84**, 522–538 (1995)
18. R.B. Grinde, T.M. Cavalier, Containment of a single polygon using mathematical programming. *Europ. J. Oper. Res.* **92**, 368–386 (1996)
19. R.B. Grinde, T.M. Cavalier, A new algorithm for the two-polygon containment problem. *Comput. Oper. Res.* **24**, 231–251 (1997)
20. R. Heckmann, T. Lengauer, Computing closely matching upper and lower bounds on textile nesting problems. *Eur. J. Oper. Res.* **108**, 473–489 (1998)
21. T. Imamichia, M. Yagiura, H. Nagamochia, An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optim.* **6**, 345–361 (2009)
22. Z. Li, V. Milenkovic, The complexity of the compaction problem, in *5th Canadian Conf. On Comp. Geom., Univ. Waterloo* (1993)
23. V.M.M. Marques, C.F.G. Bispo, J.J.S. Sentieiro, A system for the compactation of two-dimensional irregular shapes based on simulated annealing, in *IECON-91 (IEEE)* (1991), pp. 1911–1916
24. T.C. Martins, M.S.G. Tsuzuki, Rotational placement of irregular polygons over containers with fixed dimensions using simulated annealing and no-fit polygons. *J. Braz. Soc. Mech. Sci. Eng.* **30**(3), 205–212 (2008)
25. V.J. Milenkovic, Rotational polygon containment and minimum enclosure using only robust 2d constructions. *Comput. Geom.* **13**(1), 3–19 (1999)
26. V.J. Milenkovic, Densest translational lattice packing of non-convex polygons. *Comput. Geom.* **22**(1–3), 205–222 (2002)
27. V.J. Milenkovic, K. Daniels, Translational polygon containment and minimal enclosure using mathematical programming. *Int. Trans. Oper. Res.* **6**(5), 525–554 (1999)
28. V.J. Milenkovic, K. Daniels, Z. Li, Placement and compaction of nonconvex polygons for clothing manufacture, in *4th Canadian Conf. On Comp. Geom., St. John's* (1992)
29. V. Mornar, A procedure to optimize the raw material usage in printed wired boards production. *Automatika* **32**(5–6), 173–176 (1991)
30. I. Mukherjee, P.K. Ray, A review of optimization techniques in metal cutting processes. *Comput. Ind. Eng.* **50**(1–2), 15–34 (2006)
31. J.F. Oliveira, J.S. Ferreira, A application of simulated annealing to the nesting problem, in *Paper Presented at the 34th ORSA/TIMS Joint National Meeting*, San Francisco, CA (1992)
32. O.Y. Sergeyeva, G. Scheithauer, J. Terno, The value correction method for packing of irregular shapes, in *Decision Making under Conditions of Uncertainty (Cutting-Packing Problems)* (Ufa State Aviation Technical University, Ufa, 1997), pp. 261–269
33. M. Sharir, S. Toledo, Extremal polygon containment problems. *Comput. Geom.* **4**(2), 99–118 (1994)
34. Y.G. Stoyan, Mathematical methods for geometric design, in *Advances in CAD/CAM, Proceedings of PROLAMAT 82, Leningrad*, Amsterdam, ed. by T.M.R. Ellis, O.J. Semenkoc (1983), pp. 67–86

35. Y.G. Stoyan,  $\phi$ -function of non-convex polygons with rotations. *J. Mech. Eng.* **6**(1), 74–86 (2003)
36. Y.G. Stoyan, G.N. Yaskov, Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints. *Int. Trans. Oper. Res.* **5**(1), 45–57 (1998)
37. Y.G. Stoyan, M.V. Novozhilova, A.V. Kartashov, Mathematical model and method of searching for a local extremum for the non-convex oriented polygons allocation problem. *Eur. J. Oper. Res.* **92**, 193–210 (1996)
38. Y.G. Stoyan, A.V. Pankratov, Regular packing of congruent polygons on the rectangular sheet. *Eur. J. Oper. Res.* **113**(3), 653–675 (1999)
39. Y.G. Stoyan, V.N. Patsuk, A method of optimal lattice packing of congruent oriented polygons in the plane. *Eur. J. Oper. Res.* **124**(1), 204–216 (2000)

# Chapter 13

## Circle and Sphere Packing

Tasks of optimally packing circles or spheres mostly lead to mixed-integer nonlinear programming problems. Therefore, such problems can especially be hard to solve. Within this chapter we will address various problems of such kind and related solution strategies.

### 13.1 Problem Statements

In the context of optimally packing circles or spheres various different tasks are of interest. Some of them are listed below. Clearly, in all cases, the non-overlapping constraint and the complete containment condition have to be satisfied.

#### 1. *Packing circles into a circle*

Given  $m$  circles  $C_i$  with radii  $r_i$ ,  $i = 1, \dots, m$ , find a circle  $C$  with minimal radius  $R$  such that all circles  $C_i$  can simultaneously be placed in it. The problem of packing circles within a containing circle arises, for instance, when forming cable looms. This task is considered, for instance, in [7, 21].

#### 2. *Packing circles into a strip*

Given  $m$  circles  $C_i$  with radii  $r_i$ ,  $i = 1, \dots, m$ , and a strip of width  $W$  and unlimited length, find a feasible arrangement of all circles within the strip such that the length required is minimal [11, 29].

#### 3. *Packing identical circles into a circle*

Given  $m$  circles each with radius  $r = 1$ , find a circle  $C$  with minimal radius in which all  $m$  circles can be placed [12, 21].

#### 4. *Packing identical circles into a square*

Given  $m$  circles with radius  $r = 1$ , find a square  $Q$  with minimal size such that all  $m$  circles fit into  $Q$ . Or, equivalently, which maximal radius allows to place  $m$

identical circles within a unit square? Such kind of problems are considered, for instance, in [24, 32].

### 5. *Packing identical circles into a rectangle*

How many circles with radius  $r$  can be arranged on a rectangle  $L \times W$ ? It is obvious that, when stowing cylindrical products on a pallet, this kind of packing problem appears. It is, for instance, addressed in [18].

Similar problems are considered in the three-dimensional case, too. For a nice popular scientific introduction in that field of optimization problems, we refer to [27]. A comprehensive description of numerous applications of circle or sphere packing can be found in [7]. The placement of identical circles within regions of different shape, for instance in a isosceles triangle, is also of interest and, for instance, addressed in [1].

## 13.2 Packing Circles into a Circle

The following investigations and proposed solution techniques can be applied not only to packing problems with circles, but also to those involving pieces with irregular shape.

### 13.2.1 Modeling

Let  $(x_i, y_i)$  denote the center point (the allocation point) and  $r_i$  the (given) radius of circle  $C_i$  for  $i \in I = \{1, \dots, m\}$ . Without loss of generality, let  $(0, 0)$  be the center point of the enclosing circle  $C$ . Hence, in the problem of optimally packing  $m$  circles within a circle of minimal radius, we have  $2m + 1$  variables, namely the coordinates of center points of pieces and the radius  $R$  of  $C$ . For abbreviation, let

$$\underline{x} := (x_1, y_1, \dots, x_m, y_m, R)^\top \in \mathbb{R}^{2m+1}$$

denote the vector of variables. Then, a first model can be stated as follows:

#### **Model I of the circle packing problem**

$$R \rightarrow \min \quad \text{s.t.}$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq r_i + r_j, \quad i, j \in I, \quad i < j, \quad (13.1)$$

$$\sqrt{x_i^2 + y_i^2} \leq R - r_i, \quad i \in I. \quad (13.2)$$

Restrictions (13.1) ensure that the circles  $C_i$  do not overlap each other, and their entire containment within  $C$  is enforced by condition (13.2). Obviously, Model I is equivalent to the second model in which another formulation of restrictions is used:

### Model II of the circle packing problem

$$f(\underline{x}) := R \rightarrow \min \quad \text{s.t.} \quad (13.3)$$

$$g_{ij}(\underline{x}) := (r_i + r_j)^2 - (x_i - x_j)^2 - (y_i - y_j)^2 \leq 0, \quad i, j \in I, \quad i < j, \quad (13.4)$$

$$g_i(\underline{x}) := x_i^2 + y_i^2 - (R - r_i)^2 \leq 0, \quad i \in I, \quad (13.5)$$

$$g_0(\underline{x}) := \max\{r_i : i \in I\} - R \leq 0. \quad (13.6)$$

In Model II we additionally have to demand that radius  $R$  is bounded below by condition (13.6) to guarantee solvability.

**Proposition 13.1** *The region*

$$G_0 := \{\underline{x} \in \mathbb{R}^{2m+1} : g_i(\underline{x}) \leq 0, i \in I \cup \{0\}\},$$

which is induced by restrictions (13.5) and (13.6), is convex.

*Proof* First of all, we show that the set

$$\widetilde{G}_i = \{(x, y, R)^\top : \widetilde{g}_i(x, y, R) := x^2 + y^2 - (R - r_i)^2 \leq 0\}$$

is convex for every  $i \in I$ . Let  $(x, y, R)^\top \in \widetilde{G}_i$  and  $(a, b, R')^\top \in \widetilde{G}_i$ . Then it remains to show that  $t(x, y, R)^\top + (1-t)(a, b, R')^\top \in \widetilde{G}_i$  holds for all  $t \in (0, 1)$ . Due to the particular structure of  $\widetilde{g}_i$ , the inequalities

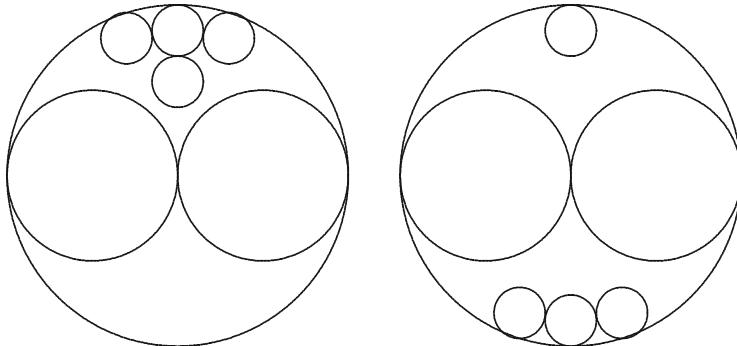
$$\left\| \begin{pmatrix} x \\ y \end{pmatrix} \right\| \leq R - r_i \quad \text{and} \quad \left\| \begin{pmatrix} a \\ b \end{pmatrix} \right\| \leq R' - r_i$$

are valid because of the assumption. Applying the Cauchy-Schwarz inequality, we obtain

$$xa + yb \leq |xa + yb| = \left| \begin{pmatrix} x \\ y \end{pmatrix}^\top \begin{pmatrix} a \\ b \end{pmatrix} \right| \leq \left\| \begin{pmatrix} x \\ y \end{pmatrix} \right\| \cdot \left\| \begin{pmatrix} a \\ b \end{pmatrix} \right\| \leq (R - r_i)(R' - r_i).$$

Moreover, for  $t(x, y, R)^\top + (1-t)(a, b, R')^\top$  and  $t \in (0, 1)$ , we have

$$\begin{aligned} & (tx + (1-t)a)^2 + (ty + (1-t)b)^2 - (tR + (1-t)R' - r_i)^2 \\ &= (tx + (1-t)a)^2 + (ty + (1-t)b)^2 - (t(R - r_i) + (1-t)(R' - r_i))^2 \\ &= t^2 \underbrace{(x^2 + y^2 - (R - r_i)^2)}_{=\widetilde{g}_i(x,y,R)\leq 0} + (1-t)^2 \underbrace{(a^2 + b^2 - (R' - r_i)^2)}_{=\widetilde{g}_i(a,b,R')\leq 0} \\ & \quad + 2t(1-t) \underbrace{(xa + yb - (R - r_i)(R' - r_i))}_{\leq 0} \leq 0. \end{aligned}$$



**Fig. 13.1** Non-convexity of  $G$

Since  $\widetilde{G}_i$  is convex, set  $\overline{G}_i := \{\underline{x} \in \mathbb{R}^{2m+1} : g_i(\underline{x}) \leq 0\}$  is convex as well. Moreover, restriction  $g_0(\underline{x}) \leq 0$  is linear. Therefore, set  $\overline{G}_0 := \{\underline{x} \in \mathbb{R}^{2m+1} : g_0(\underline{x}) \leq 0\}$  is also convex. As intersection of convex regions the convexity of  $G_0$  follows. ■

Obviously, all circles can be arranged within a circle with radius  $R_{max} := \sum_{i \in I} r_i$  so that the feasible region  $G$  can be described by

$$G := G_0 \cap G_1 \cap C_{max}$$

where

$$G_1 := \{\underline{x} \in \mathbb{R}^{2m+1} : g_{ij}(\underline{x}) \leq 0, i, j \in I, i < j\}, \quad C_{max} := \{\underline{x} \in \mathbb{R}^{2m+1} : R \leq R_{max}\}.$$

Region  $G$  is bounded but not convex. Since the objective function is linear, it suffices to search minimal solutions on the frontier of  $G$ . Due to the non-convexity of  $G$  a lot of local extreme points can exist.

*Example 13.1* Consider the packing of two circles with radius 10 and of  $m - 2$  circles ( $m > 3$ ) with different radii  $r_i = 3 + \varepsilon_i$  where  $|\varepsilon_i| \leq 1, i = 3, \dots, m$ .

If  $m$  is not too large, as in Fig. 13.1 (the remaining  $m - 6$  circles are not drawn), all circles can be placed within a circle of radius 20. Each different (feasible) distribution of the  $m - 2$  small circles yields an optimal packing. However, the connecting line between the two feasible points of  $G$ , depicted in Fig. 13.1, contains non-feasible ones. The non-feasibility is caused by the violation of restrictions (13.1) or (13.4). Hence, the feasible region is not convex.

If we enlarge the number  $m$  such that not all circles fit within a circle of radius 20, then we obtain local minimum solutions which are not global ones, in general. That happens, in particular, if  $\varepsilon_i \neq 0$  and  $\varepsilon_i \neq \varepsilon_j$  for all  $i \neq j, i, j \in \{3, \dots, m\}$ , hold. ■

To avoid *rotational symmetry* and *reflections*, we use in the following additional restrictions:

$$x_1 \geq 0, \quad y_1 = 0, \quad y_2 \geq 0. \quad (13.7)$$

Since the number of local solutions increases exponentially with  $m$ , even if (13.7) is used, heuristics or metaheuristics are applied, in general, to compute near-optimal or even optimal patterns. However, by means of nonlinear programming methods, at least proved local optimal solutions can be obtained [7]. Non-smooth optimization methods are applied in [26] to obtain global minima in a balanced circle packing problem. The usage of piecewise linear functions to compute tight bounds is proposed in [25] by means of circle cutting problems.

### 13.2.2 Computing a Local Minimum Solution

Let us consider Model II of the circle packing problem. A necessary condition for a local solution is given by the Karush-Kuhn-Tucker conditions of problem (13.3)–(13.6), see [22]. If  $\underline{x}$  is a local extreme point, then *dual* variables  $u_i$  and  $u_{ij}$  exist such that the following conditions are satisfied:

$$\begin{aligned} \nabla f(\underline{x}) + \sum_{i,j \in I, i < j} u_{ij} \nabla g_{ij}(\underline{x}) + \sum_{i \in I \cup \{0\}} u_i \nabla g_i(\underline{x}) &= 0, \\ u_{ij} g_{ij}(\underline{x}) &= 0, \quad i, j \in I, \quad i < j, \quad u_i g_i(\underline{x}) = 0, \quad i \in I \cup \{0\}, \\ u_{ij} &\geq 0, \quad i, j \in I, \quad i < j, \quad u_i \geq 0, \quad i \in I \cup \{0\}, \\ g_{ij}(\underline{x}) &\leq 0, \quad i, j \in I, \quad i < j, \quad g_i(\underline{x}) \leq 0, \quad i \in I \cup \{0\}. \end{aligned}$$

To verify local optimality of a feasible  $\underline{x}$ , at first the *non-active* restrictions, i.e., those with  $g_i(\underline{x}) < 0$  or  $g_{ij}(\underline{x}) < 0$ , respectively, are identified and the related dual variables are set to be 0.

If we can find exactly  $2m + 1$  restrictions among the active ones whose gradients are linearly independent, then the other  $u$ -values are obtained by solving the respective linear system of equations. It remains to verify the sign condition. If the gradients of active restrictions are not linearly independent, then there exist appropriate techniques for that particular degenerate case, for example described in [29]. In the case of less than  $2m + 1$  active restrictions other suitable methods can be applied.

In general, a feasible solution  $\underline{x}$  obtained by a heuristic does not represent a local extreme point, that means, there does not exist  $\underline{u} \in \mathbb{R}_+^{\overline{m}}$ , where  $\overline{m} = m(m - 1)/2 + m + 1$ , such that the necessary optimality condition (Karush-Kuhn-Tucker condition) is satisfied.

To get an improved feasible solution different approaches can be applied. Let  $J = \{(i, j) : i, j \in I, i < j\}$  denote an appropriate index set of all restrictions in (13.4). Furthermore, let  $J_0 = \{(i, j) \in J : g_{ij}(\underline{x}) = 0\}$  and  $I_0 := \{i \in I \cup \{0\} : g_i(\underline{x}) = 0\}$  be index sets of active restrictions of  $\underline{x}$ . If  $|I_0| + |J_0| < 2m + 1$  holds, then by solving the problem

$$R \rightarrow \min \quad \text{s.t.} \quad g_{ij}(\underline{x}) = 0, \quad (i, j) \in J_0, \quad g_i(\underline{x}) = 0, \quad i \in I_0, \quad (13.8)$$

an improved solution can possibly be obtained. Applying a penalty method to problem (13.8) leads to a non-restricted minimization problem

$$z_t(\underline{x}) := R + t \left( \sum_{(i,j) \in J_0} g_{ij}^2(\underline{x}) + \sum_{i \in I_0} g_i^2(\underline{x}) \right) \rightarrow \min$$

where penalty parameter  $t$  has to be chosen sufficiently large. (We refer, e.g., to [22] for a general description of penalty methods.) In this approach, it has simultaneously to be ensured that all restrictions which do not belong to  $J_0$  and  $I_0$  are satisfied as well. Further methods of nonlinear programming can be applied but all of them lead only to local optimal solutions, in general.

To any feasible solution  $\underline{x}$  an undirected graph  $G = (V, E)$  with vertex set  $V := I \cup \{0\}$  and edge set  $E$  can be assigned which characterizes the mutual interaction of the circles. Edge  $(i, j)$  with  $i, j \in I, i < j$ , belongs to  $E$  if and only if  $g_{ij}(\underline{x}) = 0$ , and  $(0, i) \in E$  if and only if  $g_i(\underline{x}) = 0$ . Hence, graph  $G$  represents the active restrictions of  $\underline{x}$ .

The mutual position of circles  $C_i, C_j, C_k$ , which are in contact, is uniquely determined (except reflection and rotation) if  $(i, j) \in E$ ,  $(i, k) \in E$ , and  $(j, k) \in E$ . Hence, to be able to find other solutions where  $C_i, C_j, C_k$  are not pairwise in contact, at least one of the three active restrictions has to become non-active.

### 13.2.3 Overcoming a Local Minimum

An obvious and often used strategy to overcome staying at a local minimum point consists in computing further feasible solutions (by the same or another heuristic) and starting for another local search from that. This approach is used, for instance, within the *Multi-Start Local Search* method as described in Sect. 7.4.

Another possibility consists in using different coordinate systems for modeling. This is based on the observation that a local solution with respect to a certain coordinate system is not necessarily a local minimizer with respect to another coordinate system. This fact is described, for instance, in [21] where *Cartesian* and *polar coordinates* are used.

Let  $(\varrho_i, \phi_i)$  denote the polar coordinates of the center point of  $C_i(x_i, y_i)$ ,  $i \in I$ . Thus, we have  $x_i = \varrho_i \cos \phi_i$  and  $y_i = \varrho_i \sin \phi_i$ . Furthermore, let  $\underline{\chi} = (\varrho_1, \phi_1, \dots, \varrho_m, \phi_m, R)^\top$  represent the vector of all variables with respect to a formulation in polar coordinates. Then, a corresponding mathematical model of the problem of optimally packing circles into a larger circle is as follows:

### Model III of the circle packing problem

$$\begin{aligned}\widetilde{f}(\underline{\chi}) &= R \rightarrow \min \quad \text{s.t.} \\ \widetilde{g}_{ij}(\underline{\chi}) &:= (r_i + r_j)^2 - \varrho_i^2 - \varrho_j^2 + 2\varrho_i\varrho_j \cos(\phi_i - \phi_j) \leq 0, \quad i, j \in I, \quad i < j, \\ \widetilde{g}_i(\underline{\chi}) &:= \varrho_i + r_i - R \leq 0, \quad i \in I, \\ \varrho_i &\geq 0, \quad \phi_i \in [0, 2\pi], \quad i \in I.\end{aligned}\tag{13.9}$$

Similar to (13.7), we can demand  $\phi_1 = 0$  and  $\phi_2 \in [0, \pi]$  to avoid symmetric solutions obtainable by rotation or reflection.

In the case that a local extreme point with respect to the polar coordinates is found, then, if some improvement was achieved, one can turn back to Cartesian coordinates, or to other coordinates, etc., taking the local solution as starting point for the changed coordinates.

A further strategy to overcome local extrema is proposed in [29] for the case  $|I_0| + |J_0| = 2m + 1$ . For a pair of circles  $C_i$  and  $C_j$  with  $r_i > r_j$ , both radii are considered as variables  $R_i$  and  $R_j$  within an auxiliary problem. Additionally, restrictions

$$R_i + R_j = r_i + r_j, \quad r_j \leq R_i \leq r_i,$$

are considered. Adding two further variables and only one equation yields a new free parameter. If in some solution of the auxiliary problem

$$\begin{aligned}R_i &\rightarrow \min \quad \text{s.t.} \\ g_{ij}(\underline{x}) &= 0, \quad (i, j) \in J_0, \quad g_i(\underline{x}) = 0, \quad i \in I_0, \quad R_i + R_j = r_i + r_j, \quad r_j \leq R_i \leq r_i,\end{aligned}$$

which has  $2m + 3$  variables and  $2m + 2$  restrictions,  $R_i = r_j$ , and hence  $R_j = r_i$ , holds, then the exchange of  $C_i$  with  $C_j$  can possibly lead to a decreased radius  $R$ . An appropriate treatment of cases  $|I_0| + |J_0| \neq 2m + 1$  is proposed in [29] as well. It is to notice that this approach yields, in general, improved local minima, but not necessarily global ones.

### 13.2.4 Further Heuristics

Besides the approaches discussed above, which are heuristics as well since local minima are computed, other, simpler, constructive heuristics are available.

Within *sequential packing* methods the already placed circles are considered to be fixed and the next circle is packed according to some strategy. A simple variant consists in packing the circles in a predefined sequence such that in each step the respective enclosing circle has minimal radius. To this end, the next circle is placed in contact to two others or such that it is tangent to a previously placed circle and the border of the enclosing circle. Such an approach is used, for instance, in [15].

A more expensive approach, which however produces better solutions, in general, is described in [17]. There, the next circle is selected from the set of not yet packed ones according to some target criterion which models the filling of empty space.

In [33] a related problem is considered. Given a circle  $C_0$  in normalized position with radius  $R_0$ , then the decision problem *Is it possible to place all  $m$  circles  $C_i$  within  $C_0$ ?* is solved approximately using the *gradient method*, cf. [22]. Due to the non-convexity of the problem, the decision answer is only correct, in general, if a feasible solution is obtained. Otherwise, one cannot conclude that no feasible solution exists.

An arrangement of  $m$  circles randomly generated without regarding (13.4) violates that restriction, in general. This can be measured using the penalty function

$$\theta_{ij}(\underline{x}) := \max\{0, g_{ij}(\underline{x})\}, \quad \theta_i(\underline{x}) := \max\{0, g_i(\underline{x})\}, \quad i, j \in I, i < j.$$

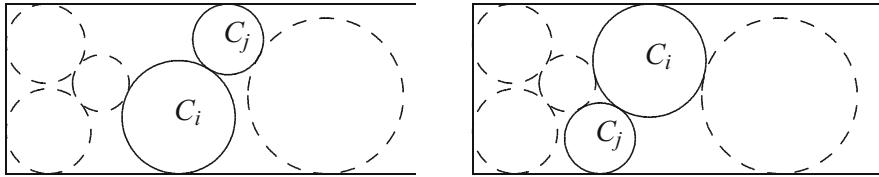
In this way, a non-restricted minimization problem

$$U(\underline{x}) := \sum_{i \in I} \theta_i(\underline{x}) + \sum_{i, j \in I, i < j} \theta_{ij}(\underline{x}) \rightarrow \min$$

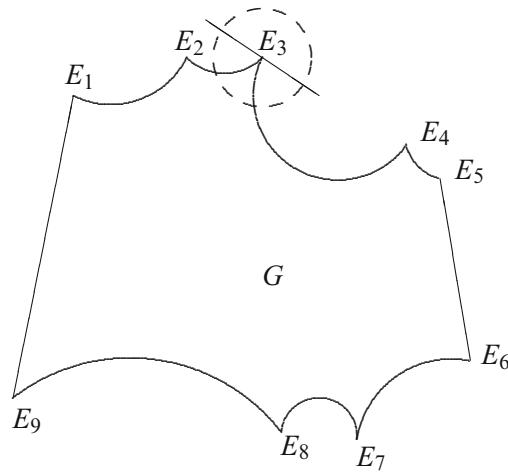
can be formulated. Hence, a local minimum point  $\underline{x}$  with  $U(\underline{x}) = 0$  is searched. This approach is applied, for instance, in [16] and [34]. If a feasible pattern is found, then, obviously, it is possible to start again with a somewhat decreased  $R_0$ . Thus, applying a sequence of decision problems, advantageously together with a bisection method, a good or even local optimal solution can be obtained.

## 13.3 Strip Packing

In case of the *Strip Packing Problem with circles*,  $m$  circles  $C_i$  with radii  $r_i$ ,  $i \in I = \{1, \dots, m\}$  have to be placed without overlap into a strip of given width  $W$  and minimal length  $L$ . Therefore, we need again  $2m + 1$  variables for the modeling:



**Fig. 13.2** Non-convexity of feasible region



**Fig. 13.3** Feasible region and extreme points

$m$  center points  $(x_i, y_i)$  and strip length  $L$ .

### Model of the strip packing problem with circles

$$L \rightarrow \min \quad \text{s.t.} \quad (13.10)$$

$$g_{ij}(\underline{x}) := (r_i + r_j)^2 - (x_i - x_j)^2 - (y_i - y_j)^2 \leq 0, \quad i, j \in I, \quad i < j,$$

$$g_i(\underline{x}) := \max\{r_i - x_i, x_i + r_i - L, r_i - y_i, y_i + r_i - W\} \leq 0, \quad i \in I. \quad (13.11)$$

Of course, restrictions (13.11) can be formulated each as four linear conditions as well. As Fig. 13.2 demonstrates, the region of all feasible packings contains, in general, non-connected components. Their number can rise exponentially with  $m$ .

Due to the linear terms in (13.11) the feasible region  $G$  of all arrangements can be illustrated as in Fig. 13.3. Because of the linearity of the objective function (13.10) it is possible to restrict the search for local solutions to extreme points  $E_k$  of  $G$ .

An approach for solving model (13.10)–(13.11) which is based on the *active set strategy* is proposed in [28, 29]. Sequential heuristics to construct feasible patterns can be found, for instance, in [14].

Moreover, a *Simulated Annealing* algorithm is used in [35] to solve the related decision problem *Is it possible to pack all  $m$  circles within a rectangle  $L \times W$ ?* As in the circle packing problem, in that case when no feasible packing could be found, it is not possible to conclude that no feasible pattern exists, in general.

Besides the strip packing problem, other containing regions and objective functions are subject of investigations, too. For instance, in [2] and [5] the arrangement of two maximal circles within a given polygon is considered.

### 13.4 Packing Identical Circles into a Circle

Besides the problem of packing  $n$  circles of radius  $r = 1$  into a circle with minimal radius  $R_n$ , addressed in Sect. 13.2, the reverse formulation is considered, too: *Which maximal radius  $r_n$  allows to pack  $n$  identical circles within the unit circle (radius 1)?* Obviously, the equalities

$$r_n \cdot R_n = 1, \quad n = 1, 2, \dots,$$

hold between the two respective optimal values.

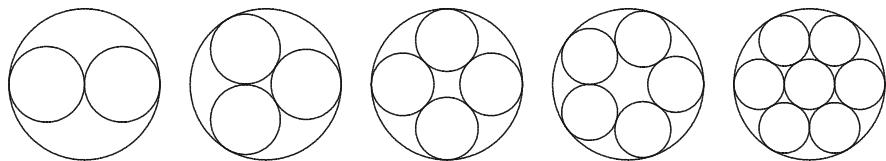
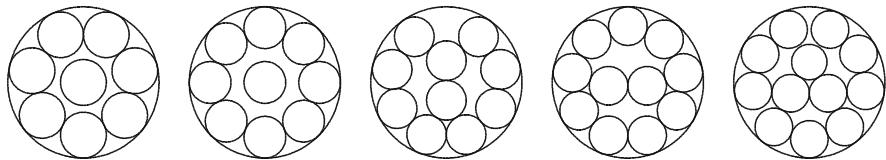
A further equivalent problem is as follows: *What is the maximum  $a_n$  of the minimal distance between two points of a given set of  $n$  points which lie within a unit circle?* It is easy to see that

$$a_n = \frac{2r_n}{1 - r_n} \quad \text{and} \quad r_n = \frac{a_n}{2 + a_n}, \quad n = 2, 3, \dots,$$

hold. Table 13.1, which is taken from [12], contains the maximal value  $a_n$  for  $n \leq 12$  and a lower bound of  $a_n$  for  $13 \leq n \leq 40$ . Furthermore, the number  $b_n$  of movable points (circles), the number  $c_n$  of active restrictions as well as the density  $d_n := nr_n^2$  of the respective pattern are given. It is remarkable, no regularities can be observed (except the monotone decrease of  $a_n$ ) which also holds for larger  $n$ . Results and corresponding patterns for  $41 \leq n \leq 65$  can be found in [12]. In Figs. 13.4 and 13.5 optimal patterns of  $n$  identical circles for  $n \leq 12$  are shown.

**Table 13.1** Maximal minimum distance of  $n$  points within a unit circle

| $n$ | $a_n$    | $b_n$ | $c_n$  | $d_n$    | $n$ | $a_n$    | $b_n$ | $c_n$ | $d_n$    |
|-----|----------|-------|--------|----------|-----|----------|-------|-------|----------|
| 1   |          | 0     | 1      | 1.000000 | 21  | 0.470332 | 2     | 38    | 0.761233 |
| 2   | 2.000000 | 0     | 3      | 0.500000 | 22  | 0.450479 | 0     | 44    | 0.743481 |
| 3   | 1.732051 | 0     | 6      | 0.646171 | 23  | 0.440024 | 0     | 46    | 0.747985 |
| 4   | 1.414214 | 0     | 8      | 0.686292 | 24  | 0.429954 | 2     | 44    | 0.751379 |
| 5   | 1.175571 | 0     | 10     | 0.685210 | 25  | 0.420802 | 1     | 48    | 0.755401 |
| 6   | 1.000000 | 0–5   | 10–14  | 0.666667 | 26  | 0.414235 | 2     | 48    | 0.765434 |
| 7   | 1.000000 | 0     | 18     | 0.777778 | 27  | 0.407631 | 0     | 54    | 0.773960 |
| 8   | 0.867767 | 1     | 14     | 0.732502 | 28  | 0.398809 | 2     | 52    | 0.773919 |
| 9   | 0.765367 | 1     | 16     | 0.689408 | 29  | 0.389211 | 4     | 50    | 0.769590 |
| 10  | 0.710978 | 0     | 20     | 0.687797 | 30  | 0.384783 | 0     | 60    | 0.781006 |
| 11  | 0.684040 | 0     | 23, 24 | 0.714460 | 31  | 0.377964 | 0     | 84    | 0.783164 |
| 12  | 0.660153 | 0     | 24     | 0.739021 | 32  | 0.368361 | 1     | 62    | 0.774106 |
| 13  | 0.618034 | 0     | 26     | 0.724465 | 33  | 0.364518 | 0     | 66    | 0.784271 |
| 14  | 0.600884 | 0     | 29     | 0.747253 | 34  | 0.356445 | 2     | 66    | 0.777947 |
| 15  | 0.567963 | 0     | 30     | 0.733759 | 35  | 0.351051 | 2     | 66    | 0.780342 |
| 16  | 0.553185 | 0     | 32     | 0.751098 | 36  | 0.348023 | 3     | 66    | 0.790884 |
| 17  | 0.527421 | 0     | 35     | 0.740302 | 37  | 0.347296 | 0     | 90    | 0.809965 |
| 18  | 0.517638 | 0, 1  | 41–44  | 0.760919 | 38  | 0.335464 | 3     | 70    | 0.784025 |
| 19  | 0.517638 | 0     | 48     | 0.803192 | 39  | 0.330148 | 2     | 74    | 0.782917 |
| 20  | 0.485164 | 1     | 38     | 0.762248 | 40  | 0.326592 | 4     | 72    | 0.788190 |

**Fig. 13.4** Optimal patterns for  $n \leq 7$ **Fig. 13.5** Optimal patterns for  $8 \leq n \leq 12$ 

## 13.5 Packing Circles into a Square

How many identical circles of radius  $r$  can be placed within a square of size  $\ell$ ? is another optimization problem frequently considered.

### 13.5.1 Modeling

Let  $n \in \mathbb{N}$  be fixed. We denote again the center points of the circles by  $(x_i, y_i)$ ,  $i \in I = \{1, \dots, n\}$ . Then the following model results:

#### Packing of identical circles into a unit square

$$\begin{aligned} R &\rightarrow \max && \text{s.t.} \\ (x_i - x_j)^2 + (y_i - y_j)^2 &\geq 4R^2, & i, j \in I, & i < j, \\ R \leq x_i \leq 1 - R, & R \leq y_i \leq 1 - R, & i \in I. \end{aligned} \quad (13.12)$$

It is obvious that the problem under consideration can be reformulated so that a maximal radius  $r_n$  has to be computed which allows to pack  $n$  identical circles into a unit square.

In a further, often used, formulation  $n$  points have to be distributed within the unit square such that the minimum distance  $a_n$  between any two points is maximized. It is easy to see, if any of  $a_n$  or  $r_n$  is known, then the other is determined by

$$r_n = \frac{a_n}{2(1 + a_n)} \quad \text{and} \quad a_n = \frac{2r_n}{1 - 2r_n}, \quad n = 2, 3, \dots,$$

respectively.

### 13.5.2 Lower Bounds

Concerning the maximum of the minimal distance between any two of  $n$  points within a unit square, the following statement is proposed in [31]. Let

$$d_{ij} := \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

denote the Euclidean distance between  $(x_i, y_i)$  and  $(x_j, y_j)$ .

**Proposition 13.2** *Let  $n \geq 2$ . Furthermore, let*

$$S_n := \{(x_i, y_i) \in [0, 1] \times [0, 1] : d_{ij} > 0, i < j, i, j \in I := \{1, \dots, n\}\}$$

*be a set of  $n$  points within the unit square and let  $S_n^*$  denote the entire set of such point sets. Then*

$$\max_{S_n \in S_n^*} \min_{i, j \in I: i < j} d_{ij} \geq \max\{L_1(n), L_2(n), L_3(n), L_4(n), L_5(n)\}$$

holds where

$$\begin{aligned} L_1(n) &:= 1/(\lceil \sqrt{n} \rceil - 1), \\ L_2(n) &:= 1/(\lceil \sqrt{n+1} \rceil - 3 + \sqrt{2 + \sqrt{3}}), \\ L_3(n) &:= 1/(\lceil \sqrt{n+2} \rceil - 5 + 2\sqrt{2 + \sqrt{3}}), \\ L_4(n) &:= \begin{cases} (k^2 - k - \sqrt{2k})/(k^3 - 2k^2), & \text{if } n = k(k+1), k \in \mathbb{N}, \\ 0, & \text{otherwise,} \end{cases} \\ L_5(n) &:= \begin{cases} \sqrt{\frac{1}{p^2} + \frac{1}{q^2}}, & \text{if } \begin{cases} n = \lceil (p+1)(q+1)/2 \rceil, \\ p^2 \leq 3q^2, q^2 \leq 3p^2, p, q \in \mathbb{N} \end{cases} \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

The proof of this statement is based on particular patterns and the fact that a lower bound valid for  $n$  is also a lower bound for  $n_0$  if  $n_0 < n$  holds.

### 13.5.3 Upper Bounds

A first non-trivial upper bound  $U_1(n)$  is a consequence of a theorem of Oler [23].

**Theorem 13.1 (Oler)** *If  $X$  represents a compact and convex subset of the plane, then the number of points in  $X$  whose distance is not less than 1 is not larger than*

$$\frac{2}{\sqrt{3}}F(X) + \frac{1}{2}U(X) + 1,$$

where  $F(X)$  denotes the area and  $U(X)$  the perimeter of  $X$ .

In case of a square  $X$  with size  $s$ , demanding

$$\frac{2}{\sqrt{3}}s^2 + \frac{1}{2}4s + 1 \geq n$$

and due to the enclosing unit square, we obtain:

$$a_n \leq U_1(n) := \frac{1 + \sqrt{1 + 2(n-1)/\sqrt{3}}}{n-1},$$

where  $a_n$  denotes the maximal minimum distance between any two points.

A second upper bound  $U_2(n)$  for  $a_n$  is proposed in [6]:

$$U_2(n) := \frac{2}{\sqrt{n\pi + C_n(\sqrt{3} - \frac{\pi}{2}) + (4\lfloor \sqrt{n} \rfloor - 2)(2 - \frac{\pi}{2}) - 2}}$$

where

$$C_n = \begin{cases} n - 2, & \text{for } 3 \leq n \leq 6, \\ n - 1, & \text{for } 7 \leq n \leq 9, \\ 3 \lfloor n/2 \rfloor - 5 + n \bmod 2, & \text{for } n \geq 10. \end{cases}$$

### 13.5.4 Optimal Patterns

The following condition is necessary for the optimality of a pattern of  $n \geq 2$  points within the unit square: if  $\{P_1, \dots, P_n\}$  represent a distribution of  $n$  points with maximal minimum distance between any two points, then at least one point of  $\{P_1, \dots, P_n\}$  is lying on each edge of the square.

Similar to Sects. 13.2 and 13.3, proving optimality of a certain pattern is very costly, in general. In [20] a proof is presented for the case  $n = 6$ . We briefly describe here the method of proving used in [24] to show optimality of  $a_n$  for  $n = 10, \dots, 20$ . For fixed  $n$ , the following four steps are performed.

1. Find a tight lower bound  $a$  for  $a_n$ . This can be done by means of appropriate heuristics.
2. Limit the set of possible packings of  $n$  circles with diameter  $a$  on a set of  $2n$ -dimensional intervals. This is performed using an *elimination procedure* which is based on a comprehensive case-by-case analysis.
3. Compute a local solution with respect to those intervals.
4. Verify that the obtained local solution is as well a global one for the considered interval.

It is obvious that this way of proving optimality is only practicable by means of computers. However, the plurality of local solutions restricts its applicability to relatively small  $n$ .

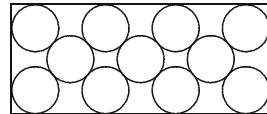
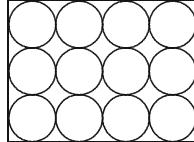
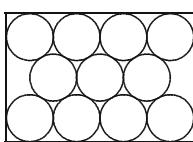
Table 13.2 (proposed in [31]) contains values  $a_n$  and numbers  $b_n$  of movable points (circles) as well as numbers  $c_n$  of active restrictions. Furthermore, the density  $d_n := n\pi r_n^2$  in an optimal pattern is given. It is remarkable again that except the monotone decrease of  $a_n$  no further rule can be observed.

## 13.6 Packing Identical Circles into a Rectangle

Concerning the packing of pallets with cylindrical goods, we are confronted with a real-world optimization problem. Due to practical purposes, frequently only regular patterns are used even if they do not reach the maximal possible number of packable pieces. As seen in the previous section, the computation of a proved optimal packing pattern is very costly, in general.

**Table 13.2** Maximal minimum distance of  $n$  points within a unit square

| $n$ | $a_n$    | $b_n$ | $c_n$ | $d_n$    | $n$ | $a_n$    | $b_n$ | $c_n$ | $d_n$    |
|-----|----------|-------|-------|----------|-----|----------|-------|-------|----------|
| 1   |          | 0     | 4     | 0.785365 | 21  | 0.271812 | 2     | 39    | 0.753358 |
| 2   | 1.414214 | 0     | 5     | 0.539012 | 22  | 0.267958 | 1     | 43    | 0.771680 |
| 3   | 1.035276 | 0     | 7     | 0.609645 | 23  | 0.258819 | 0     | 56    | 0.763631 |
| 4   | 1.000000 | 0     | 12    | 0.785398 | 24  | 0.254333 | 0     | 56    | 0.774963 |
| 5   | 0.707107 | 0     | 12    | 0.673765 | 25  | 0.250000 | 0     | 60    | 0.785398 |
| 6   | 0.600925 | 0     | 13    | 0.663957 | 26  | 0.238735 | 2     | 56    | 0.758469 |
| 7   | 0.535898 | 1     | 14    | 0.669311 | 27  | 0.235850 | 0     | 55    | 0.772311 |
| 8   | 0.517638 | 0     | 20    | 0.730964 | 28  | 0.230536 | 1     | 56    | 0.771854 |
| 9   | 0.500000 | 0     | 24    | 0.785398 | 29  | 0.226883 | 1     | 65    | 0.778906 |
| 10  | 0.421280 | 0     | 21    | 0.690036 | 30  | 0.224503 | 0     | 65    | 0.792019 |
| 11  | 0.398207 | 2     | 20    | 0.700742 | 31  | 0.217547 | 4     | 54    | 0.777297 |
| 12  | 0.388730 | 0     | 25    | 0.738468 | 32  | 0.213082 | 3     | 63    | 0.776004 |
| 13  | 0.366096 | 1     | 25    | 0.733265 | 33  | 0.211328 | 1     | 64    | 0.788852 |
| 14  | 0.348915 | 1     | 32    | 0.735679 | 34  | 0.205605 | 0     | 80    | 0.776649 |
| 15  | 0.341081 | 0     | 36    | 0.762056 | 35  | 0.202764 | 0     | 80    | 0.781227 |
| 16  | 0.333333 | 0     | 40    | 0.785398 | 36  | 0.200000 | 0     | 84    | 0.785398 |
| 17  | 0.306154 | 1     | 34    | 0.733550 | 37  | 0.196238 | 2     | 73    | 0.783303 |
| 18  | 0.300463 | 0     | 38    | 0.754653 | 38  | 0.195342 | 0     | 74    | 0.797041 |
| 19  | 0.289542 | 2     | 37    | 0.752308 | 39  | 0.194365 | 0     | 80    | 0.811179 |
| 20  | 0.286612 | 0     | 44    | 0.779494 | 40  | 0.188176 | 2     | 85    | 0.787979 |

**Fig. 13.6** Regular patterns

Favorable regular patterns, as shown in Fig. 13.6, can easily be computed for standardized pallets as described, for instance, in [10, 18]. More complex patterns, similar to that of 2- and 3-group patterns of rectangles, Sect. 6.5, are proposed in [8, 9]. Therein, cutting problems in the metalworking industry are investigated.

An inverse problem is considered in [19] where minimum perimeter rectangles are computed that enclose congruent non-overlapping circles.

## 13.7 Sphere Packing

Concerning the three-dimensional variant of circle packing, similar optimization problems appear and respective models can straightforwardly be translated from the two- to the three-dimensional case [4]. Packing of spheres with different radii within

a cuboid is, for instance, addressed in [30]. Therein, the gradient method is used to obtain local solutions.

In the context of sphere packing *Kepler's conjecture* attains vital importance. Kepler's conjecture states that no packing pattern of identical spheres, in the three-dimensional Euclidean space, has average density larger than

$$\frac{\pi}{\sqrt{18}} \approx 0.74048.$$

This density is reached by packing the spheres in pyramidal and hexagonal form.

In 1998, Hales announced that he has found a proof of Kepler's conjecture [13]. His proof contains a comprehensive case-by-case analysis where subproblems are analyzed by means of a computer. Because of the computer-based arguments his proof is still controversial. In 2014, Hales announced the completion of another, formal proof that can be verified by automated proof checking software.

A comprehensive overview of (theoretical) results concerning packing  $d$ -dimensional spheres with  $d \geq 2$  is given in [3].

## 13.8 Exercises

**Exercise 13.1** Compute a rectangle of minimal area which can completely contain a circle with radius  $R$  and a circle with radius  $r$  without overlap.

**Exercise 13.2** Given three circles  $C_i(x_i, y_i)$  with radii  $r_i > 0$ ,  $i = 1, 2, 3$ , which are pairwise in contact. Starting with center points  $(0, 0)$  and  $(r_1 + r_2, 0)$  for circles  $C_1$  and  $C_2$ , compute all points  $(x_3, y_3)$  such that the three circles are in contact.

**Exercise 13.3** Let  $m \geq 2$ . Show, if  $\underline{x}$  represents a feasible pattern of  $m$  circles within a circle with radius  $R$  and if  $G = (V, E)$  denotes the associated graph, defined on page 390, then the following assertions hold.

- (a) If  $|\{i \in I : (0, i) \in E\}| \leq 1$ , then  $\underline{x}$  is not optimal.
- (b) If  $R = r_i + r_j$  holds for two indices  $i, j \in I$ ,  $i \neq j$ , then  $\underline{x}$  is optimal.

**Exercise 13.4** Compute the maximal radii  $r_n$  for  $n = 1, 2, \dots, 7$  such that  $n$  identical circles fit within a circle of radius 1.

**Exercise 13.5** Consider the arrangement of four circles with radius  $r_i = 1$  within a circle with minimal radius  $R$ . Starting from pattern  $\underline{x}^0 = (1, 0, 0, \sqrt{3}, -1, 0, 0, -\sqrt{3}, \sqrt{3} + 1)^\top$ , solve the auxiliary problem

$$\begin{aligned} F(\underline{x}) := (g_{24}(\underline{x}))^2 &\rightarrow \min && \text{s.t.} \\ g_{12}(\underline{x}) = 0, g_{23}(\underline{x}) = 0, g_{34}(\underline{x}) = 0, g_{14}(\underline{x}) = 0, g_2(\underline{x}) = 0, g_4(\underline{x}) = 0 \end{aligned}$$

with some suitable method.

**Exercise 13.6** Given two circles  $C_i$  with radii  $r_i$ ,  $i = 1, 2$ . Compute a circle  $C$  with center point  $(0, 0)$  and minimal radius  $R$  as well as the allocation points  $(x_i, y_i)$ ,  $i = 1, 2$ , such that  $C_1(x_1, y_1)$  and  $C_2(x_2, y_2)$  are located completely in  $C$  without overlap.

**Exercise 13.7** Given three circles  $C_i$  with radii  $r_1 = r_2 = 1$  and  $r_3 \in (0, \infty)$ . Compute circle  $C = C(r_3)$  with center point  $(0, 0)$  and minimal radius  $R(r_3)$  as well as the allocation points  $(x_i, y_i)$ ,  $i = 1, 2, 3$ , such that all three circles are located completely in  $C(r_3)$  without overlap.

## 13.9 Solutions

**To Exercise 13.1** Without loss of generality, we assume  $R = 1 \geq r > 0$ . Let  $C(r)$  denote a circle with radius  $r$ .

**Case 1:** For sufficiently small  $r$ , that means, for  $r \leq r_0$ , both circles can be placed within a square of size 2. Because of  $1 + r_0 + \sqrt{2}r_0 = \sqrt{2}$ , we can calculate radius  $r_0$  and obtain  $r_0 = (\sqrt{2} - 1)/(\sqrt{2} + 1) = (\sqrt{2} - 1)^2 = 3 - 2\sqrt{2}$ . Hence, for  $r \leq r_0$  the minimal value is equal to 4 (or  $4R^2$ ).

**Case 2:** Now, let  $r > r_0$ .

To obtain minimal area, the two circles have to be in contact. We suppose that the center point of  $C(1)$  is  $(0, 0)$  and the center point of  $C(r)$  is located on the straight line with ascent  $\tan \alpha$  within the first quadrant. Then the rectangle which encloses both circles  $C(1)$  and  $C(r)$  has length  $a(\alpha)$  and height  $b(\alpha)$  with

$$a(\alpha) = 1 + \max\{1, (1+r)\cos\alpha + r\}, \quad b(\alpha) = 1 + \max\{1, (1+r)\sin\alpha + r\}.$$

Thus, three subcases appear for area  $A(\alpha) = a(\alpha) \cdot b(\alpha)$ :

- (i)  $0 \leq \alpha \leq \alpha_0$ , where  $\alpha_0 = \arcsin(1-r)/(1+r)$  is determined by condition  $b(\alpha) = 2$ , that means, by  $1 = r + (1+r)\sin\alpha_0$ :

$$A(\alpha) = 2a(\alpha) = 2(1 + (1+r)\cos\alpha + r).$$

Since  $A'(\alpha) \leq 0$ , function  $A$  is monotone decreasing and has its minimum in  $[0, \alpha_0]$  for  $\alpha = \alpha_0$ .

- (ii)  $\alpha_1 \leq \alpha \leq \pi/2$  with  $\alpha_1 = \pi/2 - \alpha_0$ : (symmetry with respect to the bisecting line  $\alpha = \pi/4$ )  
 (iii)  $\alpha_0 \leq \alpha \leq \alpha_1$ :  $A(\alpha) = (1 + (1+r)\cos\alpha + r)(1 + (1+r)\sin\alpha + r)$

$$\begin{aligned} A'(\alpha) &= -(1+r)\sin\alpha(1 + (1+r)\sin\alpha + r) \\ &\quad +(1 + (1+r)\cos\alpha + r)(1+r)\cos\alpha \\ &= (1+r)^2(\cos^2\alpha + \cos\alpha - \sin^2\alpha - \sin\alpha) \\ &= (1+r)^2(\cos\alpha - \sin\alpha)(\cos\alpha + \sin\alpha + 1). \end{aligned}$$

Hence,  $A'(\alpha) \geq 0$  follows for  $\alpha_0 \leq \alpha \leq \pi/4$ . Therefore,  $A$  takes its minimum for  $\alpha = \alpha_0$ .

The considerations for  $\pi/4 \leq \alpha \leq \alpha_1$  yield a symmetric solution  $\alpha = \alpha_1$ .

Summarizing, the minimal area is equal to  $A(\alpha_0) = A(\alpha_1) = 2(1 + r + 2\sqrt{r})$ .

Note that the necessary condition  $A'(\alpha) = 0$  leads to a local maximum.

**To Exercise 13.2** We assume that  $(x_1, y_1) = (0, 0)$  and  $(x_2, y_2) = (r_1 + r_2, 0)$  hold. Hence, the center point  $(x, y) = (x_3, y_3)$  has to be found. Using

$$d_{12} := r_1 + r_2, \quad d_{13} := r_1 + r_3, \quad d_{23} := r_2 + r_3,$$

we obtain, because of  $x^2 + y^2 = d_{13}^2$  and  $(d_{12}-x)^2 + y^2 = d_{23}^2$ , a linear equation for  $x$  and, therefore, the two solutions:  $x = (d_{12}^2 + d_{13}^2 - d_{23}^2)/(2d_{12})$ ,  $y = \pm \sqrt{d_{13}^2 - x^2}$ .

**To Exercise 13.3** (a) It is obvious that, if no restriction is active, then the enclosing circle can be decreased while maintaining the center points. In that case when only a single restriction is active among those which determine the enclosing circle, then it is not minimal. To explain this assertion in more detail, we construct, starting with the feasible pattern  $\underline{x} = (x_1, y_1, \dots, x_m, y_m, R_0)$ , another circle with smaller radius. Without loss of generality, we suppose that circle  $C_m$  determines circle  $R_0$  and that  $x_m > 0$ ,  $y_m = 0$ , and, therefore,  $R_0 = x_m + r_m$  hold. We decrease  $R$  and simultaneously shift the center point  $(x, y)$  so that always  $h_m(x, R) := x + R - x_m - r_m = 0$  and  $y = 0$  are satisfied.

Because of the assumption,  $h_i(0, R_0) < 0$  holds with  $h_i(x, R) := \sqrt{(x - x_i)^2 + y_i^2} - (R - r_i)$  for  $i = 1, \dots, m-1$ . Hence, we consider the optimization problem

$$R \rightarrow \min \quad \text{s.t.} \quad h_i(x, R) \leq 0, \quad i = 1, \dots, m-1, \quad h_m(x, R) = 0.$$

Since  $h_m(x, R) = 0$ , it follows that  $R = x_m + r_m - x$ , and, because of  $h_i(x, R) \leq 0$ , it follows that

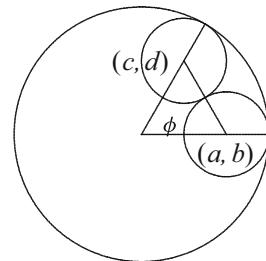
$$\sqrt{(x - x_i)^2 + y_i^2} \leq R - r_i \quad \text{and} \quad (x - x_i)^2 + y_i^2 \leq (x_m + r_m - r_i - x)^2$$

hold for  $i \in \{1, \dots, m-1\}$ . With  $x^2 - 2xx_i + x_i^2 + y_i^2 \leq (R_0 - r_i)^2 - 2x(R_0 - r_i) + x^2$ , we obtain  $2x \leq ((R_0 - r_i)^2 - x_i^2 - y_i^2)/(R_0 - r_i - x_i)$ . Since  $R$  becomes smaller when  $x$  increases,

$$x^* := \min_{i=1, \dots, m-1} \frac{(R_0 - r_i)^2 - x_i^2 - y_i^2}{2(R_0 - r_i - x_i)} \quad \text{and} \quad R^* := R_0 - x^*$$

yield, because of  $x^* > 0$ , a feasible pattern with minimal radius.

**Fig. 13.7** Notations to Exercise 13.4



(b) Since the minimal radius of the enclosing circle has to be at least as large as  $\max\{r_i + r_j : i < j, i, j \in I\}$ , optimality follows.

**To Exercise 13.4** We consider a pattern of two neighbored circles which are in contact to the frontier of the unit circle. The coordinates of the center point of the first circle are denoted by  $(a, b) := (\varrho, 0)$  and that of the second one by  $(c, d) := (\varrho \cos \phi, \varrho \sin \phi)$  in polar coordinates where  $\varrho$  is determined by radius  $r = r(n)$  since  $\varrho + r = 1$ . Angle  $\phi$  depends on the number of identical circles which are located close to the frontier and are pairwise in contact, that means  $\phi = 2\pi/n$ . According to Fig. 13.7, it follows that  $(\varrho \cos \phi - \varrho)^2 + (\varrho \sin \phi)^2 = (2r)^2$  and  $\varrho = 1 - r$  are satisfied for  $n > 2$ . Furthermore, with  $R := 1 - \cos \phi$ , it follows that  $r^2(2 - R) + 2Rr - R = 0$  holds and, therefore,  $r = (\sqrt{2R} - R)/(2 - R)$ . We obtain  $r(2) = 1/2$ ,  $r(3) = 2\sqrt{3}/3 - 1 \approx 0.4641$ ,  $r(4) = \sqrt{2}/2 - 1 \approx 0.4142$ ,  $r(5) \approx 0.3702$  and  $r(6) = 1/3$ . These patterns are optimal. For  $n = 7$ , an optimal pattern is obtained by adding the seventh circle in central position.

**To Exercise 13.5** A global minimum is given by  $\underline{x} = (\sqrt{3}, 0, 0, 1, -\sqrt{3}, 0, 0, -1, 2)^\top$  since  $g_{24}(\underline{x}) = 0$ . The corresponding circle with radius  $R = 2$  is infeasible since restrictions  $g_1(\underline{x}) \leq 0$  and  $g_3(\underline{x}) \leq 0$  are disregarded in the auxiliary problem.

**To Exercise 13.6** Obviously, the solution is symmetric with respect to rotation. With  $R := r_1 + r_2$ , we obtain:  $x_1(\phi) = (R - r_1) \cos \phi$ ,  $y_1(\phi) = (R - r_1) \sin \phi$ ,  $x_2(\phi) = -(R - r_2) \cos \phi$ ,  $y_2(\phi) = -(R - r_2) \sin \phi$ .

**To Exercise 13.7** For  $r_3 \leq 2/3$ , all three circles fit within an enclosing circle with radius  $R = 2$ . For  $r_3 > 2/3$ , because of the demanded center point position of the enclosing circle, the nonlinear system of equations

$$\begin{aligned} x_i^2 + y_i^2 &= (R - r_i)^2, \quad i = 1, 2, 3, \\ (x_i - x_j)^2 + (y_i - y_j)^2 &= (r_i + r_j)^2, \quad 1 \leq i < j \leq 3, \end{aligned}$$

has to be solved. To avoid rotational symmetry, we choose the following arrangement:  $x_1 = x_2 < 0$ ,  $y_1 = 1$ ,  $y_2 = -1$ ,  $x_3 > 0$ , and  $y_3 = 0$ . Therefore, we initially

obtain the reduced system of equations

$$x_1^2 + 1 = (R - 1)^2, \quad x_3 = R - r_3, \quad x_3 - x_1 = \sqrt{(1 + r_3)^2 - 1} = \sqrt{r_3^2 + 2r_3}.$$

Replacing  $x_3$  yields  $x_1 = R - a$  with  $a := r_3 + \sqrt{r_3^2 + 2r_3} \geq 2$ . Therefore,  $x_1$  can be eliminated:  $x_1^2 = (R - 1)^2 - 1 = (R - a)^2$ . Finally,  $R = a^2/(2a - 2)$  follows.

## References

1. G. Belov, G. Scheithauer, Setup and open-stacks minimization in one-dimensional stock cutting. *INFORMS J. Comput.* **19**(1), 27–35 (2007)
2. S. Bespamyatnikh, Packing two disks in a polygon. *Comput. Geom.* **23**(1), 31–42 (2002)
3. K. Bezdek, Sphere packings revisited. *Eur. J. Comb.* **27**(6), 864–883 (2006)
4. E.G. Birgin, F.N.C. Sobral, Minimizing the object dimensions in circle and sphere packing problems. *Comput. Oper. Res.* **35**(7), 2357–2375 (2007)
5. P. Bose, P. Morin, A. Vigneron, Packing two disks into a polygonal environment. *J. Discrete Algorithms* **2**(3), 373–380 (2004)
6. L.G. Casado, D.I. Garcia, P.G. Szabo, T. Csendes, Equal circle packing in a square II – New results for up to 100 circles using the TAMSASS-PECS algorithm, in *New Trends in Equilibrium Systems* (Kluwer, Boston, 2000), pp. 1–16
7. I. Castillo, F.J. Kampas, J.D. Pinter, Solving circle packing problems by global optimization: numerical results and industrial applications. *Eur. J. Oper. Res.* **191**, 786–802 (2008)
8. Y. Cui, Dynamic programming algorithms for the optimal cutting of equal rectangles. *Appl. Math. Model.* **29**(11), 1040–1053 (2005)
9. Y. Cui, Generating optimal multi-segment cutting patterns for circular blanks in the manufacturing of electric motors. *Eur. J. Oper. Res.* **169**, 30–40 (2006)
10. K.A. Dowsland, Optimising the palletisation of cylinders in cases. *OR Spektrum* **13**, 204–212 (1991)
11. J.A. George, J.M. George, B.W. Lamar, Packing different-sized circles into a rectangular container. *Eur. J. Oper. Res.* **84**, 693–712 (1995)
12. R.L. Graham, B.D. Lubachevsky, K.J. Nurmela, P.R.J. Östergård, Dense packings of congruent circles in a circle. *Discrete Math.* **181**, 139–154 (1998)
13. T.C. Hales, A proof of the Kepler conjecture. *Ann. Math.* **162**, 1065–1185 (2005)
14. M. Hifi, R. M'Hallah, Approximate algorithms for constrained circular cutting problems. *Comput. Oper. Res.* **31**(5), 675–694 (2004)
15. M. Hifi, R. M'Hallah, Strip generation algorithms for constrained two-dimensional two-staged cutting problems. *Eur. J. Oper. Res.* **172**, 515–527 (2006)
16. W.Q. Huang, M. Chen, Note on: an improved algorithm for the packing of unequal circles within a larger containing circle. *Comput. Ind. Eng.* **50**(3), 338–344 (2006)
17. W.Q. Huang, Y. Li, C.M. Li, R.C. Xu, New heuristics for packing unequal circles into a circular container. *Comput. Oper. Res.* **33**(8), 2125–2142 (2006)
18. H. Isermann, Heuristiken zur Lösung des zweidimensionalen Packproblems für Rundgefäß. *OR Spektrum* **13**, 213–223 (1991)
19. B.D. Lubachevsky, R.L. Graham, Minimum perimeter rectangles that enclose congruent non-overlapping circles. *Discrete Math.* **309**(8), 1947–1962 (2009)
20. H. Melissen, Densest packing of six equal circles in a square. *El. Math.* **49**, 27–31 (1994)
21. N. Mladenovic, F. Plastria, D. Urosevic, Reformulation descent applied to circle packing. Working Paper, Belgrad (2003)

22. J. Nocedal, S.J. Wright, *Numerical Optimization* (Springer, New York, 1999)
23. N. Oler, An inequality in the geometry of numbers. *Acta Math.* **105**, 19–48 (1961)
24. R. Peikert, Dichteste Packungen von gleichen Kreisen in einem Quadrat. *El. Math.* **49**, 16–26 (1994)
25. St. Rebennack, Computing tight bounds via piecewise linear functions through the example of circle cutting problems. *Math. Methods Oper. Res.* **84**, 3–57 (2016)
26. P.I. Stetsyuk, T.E. Romanova, G. Scheithauer, On the global minimum in a balanced circular packing problem. *Optim. Lett.* **10**(6), 1347–1360 (2016)
27. I. Stewart, Wie viele kreisförmige Kekse passen auf ein Kuchenblech? *Spektrum der Wissenschaft* **3**, 112–114 (1999)
28. Y.G. Stoyan, G.N. Yaskov, Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints. *Int. Trans. Oper. Res.* **5**(1), 45–57 (1998)
29. Y.G. Stoyan, Y. Yaskov, A mathematical model and a solution method for the problem of placing various-sized circles into a strip. *Eur. J. Oper. Res.* **156**(3), 590–600 (2004)
30. Y.G. Stoyan, Y. Yaskov, G. Scheithauer, Packing of various radii solid spheres into a parallelepiped. *Cent. Eur. J. Oper. Res.* **11**(4), 389–408 (2003)
31. P.G. Szabo, T. Csendes, L.G. Casado, D.I. Garcia, Equal circle packing in a square I – Problem setting and bounds for optimal solutions. *New Trends in Equilibrium Systems* (Kluwer, Boston, 2000), pp. 1–15
32. P.G. Szabo, M.C. Markot, T. Csendes, E. Specht, L.G. Casado, I. Garcia, *New Approaches to Circle Packing in a Square* (Springer, New York, 2007)
33. H. Wang, W. Huang, Q. Zhang, D. Xu, An improved algorithm for the packing of unequal circles within a larger containing circle. *Eur. J. Oper. Res.* **141**, 440–453 (2002)
34. D.F. Zhang, A.S. Deng, An effective hybrid algorithm for the problem of packing circles into a larger containing circle. *Comput. Oper. Res.* **32**(8), 1941–1951 (2005)
35. D.F. Zhang, Y. Liu, S. Chen, Packing different-sized circles into a rectangular container using simulated annealing algorithm, in *ENFORMATIKA* (2004), pp. 388–391

# Index

- Aggregation of constraints, 130
- Algorithm
  - genetic, 209
  - of Ford and Moore, 23
  - of Gilmore and Gomory, 20
  - of Steinberg, 198
  - of Wang, 172
- Allocation interval, 246
- Allocation point, 124, 125, 184, 211, 247, 285, 320, 322, 386
  - reduced set of potential, 37, 162
  - set of potential, 34, 159
- Alternative concept, 28, 90
- Arcflow model, 103
- Area bound, 129, 149, 186, 213
- Bar relaxation, 192, 283, 333
  - combined, 134
  - horizontal, 132
  - of Kantorovich-type, 131
  - vertical, 133
- Basic interval, 253
- Basis matrix, 78
- Beasley-type model, 125, 184
- Best Fit, 60
  - one-dimensional, 88
- Best Fit Decreasing, 60
- Best Fit Decreasing Height, 198
- Bin packing problem
  - Gilmore/Gomory model, 49
  - harmonic algorithm, 216
  - multiple, 68, 105
  - one-dimensional, 13, 47, 74, 216, 219
- three-dimensional, 318, 329, 338
- two-dimensional, 227
  - variable size, 68
- Bisection, 90
- BL heuristic, 371
- Block pattern, 286
- Bottom-left justified, 126
- Branch-and-bound, 25, 247
  - for the knapsack problem, 25
  - for the OPP, 147
  - for the SPP, 211
- Circle packing problem, 10, 385
- Column generation, 151, 192, 334, 338
- Constraint programming, 124, 152
- Container loading problem, 317
- Containment condition, 3, 352
- Contiguous condition, 135
- Contiguous relaxation
  - horizontal, 136
  - vertical, 137
- Continuous bound, 112
- Continuous relaxation, 26, 76, 107, 128
- Contour, 146, 197, 211, 324, 370
  - approach, 211
  - concept, 147
  - point, 147, 150, 211
  - region, 211
- Convex hull, 5, 364
- Convex object, 5
- Convex polygon, 348
- Cutting pattern, 5

- Cutting problem
  - one-dimensional, 12
  - two-dimensional, 123, 157, 183, 345
- Cutting stock problem, 74
  - 2-stage guillotine, 115
  - arcflow model, 103
  - Gilmore/Gomory model, 75
  - IRUP, 107
  - MIRUP, 107
  - one-cut model, 101
  - one-dimensional, 13, 74
  - pattern-oriented model, 75
  - standard model, 75
  - three-dimensional, 115, 177
  - two-dimensional, 115
- Discrete function
  - decreasing, 297
  - increasing, 297
- Divisible case, 108
- Dominance, 29
  - of feasible solutions, 29
  - of patterns, 151, 255, 303
- Dual feasible function, 54
  - data-dependent, 58
  - discrete, 57
  - maximal, 56
- Due dates, 104
- Dynamic programming, 20, 247, 294, 322
  - backward, 36
  - forward, 33, 36
- Equivalence
  - of instances, 59, 94, 109, 280
  - of models, 76
  - of patterns, 151
- Farley bound, 82
- Feasibility problem, 123
- Finger-joining technology, 246
- First Fit, 60
  - one-dimensional, 88
- First Fit Decreasing, 60
- First Fit Decreasing Height, 197
- Fix-length, 246
- Floor Ceiling heuristic, 241
- Forward state strategy, 247
- G4 heuristic, 284, 302, 309
- Genetic Algorithm, 209, 328
- Gilmore/Gomory model, 49, 75
- Guillotine
  - pattern, 157, 291, 328
- Guillotine cut, 157
  - stage number, 157
- Guillotine cutting, 157
  - 2-stage, 164, 219
  - 3-stage, 165, 264
  - exact case, 164, 165, 214
  - general, 158
  - non-exact case, 164, 165, 214, 220
- Hesse normal form, 361
- Heuristic
  - BFDH, 198
  - BL, bottom-up left-justified, 196
  - Bottom Left, 207
  - FFDH, 198
  - First Fit Shelf, 204
  - Harmonic Shelf, 203, 216
  - Next Fit Shelf, 204
  - NFDH, 197
  - of Steinberg, 144
  - Reverse Fit, 198
  - sequential packing, 370
- HFF heuristic, 240
- HNF heuristik, 240
- Hodograph, 12, 350, 370
- Hybrid First Fit heuristic, 240
- Hybrid Next Fit heuristik, 240
- Integer property, 108
- Integer round-up property, 108
- Kantorovich-type model, 97
- Karush-Kuhn-Tucker conditions, 389
- Kepler's conjecture, 400
- Kerf, 16, 273
- Knapsack problem, 19, 149, 164, 268, 328, 334, 338
  - 0/1, 7, 20
  - branch-and-bound, 25
  - dynamic programming, 23
  - generalized 0/1, 3
  - longest path method, 23, 30
  - two-dimensional, 123
- Lattice packing, 347
- Lattice point, 371

- Layer
  - pattern, 338
  - relaxation, 338
- Left-justified, 35
- Lexicographic sorting, 30, 98
- LIFO
  - Last In First Out, 147
  - strategy, 147, 212, 309
- Linear programming problem, 78
- Local Search, 207
  - iterated , 208
  - multi-start, 208
- Local solution, 207, 389
- Longest path method, 254
- LP relaxation, 76, 128
- M4 heuristic, 302
- Material bound, 112, 129, 149, 186, 213
- Maximal embedded rectangle, 347
- Metaheuristic, 207, 208, 372
- Minimal enclosing rectangle, 347
- Minimum Bin Slack, 62
- Minimum enclosure problem, 347
- Minimum height bound, 186
- Minkowski sum, 12, 350
- Model
  - Beasley-type, 125
  - Kantorovich-type, 131
  - Padberg-type, 127
- Modified integer round-up property, 108
- Multi container loading problem, 318
- Multi pallet loading Problem, 305
- Multi-Start Local Search, 390
- Next Fit, 60
  - one-dimensional, 88
  - two-dimensional, 203
- Next Fit Decreasing, 60
- Next Fit Decreasing Height, 197
- Nicholson principle, 38
- No-fit polygon, 12, 350
- Non-active restriction, 389
- Non-convex polygon, 365
- Non-overlapping condition, 3
- Normalized
  - $\Phi$ -function, 360
  - pattern, 8, 35
  - position, 2, 348
- NP-hard, 14, 19
- Offline, 61, 195
- One-cut model, 101, 176
- Online, 61, 195
- Open stack problem, 104
- Orthogonal packing problem, 123
- Packing pattern, 5
- Packing vector, 338
- Padberg-type model, 127, 186, 229
- Pallet loading problem, 280
  - area bound, 282
  - Barnes bound, 283
  - distributer's, 279
  - guillotine, 291
    - block problem, 293
  - Isermann bound, 283
  - manufacturer's, 279
  - material bound, 282
  - minimal area bound, 283
  - minimum size instance, 282
- Pattern, 5, 125, 284
  - $\ell$ -pattern, 291
  - $k$ -block, 285
  - $k$ -block structure, 286
  - 1-group, 170
  - 2- and 3-block, 170
  - 2- and 3-group, 399
  - 4-block structure, 302
  - block- $\ell$ -pattern, 292
  - bottom-left justified, 126, 211
  - degree of utilization, 110
  - density, 110
  - dominance, 151
  - elementary, 74, 108
  - equivalence of, 151
  - feasible, 74
  - G4-structure, 287
  - guillotine, 157, 291
    - structure, 286
  - homogeneous, 74, 168, 285
  - left-justified, 35
  - maximal, 74, 108
  - monotone, 147
  - non-guillotine, 157
  - normalized, 8, 35, 126, 153, 211, 289, 324
    - proper, 74, 108
    - vector, 74
  - Performance bound, 63
  - Performance ratio, 63
    - asymptotic, 205
    - average-case, 67
    - worst-case, 67
      - absolute, 51, 63
      - asymptotic, 51, 63

- of a heuristic, 63
- of a lower bound, 51
- Periodicity of solutions, 29
- Permutation model, 14
- $\Phi$ -function, 9, 346
  - normalized, 360
- Polar coordinates, 403
- Polygon
  - convex, 358
  - nonconvex, 362
  - packing problem, 346
- Pseudo-polynomial, 21
- Reduced set of potential allocation points, 94, 302
- Reduction method, 59
- Reference interval, 253
- Reference point, 2
- Relaxation
  - continuous, 26, 76, 321
  - Kantorovich-type, 131
  - LP, 76
  - proper, 114
- Residual instance, 109
- Residual problem, 89
  - of a KP, 25
- Restricted master problem, 338
- Rotation, 371
  - matrix, 359
  - of polygons, 358
- Rotation matrix, 3
- Reference point, 348
- Salzer-series, 120, 216
- Sequence-pair, 129
- Sequential value correction method, 89, 103, 210, 372
- Setup costs, 89, 103
- Simplex method
  - primal, 79
  - revised, 78, 79
- Simplex multiplier, 79
- Simulated Annealing, 209, 372, 394
- Skyline, 197, 211
- Skyline algorithm, 145
- Strip packing problem, 183
  - lower bounds, 186
  - three-dimensional, 318
  - two-dimensional, 183
  - with guillotine constraint, 218
- Symmetry
  - of solutions, 48
  - rotational, 389
- Tabu Search, 209, 282, 328, 372
- Translation, 2
- Trimming cut, 220
- Volume bound, 129, 310
- Worst Fit, 60
- Worst Fit Decreasing, 60
- Worst-case example, 217
- Worst-case performance ratio
  - absolute, 51, 63
  - asymptotic, 51, 63
  - of a heuristic, 63
  - of a lower bound, 51