Integer programming column generation strategies for the cutting stock problem and its variants

Nancy Perrot

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Nancy PERROT

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : MATHÉMATIQUES APPLIQUÉES

****

INTEGER PROGRAMMING COLUMN GENERATION STRATEGIES FOR THE CUTTING STOCK PROBLEM AND ITS VARIANTS

****

Soutenue le : 29 Juin 2005

Après avis de :

MM.	Alberto CAPRARA	Professeur, Université de Bologne (Italie)	**Rapporteurs**
	Jacques DESROSIERS	Professeur, École HEC de Montréal (Canada)	

Devant la commission d'examen formée de :

MM.	Paul MOREL	Professeur, Université Bordeaux 1	**Président**
	Jacques DESROSIERS	Professeur, École HEC de Montréal (Canada)	**Examinateurs**
	Denis JAUBERT	Industriel, Greycon (Angleterre)	
	Brigitte JAUMARD	Professeur, Université de Montréal (Canada)	
	Claude LEMARÉCHAL	Directeur de Recherche, INRIA Rhone-Alpes	
	François VANDERBECK	Professeur, Université Bordeaux 1	

Remerciements

Je tiens à remercier vivement mon directeur de thèse, François Vanderbeck, qui est à l'origine de cette thèse. Il a su manifester tout au long de ce travail intérêt et confiance.

Je remercie les professeurs Jacques Desrosiers et Alberto Caprara d'avoir accepté d'être rapporteurs de ce travail, ainsi que pour leurs suggestions et conseils.

Je suis particulièrement reconnaissante à Claude Lemaréchal de m'avoir permis d'étendre mon domaine de recherche et de m'avoir ouvert des opportunités intéressantes tout au long de ma thèse. Il a accepté d'être membre de mon jury de thèse, je l'en remercie. Je remercie également l'ensemble des membres du jury de thèse et en particulier Paul Morel qui a présidé ce jury.

Un immense merci à Olivier Briant pour ces remarques constructives, sa disponibilité et ses connaisssances dans de nombreux domaines, sans oublier sa précieuse amitié et son soutien permanent. Merci également à mes collègues qui m'ont permis de travailler dans une ambiance excellente, en particulier à Magalie et Marie pour leur présence, leur amitié et leurs encouragements. Enfin, je salue chaleureusement mes proches, et tout particulièrement mes parents qui m'ont toujours fait confiance et soutenue, et auxquels je dédie ce travail.

Contents

Abstract

This thesis gives a comprehensive view of the scope of formulations and related solution approaches for the cutting stock problem (CSP) and its variants. The focus is on branch-and-price approaches. Specialized algorithms are developed for knapsack subproblems that arise in the course of the algorithm. Thorough numerical tests are used to identify good strategies for initialization, stabilization, branching, and producing primal solutions. Industrial variants of the problem are shown to be tractable for a branch-and-price approach.

The models studied are the following: the standard cutting stock and bin packing problems, a variant in which the production levels lie in a prescribed interval of tolerance, the multiple width cutting stock problem in which stock pieces are of different size, a variant with additional technical constraints that arise typically in industrial applications, and a variant where the number of distinct cutting patterns used is minimized given a waste threshold.

First, we consider various formulation of the Cutting Stock Problem (CSP): different models of the knapsack subproblem can be exploited to reformulate the CSP. Moreover, we introduce different ways of modeling local exchanges in the solution (primal exchanges imply dual constraints that stabilize the column generation procedure). Some models are shown to be valid integer programming (IP) reformulations while others define relaxations. The dual bounds defined by their LP solution are compared theoretically.

Then, we study the variants of the knapsack subproblem that arise in a column generation approach to the CSP. The branching constraints used in the branch-and-price algorithm can result in class bound and setup cost or the need for a binary decomposition in the subproblem. We show how standard knapsack solvers (dynamic programming approach and specialized branch-and-bound algorithm) can be extended to these variants of the knapsack problem.

Next, we discuss some branch-and-price implementation strategies. We compare different modes of initialization of the column generation procedure, we present our numerical study of various stabilization strategies to accelerate convergence of the procedure. We compare in particular the impact of the various ways of introducing local exchanges in our primal model and other stabilization techniques such as dual solution smoothing techniques or penalization from a stability center that prevent the fluctuation of the dual variables. To generate the columns we study different strategies based on the use of heuristic columns or on a multiple generation of columns. We also consider the use of heuristics based on column generation to find a primal bound. These are compared to a classic constructive heuristic. Then, we compare the different branching rules that are used in the branch-and-price procedure.

Finally, we present numerical results on two industrial applications that correspond to the variant with technical restrictions for which we minimize first the waste and then the number of setups.

Introduction

Le problème de découpe consiste à découper des pièces de petite taille dans de larges rouleaux, de façon à satisfaire une demande associée à chacune de ces pièces. L'objectif est principalement de réduire au minimum la perte correspondant à la partie inutilisée de ces rouleaux. Une solution est donnée par un ensemble de plans de découpe réalisables, c'est à dire des façons de couper les petites pièces sur les rouleaux, de façon à ce que leur production totale couvre les demandes. La dimension, d, peut être 1, 2 ou 3 où d est le nombre de dimensions significatives des rouleaux et des pièces commandées. Nous pouvons même avoir $d = 1\frac{1}{2}$ si les pièces demandées ont 2 dimensions significatives tandis que les rouleaux ont une dimension fixe et une dimension variable; d peut être supérieur à 3 si on considère des dimensions de temps ou de poids. Les processus de découpe changent selon les types de coupes, les placements des pièces, le nombre d'étapes dans le processus de découpe, etc. Il peut également y avoir des contraintes additionnelles ou des objectifs secondaires au problème (équilibrage de la charge de travail entre différentes machines de découpe, minimisation du nombre de plans de découpe différents, ou respect des dates dues par exemple).

Ici nous nous intéressons au problème de découpe unidimensionnel. Ce modèle est d'un grand intérêt dans le domaine de la recherche. La méthode de génération de colonnes a été développée sur cette application. C'est également une application pratique qui intervient, par exemple, dans les industries d'acier ou de papier. Ce dernier processus de découpe est illustré par la figure 2.

Dans le chapitre 1 nous présentons les formulations compactes des diverses variantes qui seront examinées. Nous commençons par les problèmes de découpe et de "bin-packing" standard. Les variantes considérées sont les problèmes dans

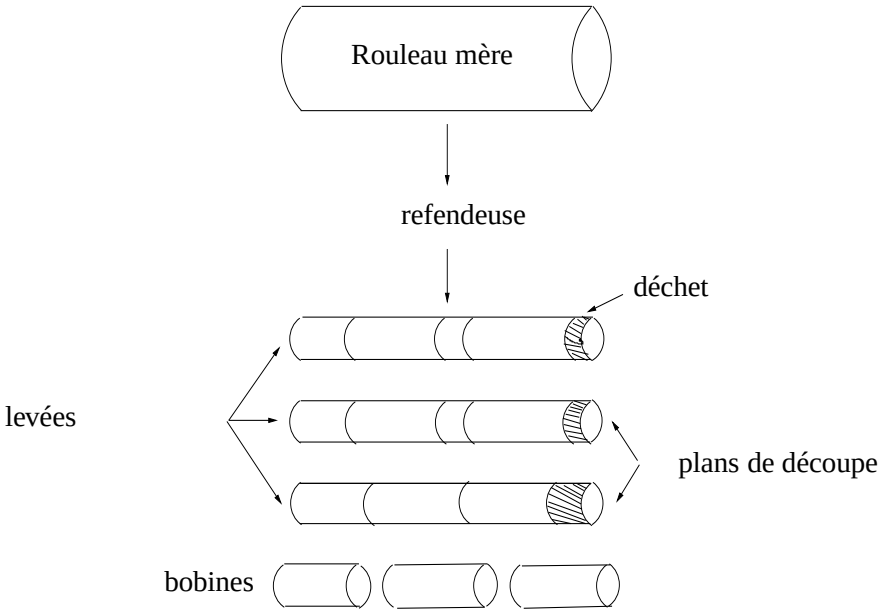


Figure 1: Le processus de découpe de rouleaux

lesquels : les niveaux de production se situent dans un intervalle prescrit de tolérance, les rouleaux à découper sont de différentes tailles, ou, des contraintes techniques doivent être prises en compte. Nous présentons également une formulation pour la minimisation du nombre de plans de découpe distincts en tant que deuxième objectif une fois que le déchet est fixé à sa valeur optimale. Nous donnons une brève revue de la littérature sur le problème de découpe qui sera complétée tout au long des chapitres par une étude détaillée des travaux spécifiquement liés au nôtre.

Le chapitre 2 traite de diverses reformulations du problème de découpe (CSP). Nous considérons d'abord différentes formulations pour le sous-problème de sac à dos. Chacune de ces formulations mène à une reformulation explicite du CSP et à une reformulation implicite qui peut être résolue en utilisant une procédure dynamique de génération de colonnes. Nous passons en revue également les approches hybrides qui ont été proposées dans la littérature. En outre, nous développons des reformulations alternatives implicites modélisant des échanges locaux dans la solution, et nous montrons comment ces échanges impliquent un effet de stabilisation de la procédure de génération de colonnes, dans la mesure

où ils reviennent implicitement à ajouter des contraintes duales. Certains de ces modèles sont des reformulations valides tandis que d'autres définissent des relaxations. Ces reformulations sont comparées d'un point de vue théorique pour leur relaxation linéaire (bornes LP) et les solutions en nombre entier (bornes primales). Certaines des formulations considérées ici sont des contributions originales de cette thèse. De plus, ce chapitre offre une classification des formulations possibles, comparant la force de la borne duale LP et présentant des observations sur leur intérêt pratique.

Le chapitre 3 traite du sous-problème de sac à dos. Le problème de sac à dos standard peut être résolu relativement efficacement en utilisant des algorithmes spécialisés. Cependant, lorsqu'on résout la reformulation de génération de colonnes en nombres entiers, nous employons des règles de branchement qui peuvent mener à des modifications du sous-problème de sac à dos. Pour modéliser correctement le coût réduit après branchement, nous devons effectuer une décomposition binaire des variables du sous-problème de sac à dos et introduire des variables de "setup" associées à chaque objet commandé. Le point central de ce chapitre est l'étude du modèle résultant appelé problème binaire multi-classe de sac à dos avec "setups". Nous montrons comment adapter des résultats standard pour le problème de sac à dos à ce modèle plus complexe. Nous caractérisons les solutions optimales de sa relaxation linéaire, nous montrons comment obtenir une borne LP en utilisant une procédure gloutonne, et nous proposons un algorithme de "branch-and-bound" en profondeur d'abord (pour le cas avec coût fixe positif) ainsi que des procédures de programmation dynamique pour résoudre ce sous-problème modifié. Cette recherche a donné lieu à la publication [32].

Dans le chapitre 4, nous étudions tour à tour chaque étape de la procédure de "branch-and-price" pour le problème de découpe et nous comparons différentes stratégies d'implémentation à travers des tests numériques. Nous présentons différents modes d'initialisation pour la procédure de génération de colonnes. Nous comparons des techniques de stabilisation sur plusieurs variantes. En particulier, nous examinons la contribution en terme de stabilisation de notre reformulation avec échanges intégrés. Nous expérimentons la méthode simple de

stabilisation “boxstep” et des techniques de lissage des variables duales. Nous faisons également la comparaison entre la génération de colonnes basée sur le LP et l’utilisation de la méthode des faisceaux pour résoudre le problème maître. Cette étude a contribué à la publication [6]. Nous développons également des heuristiques primales basées sur l’approche de génération de colonnes que nous comparons à une heuristique constructive standard. Ensuite, nous considérons différentes stratégies pour générer des colonnes (colonne unique ou colonnes multiples, exactes ou solutions heuristiques du sous-problème et une stratégie de diversification). Finalement, nous testons la contribution individuelle de nos règles de branchement pour la convergence vers une solution en nombre entier et pour l’amélioration des bornes duales.

Le chapitre 5 est consacré à l’étude de problèmes de découpe industriels qui combinent les difficultés des variantes du CSP passées en revue au chapitre 1. Leur formulation est construite sur base de celles présentées au chapitre 2. Une étude de cas du problème réel de la papeterie de Condat est présentée. Le problème implique une tolérance sur les niveaux de production, des restrictions techniques du processus de découpe et deux critères d’optimisation (déchet et nombre de plans de découpe différents). Une deuxième application réelle avec une contrainte minimale sur la multiplicité des plans de découpe est étudiée. Les meilleures stratégies dérivées de notre étude expérimentale du chapitre 4 sont appliquées à ces problèmes. Elles nous permettent de montrer qu’on peut résoudre des exemples réels avec un code générique de “branch-and-price” utilisant un solveur commercial de programme en nombre entier mixte (MIP) pour les solutions du problème maître et du sous-problème.

Introduction

In the cutting stock problem, one has a supply of pieces (objects) of stock material on one hand and a set of demands for "small" pieces of this material on the other hand. One must satisfy these demands by cutting the required pieces out of the stock pieces. The objective is primarily to minimize the waste that is counted as the unused part of used pieces of stock material. A solution is given by a set of feasible *cutting patterns*, i.e. assortments of order pieces that can be cut out of a given piece of stock material, such that their accumulated production of ordered pieces covers the demands. The dimensionality, d, can be 1, 2 or 3 where d is the number of dimension of the stock and order pieces that are significant. We can even have $d = 1\frac{1}{2}$ if order pieces have 2 significant dimensions while stock pieces have a fixed dimension and a variable one; d can be greater than 3 if time or weight dimensions are considered. The cutting processes vary according to the types of cuts that are allowed (guillotine or nested, orthogonal or not), the geometrical arrangements of pieces, the number of cutting stages, etc. There might also be some side-constraints or secondary objectives to the problem to do with the balancing of the workload between different cutting machines, the minimization of the number of different cutting pattern used, or the respect of due dates for instances.

Here we shall be concerned with the one-dimensional cutting stock problem. This model is of great interest from a research point of view. It has been the application on which the column generation method was developed. But it is also a practical application that arises in steel or paper industries, for example. The latter cutting process is illustrated in Figure 2.

In chapter 1 we present the compact formulations of the various variants that

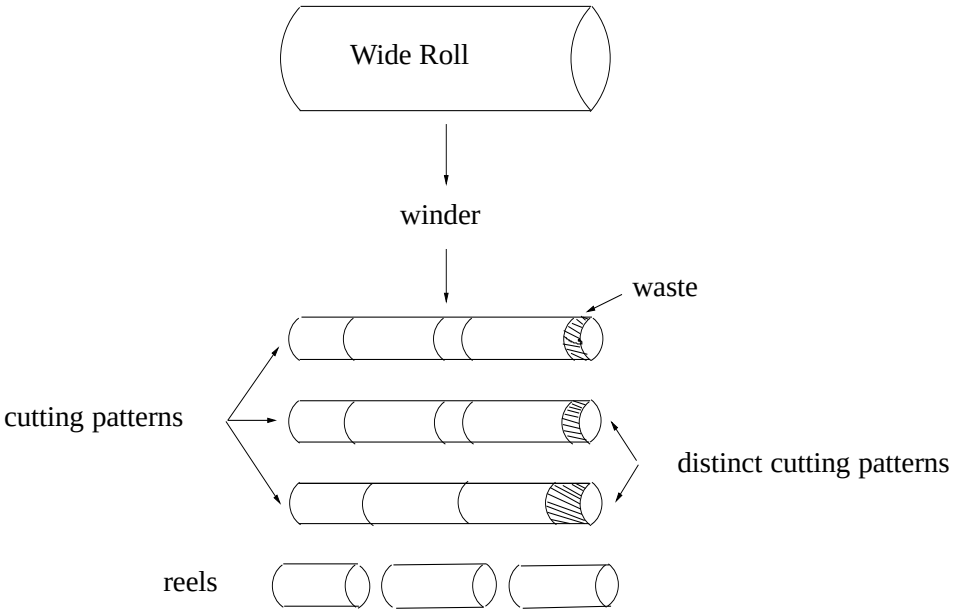


Figure 2: The cutting of paper rolls

shall be examined. We start with the standard cutting stock and bin packing problems. Variants are problems in which the production levels lie in a prescribed interval of tolerance, the multiple widths cutting stock problem for which stock pieces can be of different sizes, problems with additional technical constraints (taking into account side constraints issued from technical or managerial considerations in real-life industrial applications). We also present a formulation for the minimization of the number of distinct cutting patterns seen as a second objective once the waste is fixed to its optimal value. Finally, we give a brief literature review on the cutting stock problem. It will be completed by in-depth review of work specifically related to ours throughout the text.

Chapter 2 deals with various reformulations of the Cutting Stock Problem (CSP). We first consider different formulations for the knapsack subproblem. Each of these formulations leads to an explicit reformulation of the CSP and to an implicit reformulation that can be solved using a dynamic column generation procedure. We also review hybrid approaches that have been proposed in the literature. Furthermore, we develop alternative implicit reformulations modeling local exchanges in the solution, and we show how these exchanges imply a

stabilization effect in the column generation procedure, because they implicitly amount to adding dual constraints. Some of these models are valid reformulations while others define relaxations. These reformulations are compared from a theoretical point of view in terms of their linear relaxation (LP bounds) and integer solutions (primal bounds). Some of the formulations considered therein are original contribution of this thesis. Moreover, this chapter offers a classification of the possible formulations, comparing the strength of the LP dual bound and commenting on their practical interest.

Chapter 3 deals with the knapsack subproblem. The standard knapsack problem can be solved relatively efficiently using specialized algorithms. But, when solving the column generation reformulation to integrality, we use branching rules that can lead to modifications to the knapsack subproblem. To model properly the reduced cost after branching, we need to operate a binary decomposition of knapsack subproblem variables and introduce setup variables associated to each order. The focus of this Chapter is the study of the resulting model named multiple-class binary knapsack problem with setups. We show how to extend standard results for the knapsack problem to this more complex model. We characterize optimal solutions to its LP relaxation, we show how to obtain the LP bound using a greedy procedure, we propose a depth-first-search branch-and-bound algorithm (for the case with positive fixed cost) and dynamic programming procedures to solve this modified subproblem. This research gave rise to publication [32].

In chapter 4, we study in turn each step of a branch-and-price procedure for cutting stock problem and compare different implementation strategies through numerical tests. We present different modes of initialization for the column generation procedure. We compare stabilization techniques on several variants. In particular, we examine the contribution to stabilization of our original reformulation with built-in exchanges. We experiment with simple boxstep stabilization method and with dual variable smoothing techniques. We also make comparison between LP-based column generation and the use of the bundle method for solving the master. This study contributed to publication [6]. We also develop primal heuristics based on the column generation approach and

compared them to standard constructive heuristics. Then, we consider different strategies for generating columns (single versus multiple columns, exact versus heuristic subproblem solutions and a strategy of diversification). Finally, we test the individual contribution of our branching rules in converging to integer solution and increasing the dual bounds.

Chapter 5 is devoted to the study of industrial cutting problems that combine the difficulties of the CSP variants reviewed in Chapter 1. Their formulation is built on those presented in Chapter 2. A case study of the real-life problem encountered at the paper mill Condat is presented. The problem involves tolerance on production levels, technical restrictions on the cutting process and two optimization criteria (waste and number of setups). A second real-life application with a minimal run length constraint is studied. The best strategies derived from our experimental study of Chapter 4 are applied to these problems. They allow us to solve real-life instances with a generic branch-and-price code that relies on a commercial mixed integer programming (MIP) solver for master and subproblem solutions.

1

THE CUTTING STOCK PROBLEM AND ITS VARIANTS

1.1 STANDARD CUTTING STOCK AND BIN PACKING PROBLEMS

In the standard cutting stock problem, we consider that demands for cut pieces are fixed: let n be the number of orders to be cut. An order i, for $i = 1, \ldots, n$, is defined by its width w_i and its demand d_i. We assume a sufficient stock of identical wide rolls, indexed by $k, k = 1, \ldots, K$, whose width is denoted W, with $w_i \leq W \ \forall i$ and

$$\lceil \frac{\sum_{i=1}^{n} w_i \, d_i}{W} \rceil \leq K \leq \sum_{i=1}^{n} d_i \ .$$

The objective is to minimize the waste resulting from the cutting process. When the demand is fixed, it is equivalent to minimizing the number of wide rolls used as these two criteria differ by a constant (the total length of stock material used is equal to the total length of produced material plus the total waste).

The problem can be formulated in terms of integer variables x_{ik} representing the number of orders i cut in wide roll k, and binary variables y_k taking value 1 if wide roll k is used and 0 otherwise. A compact formulation (due to Kantorovich [19]) is thus:

$$Z^k = \min \sum_{k=1}^{K} y_k$$

$$[\text{F1}] \qquad \text{s.t.} \quad \sum_{k=1}^{K} x_{ik} \;\geq\; d_i \qquad\qquad i = 1, \ldots, n \tag{1.1}$$

$$\sum_{i=1}^{n} w_i \, x_{ik} \;\leq\; W \, y_k \qquad\qquad k = 1, \ldots, K \tag{1.2}$$

$$x_{ik} \in \{0, \ldots, u_i\} \qquad i = 1, \ldots, n \,; k = 1, \ldots, K \tag{1.3}$$

$$y_k \in \{0, 1\} \qquad\qquad k = 1, \ldots, K$$

where

$$u_i = \min\{d_i, \lfloor \frac{W}{w_i} \rfloor\} \tag{1.4}$$

is a natural upper bound for variables x_{ik}. Constraints (1.1) ensure the satisfaction of the demand for each order (one could use equality constraint but their relaxation into covering constraint leads to the same optimal solution value). Constraints (1.2) are knapsack constraints that ensure that cutting patterns are feasible.

A well known variant of this problem is when all orders demands are equal to one, i.e. $d_i = 1$ for $i = 1$ to n. This problem is known as the **bin-packing problem**. However in bin packing problems there are typically several items of the same width. These items should better be aggregated into a single item whose demand becomes $d_i > 1$ in order to avoid symmetry in modeling the problem. Hence, the bin packing problem can be understood as a cutting stock problem with small demand levels.

1.2 A VARIANT WITH INTERVALS ON PRODUCTION

In industrial applications, the production requirements are sometimes expressed with a tolerance, which translates into intervals of admissible production levels. Let $\underline{d_i}$ and $\overline{d_i}$ be respectively the lower and upper bounds on the production of order i. Then, the objective of minimizing the waste that occurred in the cutting process is no longer equivalent to minimizing the number of used stock rolls but must be stated explicitly. We give the full formulation for further reference. It is:

$$Z^R = \min \sum_{k=1}^{K} (W\, y_k - \sum_i w_i\, x_{ik})$$

$$[\text{F2}] \qquad \text{s.t.} \quad \sum_{k=1}^{K} x_{ik} \;\geq\; \underline{d_i} \qquad\qquad i = 1, \ldots, n \tag{1.5}$$

$$\sum_{k=1}^{K} x_{ik} \;\leq\; \overline{d_i} \qquad\qquad i = 1, \ldots, n \tag{1.6}$$

$$\sum_{i=1}^{n} w_i\, x_{ik} \;\leq\; W\, y_k \qquad\qquad k = 1, \ldots, k$$

$$x_{ik} \in \{0, \ldots, u_i\} \qquad i = 1, \ldots, n\,; k = 1, \ldots, K$$

$$y_k \in \{0, 1\} \qquad\qquad k = 1, \ldots, K$$

where $u_i = \min\{\overline{d_i}, \lfloor \frac{W}{w_i} \rfloor\}$. The satisfaction of the demand is now stated with two types of constraints: the covering (1.5) and the packing (1.6) constraints.

1.3 THE MULTIPLE WIDTH CUTTING STOCK PROBLEM

The multiple width cutting stock problem is a generalization of the cutting stock model [F1] (1.1) in which the stock is made of non identical wide rolls. The width of roll k is noted W_k. Then [F1] becomes:

$$Z^k = \min \sum_{k=1}^{K} y_k \tag{1.7}$$

[F3] s.t. $$\sum_{k=1}^{K} x_{ik} \geq d_i \qquad i = 1, \ldots, n \tag{1.8}$$

$$\sum_{i=1}^{n} w_i \, x_{ik} \leq W_k \, y_k \qquad k = 1, \ldots, K \tag{1.9}$$

$$x_{ik} \in \{0, \ldots, u_i\} \qquad i = 1, \ldots, n \, ; k = 1, \ldots, K$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, K$$

1.4 A VARIANT WITH TECHNICAL RESTRICTIONS

In the context of industrial production, there might be additional constraints, so called *side constraints*, imposed for technical reasons (like the characteristics of the machines used) or managerial reasons. Classical examples are a minimum width required for cutting patterns, and the number of orders cut in a wide roll cannot exceed the cut capacity of the winder. The minimum and the maximum widths to be cut are noted respectively W_{min} and W_{max} and the maximum cardinality of a cut set C. Furthermore we consider an interval on the demand because in real applications it is often the case.
It amounts to add additional constraints in $[F2]$:

$$Z^R = \min \sum_{k=1}^{K} (W \, y_k - \sum_i w_i \, x_{ik}) \tag{1.10}$$

$$[\text{F4}] \qquad \text{s.t.} \quad \sum_{k=1}^{K} x_{ik} \;\geq\; \underline{d_i} \qquad i = 1, \ldots, n \tag{1.11}$$

$$\sum_{k=1}^{K} x_{ik} \;\leq\; \overline{d_i} \qquad i = 1, \ldots, n \tag{1.12}$$

$$W_{min}\, y_k \;\leq\; \sum_{i=1}^{n} w_i\, x_{ik} \qquad k = 1, \ldots, K \tag{1.13}$$

$$\sum_{i=1}^{n} w_i\, x_{ik} \;\leq\; W_{max}\, y_k \qquad k = 1, \ldots, K \tag{1.14}$$

$$\sum_{i=1}^{n} x_{ik} \leq C \qquad k = 1, \ldots, K \tag{1.15}$$

$$x_{ik} \in \{0, \ldots, u_i\} \qquad i = 1, \ldots, n\,; k = 1, \ldots, K$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, K$$

where $u_i = \min\{\overline{d_i}, \lfloor \frac{W}{w_i} \rfloor\}$.

1.5 THE MINIMIZATION OF SET-UPS

The main objective in cutting stock problems is to minimize the waste. However other criteria are important in determining what a good production planning is. One of them is the minimization of setups, i.e. of the number of distinct cutting patterns used. Indeed, in real process cut, time is spent between each new pattern to cut and a waste is incurred in trial run to check the correct position of the knives.

This problem can be modeled by introducing the concept of cloning. The cutting pattern used for a wide roll k can be reproduced identically on other wide roll. A compact but non linear formulation for this problem is given below. It involves new integer variables z_k representing the number of times the cutting of wide roll k is cloned on other wide roll. Binary variables y_k do take a different

meaning here: $y_k = 1$ if wide roll k is cut using an original cutting pattern (not used yet) and that may be used as a model (a patron) for cutting other wide rolls, while $y_k = 0$ if either wide roll k is not used or it is used with a cutting pattern that is the clone of another wide roll. As it was the case in [F1] (1.1), the variables definition allows for multiple representation of the same solution obtained by permuting the role of the k indices.

The formulation takes the form:

$$Z = \min \sum_{k=1}^{K} y_k$$

$$\text{[F5]} \qquad \text{s.t.} \quad \sum_{k=1}^{K} x_{ik}\, z_k \;\geq\; d_i \qquad i = 1, \ldots, n \tag{1.16}$$

$$\sum_{i=1}^{n} w_i\, x_{ik} \;\leq\; W\, y_k \qquad k = 1, \ldots, K$$

$$z_k \;\leq\; K\, y_k \qquad k = 1, \ldots, K$$

$$x_{ik} \in \{0, \ldots, u_i\} \qquad i = 1, \ldots, n; k = 1, \ldots, K$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, K$$

$$z_k \in I\!N \qquad k = 1, \ldots, K$$

This formulation can be linearized by applying a two-stage transformation: decompose integer variables in binary components associated with powers of two and then define new variables to represent products of binary variables. This shall be done in Chapter 2.

If there are interval constraints on production, constraints (1.16) are replaced by:

$$\underline{d_i} \;\leq\; \sum_{k=1}^{K} x_{ik}\, z_k \;\leq\; \overline{d_i} \qquad i = 1, \ldots, n$$

A constraint can be added to bound the waste, either by bounding the total number of wide rolls used :$\sum_{k=1}^{K} z_k \leq U$ or, when production is not fixed, the explicit waste

$$\sum_{k=1}^{K}(W - \sum_{i=1}^{n} w_i \, x_{ik}) \, z_k \leq R \, . \tag{1.17}$$

In the former case, one better redefine $K = U$ instead of adding constraint $\sum_{k=1}^{K} z_k \leq U$. In a hierarchical optimization approach where the priority is the waste minimization while setup minimization is a secondary objective, U (resp. R) is computed first using model [F1] (1.1) (resp. [F2]), then one solves [F5] in a second stage.

1.6 REVIEW OF THE LITERATURE

The one-dimensional cutting stock problem has been intensively studied in the two last decades. In [12], Dyckhoff developed a typology of cutting and packing problems according to the dimensionality, the kind of assignment, the assortment of large objects and the assortment of small items. A more recent improved typology is presented in [46]. It is based on the Dyckhoff's typology, defining new problem categories and it gives a review of all recent papers in the cutting and packing problems area.

Exact approaches to solve this problem make use of column generation. Vance in [39] developed a branch-and-price algorithm using the Dantzig-Wolfe reformulation, and branching directly on variables associated with the choice of cutting patterns. Different branching schemes were proposed by Vanderbeck in [40]. Scheithauer et al.([33]) developed a cutting plane algorithm. A different approach was used by Valério de Carvalho who worked with an arc flow formulation with side constraints in [35].

The cutting stock problem admits different formulations that are well suited for column generation. The choice of a particular formulation has sometimes been

motivated by the ease to implement branching or by the stabilization effect that the formulation can have on the column generation procedure. In [38], Valério de Carvalho gives a survey of models for the one dimensional cutting stock and packing problems: the integer linear formulation of Kantorovitch described in [19], the Dantzig-Wolfe reformulation that gives stronger dual bounds, the position indexed model, some alternative one-cut models and an extended model obtained by adding extra columns in the Gilmore-Gomory model. A modeling as a Vehicle Routing problem is also considered. In [10], Desrosiers and Lübbecke give a review of the usual formulations that are well suited for applying a column generation procedure to the cutting stock problem, in particular the Gilmore and Gomory model ([14]), and the arc flow formulation proposed by Valério de Carvalho in [38]. Further pointers to the literature on formulations for the CSP are provided in Chapter 2.

With regards to implementation strategies, the literature has mainly focused on the issue of stabilization. In his thesis, Neame, [30], presents a new simple technique to stabilizing the column generation procedure by smoothing the dual values and he tests it to solve the binary cutting stock problem. He makes comparisons with other main approaches (varying box size step method, a linear norm penalty method, an hybrid method (du Merle et al. [11])) and then shows that it performs well on the binary cutting stock problem, the average time on difficult instances is reduced by a factor of three. In [37], Valerio de Carvalho proposes to introduce dual cuts in the dual formulation: he initializes the column generation procedure with valid cuts (see chapter 2) and shows that they accelerate the procedure on bin-packing problems.

In recent work on dealing with industrial cutting stock problem, few papers use exact methods. Johnson et al., in [18], propose a model combining skiving (joining smaller rolls to form larger rolls) and the one dimensional cutting stock problem. Their model consists in generating the cutting patterns while minimizing the waste and takes into account two technical constraints (minimum width used in a pattern and a bound on the number of cut pieces in a pattern). Their solution method combines heuristics and the use of the LP formulation. In [8], Correira, Oliveira and Ferreira describe two models (minimizing the waste)

for the same problem with additional technical constraints. In a first stage, they generate a priori cutting patterns to be included in a LP formulation. Then, they obtain an integer solution to the LP solution using a rounding heuristic procedure. Lee, in [24], proposes “in situ” column generation approach that we discuss in Chapters 2 and 5.

2

REFORMULATIONS AND COLUMN GENERATION

The compact formulations of Section 1 suffer from several drawbacks. One of these drawbacks is the weakness of their LP relaxation: the lower bound obtained by relaxing the integrability constraints is typically weak. For the standard cutting stock problem it has the value

$$Z_{LP} = \frac{\sum_{i=1}^{n} w_i d_i}{W} \ .$$

When the cutting process involves a lot of waste, this bound can be far from the optimum. For example, when $w_i = \frac{W}{2} + \varepsilon, \forall i$, the optimal solution is $\sum_{i=1}^{n} d_i$ and the gap between this optimal solution and the lower bound approaches 50%. The second weakness of Section 1 models is the symmetry inherent to indexing variables with k: several equivalent solutions can be obtained by exchanging cutting patterns between wide rolls, i.e. by permuting the k indexes in the solution of the compact formulation [F].

Better formulations that avoid (in part) the above drawbacks can be derived by exploiting the structure of the problem. The knapsack constraints (1.2) represent

a block diagonal matrix, while the covering constraints (1.1) act as linking constraints. The compact formulation exhibits symmetry because it makes no use of the fact that the knapsack subproblems are identical. On the other hand, stronger dual bound can be obtained by convexification of the knapsack subproblem (using the Dantzig-Wolfe reformulation principle [9]). Indeed, efficient algorithms (although not polynomial) are known for the knapsack problem (either dynamic programming or specialized branch-and-bound methods) that can be exploited to carry on an implicit reformulation of the cutting stock problem in terms of the weights associated with knapsack subproblem solutions. Such extensive formulation is to be solved using dynamic column generation. Alternatively, an explicit reformulation can be obtained using the variable redefinition technique of [29].

The alternative reformulations are developed in this chapter. We begin by considering the various formulations of the integer knapsack subproblem as they underly the different formulations of the cutting stock problem. Indeed, a variable change in the subproblem can be applied to the whole problem to give an explicit reformulation. Moreover, each subproblem formulation calls for its own solution method. Then, we consider successively explicit and implicit reformulations for the standard cutting stock problem. Finally, we review in turn the different variants and say how to adapt subproblem and global problem reformulations.

2.1 FORMULATIONS FOR THE KNAPSACK SUBPROBLEM

The sub-system X, whose solutions are valid cutting patterns, is defined by constraints (1.2) and (1.3). Thus, the natural formulation for an optimization Integer

Subproblem (ISP) over X is

$$\max_{x \in X} \sum_{i=1}^{n} p_i \, x_i$$

$$\text{[ISP1]} \quad \text{where } X = \{ \sum_{i=1}^{n} w_i \, x_i \;\; \leq \;\; W$$

$$x_i \;\; \leq \;\; u_i \quad i = 1, \ldots n$$

$$x_i \;\; \in \;\; I\!N \quad i = 1, \ldots n\}$$

It can be solved by using a dynamic programming procedure in $O(nW^2)$, or a specialized branch-and-bound algorithm [28]. However, for the latter, it can be better to transform the subproblem into a 0-1 knapsack problem, which can be done polynomially. Moreover, the binary decomposition shall also be useful in our definition of branching constraints. To avoid introducing symmetric representations of some solutions in the transformation, [42] showed that it is better to use a multiple class binary knapsack model.

Let $n_i = \lfloor \log_2(\overline{d}_i) \rfloor + 1$. We apply the change of variable:

$$x_i = \sum_{j=1}^{n_i} 2^{j-1} \, x_{ij} \quad \forall i = 1, \ldots, n \tag{2.1}$$

with $x_{ij} \in \{0, 1\}$. We denote by $m_{ij} = 2^{j-1} \quad \forall j = 1, \ldots, n_i$, the multiplicities of item i. The Binary Subproblem (BSP) takes the form of a **multiple class binary knapsack**:

$$\max_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{n_i} p_{ij} \, x_{ij}$$

$$\text{[BSP1]} \quad \text{where } X = \{ \sum_{i=1}^{n} \sum_{j=1}^{n_i} w_i \, m_{ij} \, x_{ij} \;\; \leq \;\; W$$

$$\sum_{j=1}^{n_i} m_{ij} \, x_{ij} \;\; \leq \;\; u_i \quad i = 1, \ldots n$$

$$x_{ij} \;\; \in \;\; \{0, 1\} \quad i = 1, \ldots n, \; j = 1, \ldots n_i\}$$

Reformulation [BSP1] does not allow to improve the LP dual bound. It admits the same set of LP solutions than [ISP1], as any solution to [ISP1] can be decomposed in an LP solution to [BSP1] using transformation:

$$x_{ij} = \frac{x_i}{2^{n_i} - 1} \quad \forall i = 1, \ldots, n \tag{2.2}$$

while the reverse transformation is given by (2.1).

The unbounded version, where we ignore the bounds (1.4), allowing a pattern cutting more pieces than the demand, is easier to solve. The dynamic programs takes $O(nW)$ operations, while the 0-1 transformation now leads to a standard binary knapsack problem. For further reference let [uISP1] (resp. [uBSP1]) denotes model [ISP1] (resp. [BSP1]) without constraints (2.1) (resp. (2.2)).

The third formulation considered here is for the unbounded version of the knapsack subproblem. The problem can be formulated as a **longest path problem** in an acyclic network that underlies the dynamic programming solution method. Assume that all items have different width w_i (for otherwise they can be aggregated into a single item). We define a graph $G = (N, A)$, where the node set is $N = 0, \ldots, W, W + 1$, each node representing a feasible level of capacity usage and the arc set is defined by $A = \cup_i A(i) \cup \{(u, W + 1) : u = 0, \ldots, W\}$ where $A(i) = \{(u, v) : 0 \leq u < v \leq W$ with $v - u = w_i\}$ are arcs representing the cutting of a piece of item i, while the other arcs represent waste. A valid cutting pattern is a path in this directed acyclic graph.

Let x_{uv} be the binary decision variable associated with arc $(u, v) \in A$: $x_{uv} = 1$ if a knife is set in position u and another in position v yielding a cut piece of size $w_i = v - u$. The reformulation in these variables, denoted as the uncapacitated

Flow Subproblem (uFSP), takes the form:

$$\max_{x \in X} \sum_{i=1}^{n} p_{uv}\, x_{uv}$$

$$\text{[uFSP1]} \quad \text{where } X = \{ \sum_{u \in N} x_{uv} = \sum_{w \in N} x_{vw} \;\; \forall v \in N \setminus \{0, W+1\} \quad (2.3)$$

$$\sum_{v \in N} x_{0v} = 1$$

$$\sum_{v \in N} x_{vW+1} = 1$$

$$x_{uv} \in \{0,1\} \;\; \forall (u,v) \in A$$

Constraints (2.3) are flow conservation constraints that ensure the feasibility of the cutting pattern.

Observe that the network flow model suffers from symmetry because different paths could correspond to the same production of cut pieces. However, Valério de Carvalho shows in [36] how to reduce the symmetry. He considers a subset of arcs using the following criteria: assuming that items are sorted in non increasing order of their width, then an arc representing an item of smaller width has its head on the tail of an arc corresponding to a larger item. With this streamlined definition of the support graph, a given subproblem solution has a unique path representation.

On the other hand, [uFSP1] is a stronger formulation than [uISP1] and [uBSP1]. Each LP solution of this subproblem can be transformed in a LP solution for [uISP1]:

$$x_i = \sum_{(u,v) \in A(i)} x_{uv}$$

and one can show that if the vector of x_{uv} is a feasible solution of the LP relaxation of [uFSP1], then the associate x_i vector is feasible for the LP relaxation of [uISP1]. But, it exists LP solutions to [uISP1] that cannot be represented as LP solutions in [uFSP1], as seen in example 1.

Example 1 *Let* $n = 2$, $W = 4$, $w_1 = 3$ *and* $w_2 = 2$.
The solution $x_1 = 1$, $x_2 = \frac{1}{2}$ *cannot be presented in [uFSP1] because the only arc leaving node* 3 *is a waste arc to node* $W + 1$.

In fact, [uFSP1] is an ideal formulation that has the "integrality property", since it is a flow problem. However it is less compact: it involves a pseudo polynomial number of variables ($O(nW)$) and constraints ($O(W)$) instead of n variables and one constraint for unbounded version of model [uISP1].

In theory, the ideal formulation for the sub-system can also be obtained using the definition of the convex hull of the integer solution. Let Q be the set of feasible cutting patterns, i.e. solutions of the sub-system X, where X is defined using formulation (u)ISP1, (u)BSP1, or uFSP1. Thus, Q denotes the enumerated set while X is a mathematical programming representation of the discrete solutions:

$$X = \{x^q\}_{q \in Q}$$

We say that x^q is the solution $x \in X$ associated with the feasible cutting pattern $q \in Q$. Then, the subproblem could be reformulated as

$$\max_{x \in X} p\, x$$

$$\text{[ESP1]} \quad \text{where } X = \{x \;=\; \sum_{q \in Q} x^q \lambda_q \tag{2.4}$$

$$\sum_{q \in Q} \lambda_q \;=\; 1 \tag{2.5}$$

$$\lambda_q \;\in\; \{0,1\} \forall q \in Q\} \; .$$

Its LP relaxation gives $conv(X)$, by definition. [ESP1] assumes we have enumerated exhaustively all solutions $x \in X$. Of course this is not realistic. This reformulation is only to be used implicitly in applying to the cutting stock problem what is known as a Dantzig-Wolfe reformulation.

2.2 EXPLICIT REFORMULATIONS

Each reformulation of the knapsack sub-systems leads to reformulations of the global standard cutting stock problem. Formulation [ISP1] gives rise to the compact formulation [F1] (1.1) presented in Section 1, which we shall be more precisely denoted by F1(ISP1). Using [BSP1] yields an alternative formulation:

$$
\begin{aligned}
& & Z = \quad \min \quad & \sum_{k=1}^{K} y_k & & \\
[\text{F1(BSP1)}] \quad \text{s.t.} \quad & \sum_{k=1}^{K} \sum_{j=1}^{n_i} m_{ij}\, x_{ijk} & \geq \quad & d_i & \forall i & \\
& \sum_{i=1}^{n} \sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ijk} & \leq \quad & W\, y_k & \forall k & \\
& \sum_{j=1}^{n_i} m_{ij}\, x_{ij} & \leq \quad & u_i & \forall i & \\
& y_k & \in \quad & \{0,1\} & \forall k & \\
& x_{ijk} & \in \quad & \{0,1\} & \forall i,\, \forall j,\, \forall k &
\end{aligned}
$$

Its interest is limited because it is not any stronger than F1(ISP1), and it does not help to avoid symmetry. Moreover it involves a larger number of variables. We mentioned it for further reference when it comes to defining branching constraints based on these binary variables.

Formulation [uFSP1], however, leads to an interesting explicit reformulation of the cutting stock problem. Valério de Carvalho ([38]) introduced this arc-flow model for bin packing problem. Here, variables x_{uvk} of each sub-system in the form [uFSP1] can be aggregated into

$$x_{uv} = \sum_{k} x_{uvk}$$

where x_{uv} now represents the number of wide roll cuttings where consecutive knifes are placed in positions u and v (i.e. the number of times the item of weight $w_i = v - u$ is placed at a distance of u of the origin in a cutting pattern). Observe that such aggregation could not be carried out in F1(ISP1) or F1(BSP1) because the disaggregate value was needed to formulate the knapsack constraints, while here the knapsack constraint is built into the definition of a flow from node 0 to W.

Thus, the reformulation takes the form:

$$
\begin{array}{lllll}
 & & \min & z & \\
[R1] & \text{s.t.} \quad \displaystyle\sum_{(u,v)\in A} x_{uv} - \sum_{(v,w)\in A} x_{vw} & = & 0 & \forall v \in N \setminus \{0, W\} \\
 & \displaystyle\sum_{v\in N} x_{0v} & = & z & \\
 & \displaystyle\sum_{v\in N} x_{vW} & = & z & \\
 & \displaystyle\sum_{(u,v)\in A(i)} x_{uv} & \geq & d_i \quad \forall i & (2.6) \\
 & x_{uv} & \in & I\!N \quad \forall (u,v) \in A &
\end{array}
$$

Reformulation [R1] has pseudo polynomial size since it involves $O(nW)$ variables and $O(W)$ constraints. But it is stronger than [F1(ISP1)] or [F1(BSP1)] as it makes use of an ideal formulation for the subproblem. Moreover the above mentioned aggregation allows to avoid the symmetry that resulted from the interchange of k indexes. However, it introduces a new symmetry. Although the reduction to the support graph proposed by Valerio de Carvalho completly eliminates the duplication of representation of solutions at the subproblem level, it remains symmetry at the global level: the paths flow can be recombined in multiple ways, much more for the cutting stock problem. A given solution to R1 admits different representations as shown by the following example problem.

Example 2 *Let $n = 2$ items with respective widths and demands $w_1 = 1, d_1 = 6$ and $w_2 = 2, d_2 = 3$, and let $W = 4$.*

The patterns $q_1 = (4, 0)$, $q_2 = (0, 2)$ and $q_3 = (2, 1)$ represented in figure 2.1 by the respective paths: $(0, 1, 2, 3, 4)$, $(0, 2, 4)$ and $(0, 2, 3, 4)$, respect the reduction criteria. However, combining them, the pattern q_3 admits another representation : $\{0, 1, 2, 4\}$, that gives rise to the same solution.

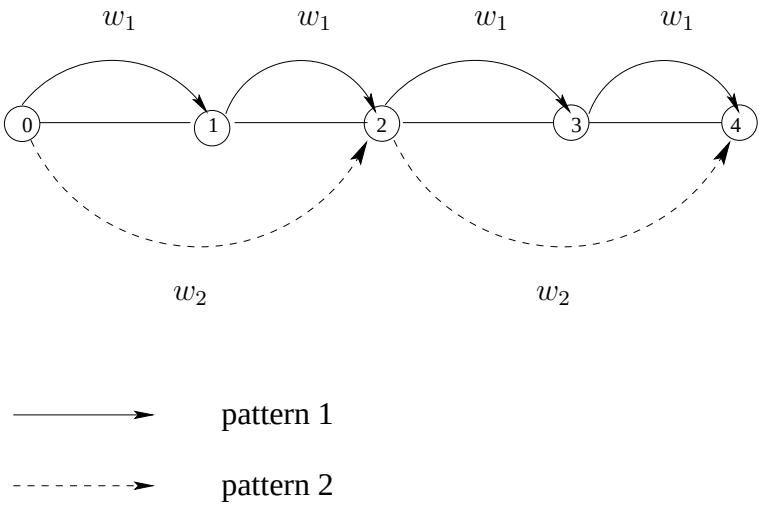


Figure 2.1: Symmetry in the arc flow formulation

The motivation for this formulation is also to provide new variables on which to branch.

2.3 IMPLICIT REFORMULATIONS AND COLUMN GENERATION

Formulation [ESP1] leads to a reformulation in terms of variables $\lambda_q^k = 1$ if subproblem solution x^q is chosen for subproblem k and zero otherwise. It takes the

form:

$$Z^M = \min \sum_{k=1}^{K} \sum_{q \in Q} \lambda_q^k \tag{2.7}$$

[M1Disag] s.t. $$\sum_{k=1}^{K} \sum_{q \in Q} x_i^q \, \lambda_q^k \;\geq\; d_i \qquad i = 1, \ldots, n \tag{2.8}$$

$$\sum_{q \in Q} \lambda_q^k \;\leq\; 1 \qquad k = 1, \ldots, K \tag{2.9}$$

$$\lambda_q^k \in \{0, 1\} \quad \forall q \in Q,\ k = 1, \ldots, K \tag{2.10}$$

where the convexity constraints $\sum_{q \in Q} \lambda_q^k = 1$ have been relaxed into $\sum_{q \in Q} \lambda_q^k \leq 1$ because the null cutting pattern is a feasible solution to X of zero cost.

Because the subproblems are identical, variables λ_q^k can be aggregated by defining variables $\lambda_q = \sum_k \lambda_q^k \;\forall q \in Q$. Thus, λ_q represents the number of times the cutting pattern $q \in Q$ is chosen in the solution. The aggregation allows to eliminate the symmetry. Thus, when X is defined by [ISP1] or [uISP1], the reformulation of the standard cutting stock problem takes the form:

$$Z^M = \min \sum_{q \in Q} \lambda_q \tag{2.11}$$

[M1] s.t. $$\sum_{q \in Q} x_i^q \, \lambda_q \;\geq\; d_i \qquad i = 1, \ldots, n \tag{2.12}$$

$$\sum_{q \in Q} \lambda_q \;\leq\; K \tag{2.13}$$

$$\lambda_q \in I\!N \qquad \forall q \in Q \tag{2.14}$$

The convexity constraint (2.13) is not binding because K is an overestimate of the number of stock pieces used, so it can be dropped. Due to its large (exponential) number of variables, this reformulation is to be solved using an integer programming column generation procedure (branch-and-price). In this context, re-formulation [M1] is called the **master** program.

The demand covering constraints (2.12) take a different form for alternative representation of X. When [BSP1] or [uBSP1] is chosen, they take the form

$$\sum_{q \in Q} \sum_{j} m_{ij}\, x_{ij}^{q}\, \lambda_q \;\geq\; d_i \quad \forall i$$

while for [uFSP1], they take the form

$$\sum_{q \in Q} \sum_{uv : v = u + w_i} x_{uv}^{q}\, \lambda_q \;\geq\; d_i \quad \forall i\ .$$

2.3.1 THE COLUMN GENERATION PROCEDURE

The dynamic column generation procedure consists in iteratively solving the master LP restricted to a subset of columns, and, in the pricing procedure, search for missing columns with negative reduced cost by solving an optimization subproblem over X. The optimal dual solution of the restricted master LP is used to define the reduced cost of a generic column. At the root node, it works as follows. Let π be the dual variables associated to the master constraint (2.12). The specific form of the reduced cost of a cutting pattern depends on the variable definitions. Using the variables of [ISP1], the reduced cost is

$$\overline{c}_q = 1 - \sum_{i=1}^{n} \pi_i\, x_i^q\ .$$

2.3.2 THE COLUMN GENERATION SUBPROBLEM

The pricing subproblem is solved in search for the most negative reduced cost over X. Using representation [ISP1], it takes the form:

$$\xi(\pi) = \max\{\sum_{i=1}^{n} \pi_i\, x_i : \sum_{i=1}^{n} w_i\, x_i \leq W,\ x_i \leq\ u_i \text{ and } x_i \in\ I\!N \text{ for } \ i = 1, \ldots n\}$$

A solution to the subproblem is a column that should be added to the master problem if it has a negative reduced cost. When the optimal reduced cost is zero, the current solution to the restricted master is proved optimal for the unrestricted master.

2.3.3 THE LAGRANGIAN DUAL BOUND

Any dual bound on the subproblem gives rise to a Lagrangian dual bound for the master. Indeed, applying Lagrangian relaxation to the covering constraints (2.12), using the Lagrangian multipliers π, yields a Lagrangian bound:

$$L(\pi) = \min \sum_{q \in Q} (1 - \sum_{i=1}^{n} \pi_i \, x_i^q) \, \lambda_q + \sum_{i=1}^{n} \pi_i \, d_i$$

$$\sum_{q \in Q} \lambda_q \; \leq \; K \tag{2.15}$$

$$\lambda_q \in \; I\!N \qquad\qquad \forall q \in Q$$

whose solution is

$$L(\pi) = \sum_{i=1}^{n} \pi_i \, d_i + \min\{K \, (1 - \xi(\pi)), 0\}$$

If the pricing subproblem is not solved exactly, any dual bound $\underline{\xi}(\pi)$ on the subproblem value $\xi(\pi)$ can be used in the above expression to define a valid lower bound on the master problem. The best bound encountered in the course of the column generation procedure is recorded:

$$LB = \max_t L(\pi^t)$$

where π^t is the dual solution at iteration t. Moreover, for an integer objective value, this bound can be rounded up: $LB = \max_t \lceil L(\pi^t) \rceil$.

2.3.4 TERMINATION CRITERIA

The column generation procedure stops:
(i) when no more negative reduced cost columns are found or,
(ii) when the current Lagrangian dual bound LB is greater or equal to the current value of the restricted master LP, or

(iii) when the current Lagrangian dual bound LB allows to prune the current branch-and-bound node, i.e.

$$LB \geq Z_{INC}$$

where Z_{INC} denotes the cost of the incumbent integer solution.

2.3.5 THE DUAL MASTER PROGRAM

Seen in the dual space, the column generation procedure is a cutting plane algorithm for the dual of the master LP. It is known as *Kelley's cutting plane method* [20]. The dual problem takes the form:

$$\max \sum_{i=1}^{n} d_i \, \pi_i - K \, \sigma \tag{2.16}$$

$$\text{[D1]} \qquad \text{s.t.} \quad \sum_{i=1}^{n} x_i^q \, \pi_i - \sigma \leq 1 \qquad \forall q \in Q \tag{2.17}$$

$$\pi_i \geq 0 \qquad i = 1, \ldots, n$$

$$\sigma \geq 0$$

The study of the dual is important because the dual multipliers play an important role in the column generation process, for the pricing of columns and the convergence of the algorithm.

This dual problem is the LP form of the *dual Lagrangian problem*:

$$\theta = \max_{\pi \geq 0} L(\pi) \tag{2.18}$$

Indeed,

$$\begin{aligned}
\theta &= \max_{\pi \geq 0} L(\pi) \\
&= \max_{\pi \geq 0} \sum_{i=1}^{n} d_i \, \pi_i + \min\{K \, (1 - \xi(\pi)), 0\} \\
&= \max \sum_{i=1}^{n} d_i \, \pi_i - K \, \sigma \\
&\qquad \text{s.t } -\sigma \leq 1 - \xi(\pi) \\
&\qquad\qquad \pi \geq 0 \\
&\qquad\qquad \sigma \geq 0 \\
&= \max \sum_{i=1}^{n} d_i \, \pi_i - K \, \sigma \\
&\qquad \text{s.t } \sum_{i=1}^{n} x_i^q \, \pi_i - \sigma \; \leq \; 1 \quad \forall q \in Q \\
&\qquad\qquad \pi \geq 0 \\
&\qquad\qquad \sigma \geq 0
\end{aligned}$$

where σ stands for the opposite of $(1 - \xi(\pi))^{-}$. Also note that the Lagrangian bound that results from dualizing constraints (1.1) in [F1] is the same as the above.

We introduce the restricted dual function $L^k(\pi)$ as being the restriction of $L(\pi)$ defined in (2.15) to the subset of columns obtained up to the k^{th} iteration, i.e.

$$\begin{aligned}
L(\pi) = \min & \sum_{q \in Q^k} (1 - \sum_{i=1}^{n} \pi_i \, x_i^q) \, \lambda_q + \sum_{i=1}^{n} \pi_i \, d_i \\
& \sum_{q \in Q^k} \lambda_q \; \leq \; K \qquad (2.19) \\
& \lambda_q \in \; I\!N \qquad\qquad \forall q \in Q^k
\end{aligned}$$

The associated restricted dual master problem takes the form:

$$\begin{aligned} \theta^k &= \max_{\pi \geq 0} L^k(\pi) \\ &= \max \sum_{i=1}^{n} d_i \, \pi_i - K \, \sigma \\ &\qquad \text{s.t} \sum_{i=1}^{n} x_i^q \, \pi_i - \sigma \; \leq \; 1 \quad \forall q \in Q^k \\ &\qquad\qquad \pi, \sigma \geq 0 \end{aligned} \tag{2.20}$$

At each iteration k of Kelley's cutting plane procedure, one maximizes θ^k and checks whether the resulting dual vector π^k is feasible for the unrestricted master by searching a violated inequality. For this, we solve the subproblem $\xi(\pi^k)$ that will provide the most violated inequality. If there is no violated inequality, π^k is feasible and therefore optimal for the unrestricted master.

2.3.6 STRENGTH OF THE DUAL BOUND

Standard Lagrangian duality theory [13] tells us that the master LP yields a bound equivalent to solving

$$\min\{\sum_{k=1}^{K} y_k : \sum_{k=1}^{K} x_{ik} \; \geq \; d_i \; \forall i, \; x_k \in conv(X), x_{ik} \leq u_i y_k \forall i, k, x_{ik}, y_k \geq 0 \forall i, k\} \; .$$

I.e. using Dantzig-Wolfe reformulation defines a master LP that is equivalent to an implicit convexification of the subproblem. The quality of this bound depends on the specific definition of X: [ISP1] and [BSP1] yield the same bound LD, while all three unbounded knapsack representations [uISP1], [uBSP1] and [uFSP1] yield a weaker bound uLD. As formulation [uFSP1] has the integrality property, formulation [R1] also yields bound uLD. In the sequel, we refer to *proper columns* to denote the solutions of a bounded knapsack subproblem, while solutions of unbounded version are *unproper*. This terminology was introduced in [45]. Using *proper columns* yields the stronger bound LD.

Finally, observe that applying a Dantzig-Wolfe reformulation to [R1] letting the covering constraints (2.6) in the master, results in a master formulation of type [M1] with the use of subproblem [uFSP1].

2.3.7 BRANCHING SCHEMES

Formulation [M1] is solved with a branch-and-price method. At each node of a branch-and-bound tree, if the current node cannot be pruned by bounds, infeasibility or optimality, then branching must takes place to eliminate the fractional solution. Branching on individual fractional variable λ_q is not appropriate. In [41] different branching schemes are studied where fractional solutions are eliminated using disjunctive constraints of the form:

$$\sum_{q \in \hat{Q}} \lambda_q \leq \lfloor \alpha \rfloor \tag{2.21}$$

or

$$\sum_{q \in \hat{Q}} \lambda_q \geq \lceil \alpha \rceil \tag{2.22}$$

where $\hat{Q} \subset Q$ and $\alpha = \sum_{q \in \hat{Q}} \overline{\lambda}_q$ for the current fractional solution $\overline{\lambda}$. The specific definitions of $\hat{Q}$ that happen to be useful in practice are described below for each formulation.

With formulation M1(ISP), branching constraints are difficult to formulate, hence the motivation for introducing the binary decomposition in the subproblem. For formulation M1(BSP), we use the following branching rules

1. $\hat{Q} = \{q \in Q : x_i^q > 0\}$, i.e. the number of used patterns involving item i must be integer. To implement this scheme, we need to introduce, in the subproblem, binary variables $y_i = 1$ if $x_i > 0$ and zero otherwise. The dual variables associated with the corresponding branching constraint (2.21) or (2.22) define a setup cost in the subproblem objective function

2. $\hat{Q} = \{q \in Q : x_{ij}^q = 1\}$, i.e. the number of used patterns involving the component of multiplicity j in the binary decomposition of item i must be integer. The dual variables associated with the corresponding branching constraint (2.21) or (2.22) define an extra cost associated to variable x_{ij} in the subproblem objective function.

3. $\hat{Q} = \{q \in Q : x_i^q > 0 \text{ and } x_j^q > 0\}$, i.e. the number of columns involving two specific items i and j must be integer. To implement this scheme, we need to introduce, in the subproblem, binary variables $y_{ij} = 1$ if $y_i = y_j = 1$ and zero otherwise for all pairs of items $i < j$. The proper value of y_{ij} variables is enforced by adding constraints

$$y_{ij} \geq y_i + y_j - 1 \quad \forall i, j : i < j$$

$$y_{ij} \leq y_i \quad \forall i, j : i < j$$

$$y_{ij} \leq y_j \quad \forall i, j : i < j$$

These three branching rules do not guarantee that any fractional solution can be separated. However, they were sufficient to eliminate all fractional solutions in the numerical experimentation reported in Chapter 4.

For M1(uFSP), the branching rule is

- $\hat{Q} = \{q \in Q : x_{uv}^q = 1\}$, i.e. the number of columns involving a particular arc (u, v) (or equivalently the total flow on arc (u, v)) is forced to be integer for all arcs. The dual variables associated with the corresponding branching constraint (2.21) or (2.22) come as an extra arc cost in the subproblem objective function.

This branching rule alone is enough to guarantee that an integer solution is obtained. Indeed, if the flow on each arc is integer, the flow decomposition

theorem guarantees that this flow can be decomposed into integer flow on path from node 0 to node W. Each of this path defines a feasible pattern, the flow on the path defines the number of time this pattern is chosen. This path flow solution is therefore a feasible integer solution for the CSP. Thus apparently, branching is easier to formulate and enforced under the arc flow formulation approach. It is to be reminded however that there is a pseudo-polynomial number of arc flow variables. Moreover, the arc flow formulation allows for different representation of a given solution and that branching efficiency will suffer from this drawback.

2.4 OTHER APPROACHES AND FORMULATIONS

Valério de Carvalho ([38]) applied an hybrid method to the bin packing problem: he solves it using "implicitly" the above column generation formulation [M1] with subproblem [uFSP1], but translates the columns for [M1] in variables for formulation [R1] and obtains the next set of dual prices by solving the restricted LP formulation [R1]. This requires adding variables and flow conservation constraint dynamically to [R1]. The procedure is not equivalent to working with formulation M1(uFSP1), as we can see in example 3, the arc flow formulation allows to define implicitly path flows that cannot be obtained as convex combination of the paths generated so far.

Example 3 *Let* $n = 2$, $W = 3$, $w_1 = 2$ *and* $w_2 = 1$.
We consider the following cutting patterns: $q_1 = (1, 0),\ q_2 = (0, 3)$. *In figure 2.2 the solution is represented as path flows, the full line corresponds to the first cutting pattern and the dotted path to the second one.*

Then, the flow $x_{12} = 0,\ x_{23} = 0,\ x_{13} = 1$ *and* $x_{34} = 1$ *that would lead to the cutting pattern* $q_3 = (1, 1)$ *is an implicit solution of the arc flow representation but it cannot be obtained by listing a convex combination of patterns* q_1 *and* q_2.

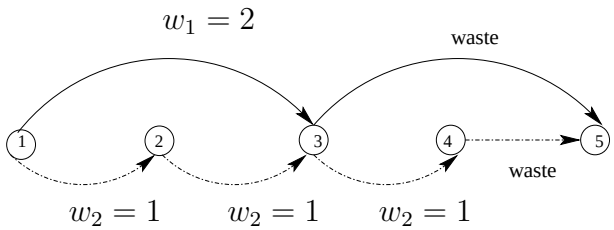


Figure 2.2: Path flow representation

Jon Lee ([24]) also introduced an hybrid method for the cutting stock problem (that he called "*in situ*" column generation): it is an hybridation of a method based on the compact formulation [F1] (1.1) and the column generation [M1]. In fact Lee proposed this approach for a variant with technical constraints and minimization of setups (he did not solve the model exactly neither). Here we just introduce the underlying idea on the standard CSP. The first cutting pattern is expressed in the original variables while the others are in the space of the column generation reformulation. Thus the formulation is:

$$\min \sum_{q \in Q} \lambda_q + y \tag{2.23}$$

$$\sum_{q \in Q} x_i^q \, \lambda_q + x_i \geq d_i \; i = 1, \ldots, n \tag{2.24}$$

$$\sum_{q \in Q} \lambda_q \leq K - 1 \tag{2.25}$$

$$\lambda_q \in I\!N \qquad \forall q \in Q \tag{2.26}$$

$$\sum_i w_i \, x_i \leq W \, y \tag{2.27}$$

$$x_i \in I\!N \qquad \forall i \tag{2.28}$$

$$y \in \{0, 1\} \, . \tag{2.29}$$

The hybrid procedure works as follows: at iteration k, one has a given subset of columns Q^k in the hybrid formulation; one solves the above integer problem restricted to Q^k to optimality and record the solution x as a new column before re-iterating. The stopping criteria to obtain an exact solution are not clear. The interest of using such approach, according to Lee, is to generate columns that are complementary to existing columns with regards to the IP solution. Observe that each generation of a column demands the solution of the IP master program to optimality.

To be exhaustive we should also mention the acyclic capacitated VRP model introduced by Ben Amor ([5]). Assuming that items are sorted in non increasing order of their width ($w_i \leq w_{i+1}\ i = 1, \ldots, n$), the network is defined as follows:

- for each item i, a set of $n_i + 1$ nodes, with $n_i = \left\lfloor \frac{W}{w_i} \right\rfloor$ noted $i_1, \ldots, i_{n_i}, i_0$, and a set of associated arcs (i_v, i_{v+1}) for $v = 1, \ldots, n_i - 1$ and (i_v, i_0) for $v = 1, \ldots, n_i$. Thus a path between nodes i_1 and i_0 define the number of items i that are chosen in a cutting pattern.

- a set of arcs between two items (i_0, j_1) such that $(w_j > w_i)$ and $w_i + w_j \leq W$,

- for each pattern k a starting and an ending nodes noted s_k and e_k, and a set of starting (respectively ending) arcs defined as (s_k, i_1) (respectively (i_0, e_k)) for $i = 1, \ldots, n$.

Each arc entering a node i_v, $v = 1, \ldots, n_i$ has an associated weight w_i. Any path defining a cutting pattern must start at node s_k and end at node e_k and respect the knapsack constraint. We note N the set of nodes, and A_k the set of arcs associated to a pattern k. We define binary variables $x_{ij}^k \forall (i, j) \in A_k$, then the formulation

takes the form:

$$\min \quad \sum_{k\in K}\sum_{i=1}^{n} x^k_{s_k,i_1}$$

$$\text{s.t.} \quad \sum_{k\in K}(\sum_{v=1}^{n_i}\sum_{(j,i_v)\in A_k} x^k_{j,i_v} \quad \geq \quad d_i \quad i=1,\dots n \tag{2.30}$$

$$\sum_{(s_k,i)\in A^k} x^k_{s_k,i} \quad = \quad 1 \quad \forall k \in K \tag{2.31}$$

$$\sum_{(i,j)\in A_k} x^k_{ij} - \sum_{(j,i)\in A_k} x^k_{ji} \quad = \quad 0 \quad \forall i \in N \setminus \{s_k, e_k\}\ \forall k \in K \tag{2.32}$$

$$\sum_{(i,e_k)\in A^k} x^k_{i,e_k} \quad = \quad 1 \quad \forall k \in K \tag{2.33}$$

$$\sum_{i=1}^{n} w_i\,(\sum_{v=1}^{n_i}\sum_{(j,i_v)\in A_k} x^k_{j,i_v}) \quad \leq \quad W\,(\sum_{j=1}^{n} x^k_{s_k,j_1}) \quad \forall k \in K \tag{2.34}$$

$$x^k_{ij} \quad \in \quad I\!N \quad \forall k \in K,\ \forall (i,j) \in A^k$$

The first constraints (2.30) ensure the satisfaction of the demand for each item, (2.31), (2.32) and (2.33) are flow conservation constraints, and constraints (2.34) ensures the feasibility of each pattern.

This formulation allows to define proper patterns, defining $n_i = \min\{\left\lfloor \frac{W}{w_i} \right\rfloor, d_i\}$, and it amounts to consider a binary decomposition of items.

2.5 MASTER FORMULATIONS WITH EXCHANGES BUILT-IN

The above cutting stock formulations can be amended to represent feasible exchanges that can be operated in a cutting pattern. For instance replacing an

item i by item k and l in a cutting pattern is feasible if $w_i \geq w_k + w_l$. Modeling such exchange is not necessary to model the problem since instead of modifying a cutting pattern one can generate a new one, leading to the same result. However, the apparent redundancy in the model can have benefit when solving it. In particular, in a column generation approach, a simple exchange can be applied to any already defined pattern for which it is feasible and therefore it defines implicitly a whole set of cutting patterns that may not yet be generated and need not to be generated. Using such exchange has been shown to accelerate the column generation procedure. In particular, Valerio de Carvalho [37] used this idea to accelerate the resolution of bin packing problems.

In this section we consider three alternatives to formulation [M1] where exchanges are modeled. We assume formulation [ISP1] for the subproblem. First, we see how the simplest exchange (replace i by j for $w_j < w_i$) can be modeled simply be re-writing the covering constraints (2.12) in a different form. Then, we consider how modeling any feasible exchange using extra variables. For these reformulations, we assume that orders are different and sorted in non increasing order of their width:

$$w_1 > w_2 > \ldots > w_n$$

2.5.1 AGGREGATING COVERING CONSTRAINTS

The production of piece of k can be used to cover the demand for an item i such that $k < i$. Thus, it is enough to enforce constraint on the aggregate demands for items $1, \ldots, k$ for all i. Modifying the covering constraints (2.12) in [M1]

accordingly leads to the following master reformulation:

$$Z^M = \min \sum_{q \in Q} \lambda_q \tag{2.35}$$

$$[\text{AgregCovM1}] \qquad \text{s.t.} \quad \sum_{q \in Q} (\sum_{k=1}^{i} x_k^q)\, \lambda_q \;\geq\; \sum_{k=1}^{i} d_k \qquad i = 1, \ldots, n \tag{2.36}$$

$$\sum_{q \in Q} \lambda_q \;\leq\; K \tag{2.37}$$

$$\lambda_q \in I\!N \qquad \forall q \in Q \tag{2.38}$$

The covering constraints are an aggregation of those of model [M1]: for item i, we sum the covering constraints of M1 corresponding to all items of width larger or equal to w_i.

To understand the stabilization effect of the built-in exchanges, let us look at the dual problem of the LP relaxation of AgregCovM1. It takes the form:

$$\max \sum_{i=1}^{n} d_i\, (\sum_{k=i}^{n} \nu_k) - K\, \sigma$$

$$\text{s.t.} \quad \sum_{i=1}^{n} (\sum_{k=i}^{n} \nu_k)\, x_i^q - \sigma \;\leq\; 1 \qquad \forall q \in Q$$

$$\nu_i \geq\; 0 \qquad i = 1, \ldots, n$$

$$\sigma \geq\; 0$$

where ν_i (respectively σ) are the dual variables associated to the constraints (2.36) (respectively (2.37)).

Then, introducing variables $\pi'_i = \sum_{k=i}^{n} \nu_k \ \ \forall i$ we obtain:

$$\max \sum_{i=1}^{n} d_i \, \pi'_i - K \, \sigma$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i^q \, \pi'_i - \sigma \ \leq \ 1 \qquad\qquad \forall q \in Q$$

$$\pi'_i - \pi'_{i+1} \geq \ 0 \qquad\qquad i = 1, \ldots, n-1$$

$$\pi'_n \geq \ 0$$

$$\sigma \geq \ 0$$

This formulation is equivalent to D1 with the addition of $n-1$ dual constraints:

$$\pi'_i \ \geq \ \pi'_{i+1} \quad i = 1, \ldots, n-1$$

that means that the reward associated to the cut of item i should be greater than that of the cut of a smaller item. These constraints belong to a family of dual cuts that were introduced by Valério de Carvalho in [37] to accelerate the column generation procedure.

2.5.2 INTRODUCING EXCHANGE VARIABLES

An alternative formulation of exchanges in the master problem is to introduce exchange variables e_{ik} representing the quantity of item i used to cover the demand for item k. Here, we consider replacing one piece of an item i by one or several copies of a smaller item j. The problem of defining such feasible exchanges can be formulated as a generalized flow model (see [1] Chapter 15). The corresponding graph is presented in Figure 2.3.

The nodes $N = \{1, \ldots, n\}$ are associated to each item i with associated demand d_i, and node 0 is the source node with supply $\sum_i d_i$. Arcs $(0, i), \forall i$ represent a production, the associated flow is $\sum_q x_i^q \lambda_q$. Arcs (i, j) between item i and j such that $w_i > w_j$, i.e. $i < j$ with our indexing, represent feasible exchanges. Indeed, the production of item i is used to cover the demand for item j by further cutting the pieces of item i into pieces for item j. The associated variable e_{ij} represents the flow entering the arc, i.e. the number of copies of i cut for producing j.

The gain on the arc is the multiplier

$$\mu_{ij} = \left\lfloor \frac{w_i}{w_j} \right\rfloor$$

thus the flow leaving the arc, $\mu_{ij}\, e_{ij}$, represents the production of item j obtained from further cutting pieces of i.

In order to avoid redundancy, we do not create the arcs that can be obtained by a relation of transitivity. Thus, beyond the creation of arc $(i, i+1)$, we consider only the arcs corresponding to an incremental increase in μ_{ij}. Moreover, in order to avoid exchanges that obviously lead to *unproper* columns, we further restrict our graph to arcs where $\mu_{ij} \leq d_j$. Hence, the arcs leaving the node associated to item i are

$$A^+(i) = \{(i, j) :\ i < j,\ \mu_{ik} < \mu_{ij} \leq d_j \forall i < k < j\}\ .$$

Similarly, $A^-(i)$ denotes the non redundant exchange arcs entering the node associated with item i.

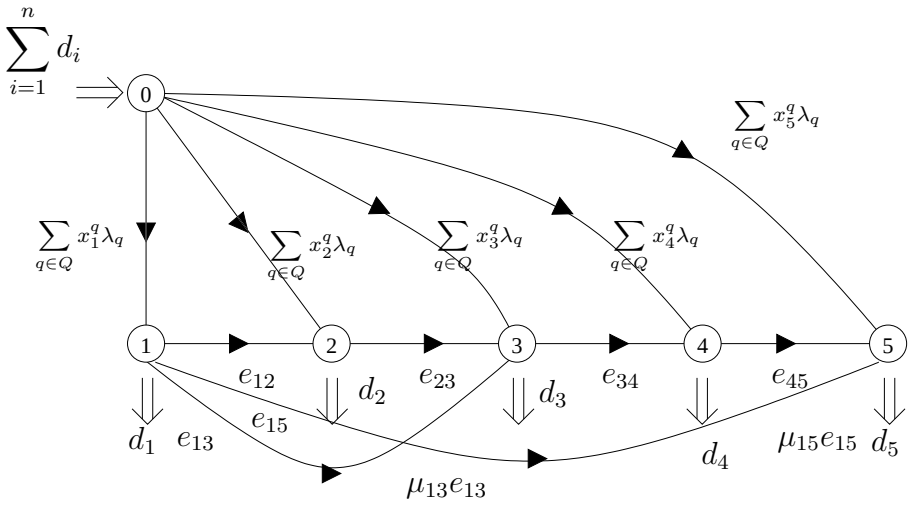


Figure 2.3: Exchange formulation

The formulation with exchange variables e_{ik} is:

$$Z^M = \min \sum_{q\in Q} \lambda_q$$

$[ExchFlowM1]$ s.t.

$$\sum_{q\in Q} x_i^q\,\lambda_q + \sum_{(k,i)\in A^-(i)} \mu_{ki}e_{ki} \;\geq\; d_i + \sum_{(i,j)\in A^+(i)} e_{ij} \qquad \forall i$$

$$\sum_{q\in Q} \lambda_q \;\leq\; K$$

$$\lambda_q \;\in\; I\!N \qquad\qquad \forall q \in Q$$

$$e_{ij} \;\in\; I\!N \qquad\qquad \forall (i,j) \in A.$$

To understand the stabilization effect of the built-in exchange, let us look at the dual problem to the LP relaxation of ExchFlowM1, ignoring the convexity constraint. It is enriched with the dual constraints associated with variables e_{ij}. It

takes the form:

$$
\begin{aligned}
& \max \sum_{i=1}^{n} d_i \, \pi_i & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i^q \, \pi_i \;\leq\; 1 & & \forall q \in Q \\
& \pi_i \geq \left\lfloor \frac{w_i}{w_j} \right\rfloor \pi_j & \forall\, i, (i,j) \in A(i) & \qquad (2.39) \\
& \pi_i \geq 0 & \forall i &
\end{aligned}
$$

Constraints (2.39) say that the reward for cutting i must be at least as large as the reward that could be obtained by transforming i in smaller products j. The coefficient μ_{ij} allows to obtain stronger dual cuts than for the preceding model (when there are larger than 1).

A simplification of the model entails taking all gain factors to be 1, i.e. cut only one copy of j out of a piece of i even if several could be cut. Then, the underlying exchange network is a simple flow problem instead of a generalized flow. Observe that eliminating all arcs that are implied by transitive relations builds down to considering only the simplest exchange arcs $(i, i+1)$ with multipliers $\mu_{ij} = 1$.

Let $DirectExchFlowM1$ denote this simplified model:

$$Z^M = \min \sum_{q \in Q} \lambda_q$$

$[DirectExchFlowM1]$ s.t.

$$\sum_{q \in Q} x_i^q \, \lambda_q + e_{i-1,i} \geq d_i + e_{i,i+1} \quad \forall i \qquad (2.40)$$

$$\sum_{q \in Q} \lambda_q \leq K$$

$$\lambda_q \in I\!N \qquad \forall q \in Q$$

$$e_{ij} \in I\!N \qquad \forall (i,j) \in A.$$

It is equivalent to AgregCovM1 in the sense that it allows the same exchanges. However the stabilization effect may not exactly be the same from a computational point of view due to the presence of the extra variables $e_{i-1,i}$.

2.5.3 USING EXCHANGE VECTORS

Model ExchFlowM1 only captures a small subset of feasible exchanges: cutting an item into a single other product. More general exchanges involve cutting an item into several different products (as done by Valério de Carvalho in [37]) , or even more general, replacing a subset of item pieces by another subset: let r (resp. a) be the indicator vector of the pieces that are replaced (resp. added). This general class of exchange can be modeled by introducing so called exchange vectors in [M1]. These vectors can be seen as the difference between two feasible columns. Exchange vectors can be defined as scaled rays of the subproblem, they are solutions of:

$$Y = \{(a,r) : \sum_i w_i a_i \leq \sum_i w_i r_i, \sum_i w_i r_i \leq W,$$

$$a_i\, r_i = 0\, \forall i,\ a_i \leq d_i\, \forall i,\ a \in I\!N^n, r_i \leq d_i \forall i, r \in I\!N^n\}$$

where bounds d_i on a_i and r_i are enforced to avoid exchanges that would obviously lead to *unproper columns*.

As the set of feasible exchanges is quite large, the idea is to consider them implicitly through a column generation technique. Let E stands for the enumerated set of Y solutions, i.e.

$$Y = \{(a^e, r^e)\}_{e \in E}$$

where the e^{th} exchange vector is characterized by a vector r^e of items that are replaced by a vector a^e of added items [1]. However, when formulating the master problem with the addition of these exchange vectors, one must return to the disaggregated master program [M1dissaggreg] given in ((2.7)-(2.10)) so as to apply exchanges to specific cutting patterns. The augmented formulation takes the form:

$$\min \sum_{k,q \in Q} \lambda_q^k \tag{2.41}$$

[ExchVectM1Disag] s.t. (2.42)

$$\sum_{k,q \in Q} x_i^q\, \lambda_q^k \ + \sum_{k,e \in E} (a_i^e - r_i^e)\, \rho_e^k \geq\ d_i \qquad \forall i \tag{2.43}$$

$$\sum_{e \in E} r_i^e\, \rho_e^k \leq\ \sum_{q \in Q} x_i^q\, \lambda_q^k \qquad \forall i\, \forall k \tag{2.44}$$

$$\sum_{q \in Q} \lambda_q^k\ \leq\ 1 \qquad \forall k \tag{2.45}$$

$$\lambda_q^k \in\ \{0,1\} \qquad \forall q \in Q,\ \forall k \tag{2.46}$$

where constraints (2.44) are needed to insure that one only replaces items that

[1] We shall sometimes use an alternative notation to define an exchange vector letting $x^e \in I\!N^n$ represents an exchange vector e where the negative components in x^e represent r^e while the positive components represent a^e.

were cut.

If we relax constraints (2.44), variables λ_q^k and ρ_e^k can be aggregated into $\lambda_q = \sum_k \lambda_q^k$ and $\rho_e = \sum_k \rho_e^k$ respectively. The relaxed master problem then takes the form:

$$\min \sum_{q \in Q} \lambda_q$$

$[ExchVectM1]$ s.t.

$$\begin{aligned}
\sum_{q \in Q} x_i^q \, \lambda_q + \sum_{e \in E} (a_i^e - r_i^e) \, \rho_e &\geq d_i && \forall i \\
\sum_{q \in Q} \lambda_q &\leq K \\
\lambda_q &\in I\!N && \forall q \in Q \\
\rho_e &\in I\!N && \forall e \in E.
\end{aligned}$$

where ρ_e is the number of times the exchange vector e is used.

The exchange vectors can be generated dynamically by solving a pricing sub-problem:

$$\max\{\sum_{i=1}^{n} \pi_i (a_i - r_i) : (a, r) \in Y\} \tag{2.47}$$

Using exchange vectors in the primal amounts to adding the following dual cuts:

$$\sum_i \pi_i \, (a_i^e - r_i^e) \leq 0 \qquad \forall e \in E \tag{2.48}$$

Hence, well chosen exchange vectors, defining undominated and elementary relations between the dual variables, can help stabilizing the overall column generation approach.

However, relaxing constraints (2.44) leads to a weaker formulation as illustrated by the following example.

Example 4 *Let* $n = 3$, $W = 1$ *and*

$w_1 = \frac{5}{8}$	$d_1 = 4$
$w_2 = \frac{3}{8}$	$d_2 = 2$
$w_3 = \frac{2}{8}$	$d_3 = 3$

Consider feasible cutting patterns

$$x^1 = (1, 1, 0)$$

$$x^2 = (1, 0, 1)$$

$$x^3 = (0, 0, 3)$$

and exchange vector

$$x^4 = (0, -2, 3)$$

represented in figure 2.4.

The optimal solution for the linear relaxation of [ExchVectM1] is 4 *with associated solution* $\lambda_1 = 4,\ \rho_4 = 1$. *While if one solves the LP of [ExchVectM1Disag] to optimality one gets* $4\frac{1}{3}$ *with associated solution* $\lambda_1 = 2,\ \lambda_2 = 2,\ \lambda_3 = \frac{1}{3}$. ∎

The weakness of [ExchVectM1] is due to the fact that the replaced pieces defined in an exchange can be taken from different patterns. It is like if we were allowed to define a cutting pattern for the aggregation of k wide rolls having total width equal to $k\,W$. This, of course, allows to save waste.

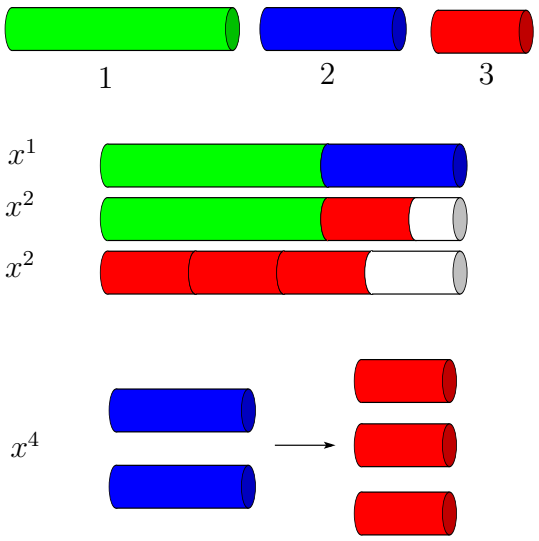


Figure 2.4: Replacing two units of item 2 by three units of item 3 using exchange vector

There is a natural subset of exchange vectors for which the relaxed master [ExchVectM1] is as strong as [ExchVectM1Disag]: When $|r| = 1$, i.e. when only one piece is replaced by a subset of smaller pieces, then the exchange obviously applies to a single cutting pattern. This subclass of exchange vectors is precisely those studied by Valerio de Carvalho [37]. Let

$$E_i = \{e \in E : r_i^e = 1,\ r_j^e = 0\ \forall j \neq i\},\ \ \overline{E}_i = \cup_{j \neq i} E_j \tag{2.49}$$

and let $[RestrExchVectM1]$ be the disaggregate master formulation where the exchange vectors are restricted to those with $|r| = 1$. The disaggregate version

takes the form :

$$\min \sum_{k,q\in Q} \lambda_q^k$$

$[RestrExchVectM1Disag]$ s.t.

$$\sum_{k,q\in Q} x_i^q\,\lambda_q^k - \sum_{k,e\in E_i} \rho_e^k + \sum_{k,e\in\overline{E}_i} a_i^e\,\rho_e^k \;\geq\; d_i \qquad \forall i$$

$$\sum_{e\in E_i} \rho_e^k \;\leq\; \sum_{q\in Q} x_i^q\,\lambda_q^k \quad \forall i,k \qquad (2.50)$$

$$\sum_{q\in Q} \lambda_q^k \;\leq\; 1 \qquad \forall k$$

$$\lambda_q^k \;\in\; I\!N \qquad \forall k,\ q\in Q$$

$$\rho_e^k \;\in\; I\!N \qquad \forall k,\ e\in E.$$

Observation 1 *When exchange vectors are restricted to those with $|r| = 1$, constraints (2.50) are redundant in the above formulation and its LP relaxation.*

Indeed, as the exchange involves transforming a single piece of a given product i, it is enough to ensure that the global production of pieces i is sufficient for satisfying demand plus transformed pieces. The aggregate solution can be disaggregated in any way by arbitrarily assigning k indices.

Thus the associate aggregate master formulation is equivalent. It takes the

form:

$$\min \sum_{q \in Q} \lambda_q$$

$[RestrExchVectM1]$ s.t.

$$\sum_{q \in Q} x_i^q \, \lambda_q - \sum_{e \in E_i} \rho_e + \sum_{e \in \overline{E}_i} a_i^e \, \rho_e \geq d_i \qquad \forall i$$

$$\sum_{q \in Q} \lambda_q \leq K$$

$$\lambda_q \in I\!N \qquad \forall q \in Q$$

$$\rho_e \in I\!N \qquad \forall e \in E.$$

whose dual includes cuts

$$\sum_j \pi_j \, a_j^e \leq \pi_i \qquad \forall e \in E_i \qquad (2.51)$$

2.5.4 EXCHANGES IN THE ARC FLOW FORMULATION

Some exchanges can be represented in the arc flow formulation by recombining path flows in different ways: a cycle in the non oriented graph represents an exchange between several products.

Example 5 *In figure 2.5, where we consider* 3 *items whose widths are* $w_1 = 5$, $w_2 = 3$ *and* $w_3 = 2$, *and* $W = 8$, *the item* 2 *can be exchanged by item* 3 *in the first pattern leading then to the third pattern. Furthermore, considering arcs between two consecutive nodes as waste, this arc flow representation models more than one to one exchanges: for example the cycle* $\{0, 5, 4, 2, 0\}$ *corresponds to the exchange of one copy of item* 1 *by two copies of items* 3 *plus a waste of* 1.

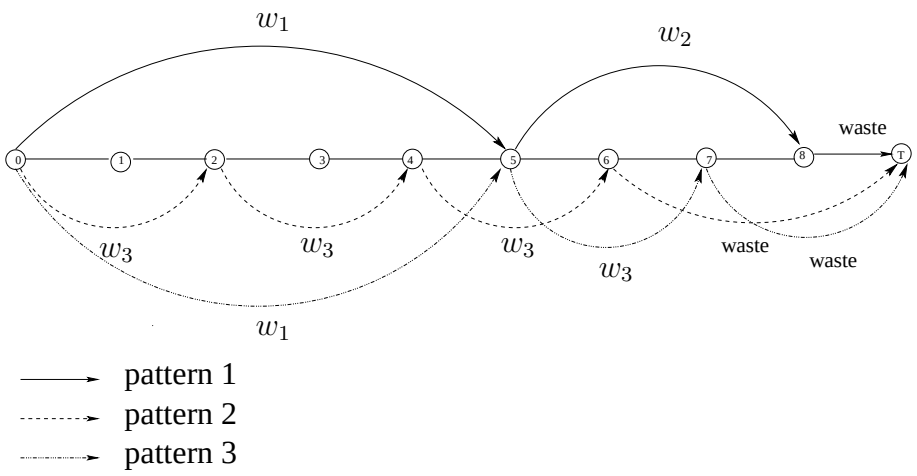


Figure 2.5: Exchanges in the arc flow formulation

2.6 COMPARING THE FORMULATIONS FOR THE STANDARD CUTTING STOCK

Let us now compare the above integer formulations of the standard cutting stock as well as their LP relaxation. Let us use the notation $F \overset{LP}{\rightarrow} F'$ to express that a LP solution to F can be converted into a LP solution to F' with the same cost. Similarly $F \overset{IP}{\rightarrow} F'$ denotes an IP solution transformation that preserves the cost.

The interest of the LP transformation $F \overset{LP}{\rightarrow} F'$ is in the implied relation between resulting dual bounds:

Observation 2 *Assuming we are dealing with minimization problems,*

$$F \overset{LP}{\rightarrow} F' \Rightarrow Z^F_{LP} \geq Z^{F'}_{LP}$$

where Z^F_{LP} is the optimal LP value for F.

The question raised in this section is whether the modification introduced in the master to help stabilizing the column generation solution method does imply

weaker LP bound.

Let us start by showing that they define relaxation. We shall then show some reverse relations that prove that some relaxations do not yield weaker dual bounds.

Proposition 1

$$M1Disag \overset{LP}{\rightarrow} M1 \overset{LP}{\rightarrow} AgregCovM1 \overset{LP}{\rightarrow} DirectExchFlowM1 \overset{LP}{\rightarrow}$$

$$ExchFlowM1 \overset{LP}{\rightarrow} RestrExchVectM1 \overset{LP}{\rightarrow} ExchVectM1$$

proof:

$M1Disag \overset{LP}{\rightarrow} M1$: Let $\lambda \in \{0,1\}^{|Q|\times K}$ be a LP solution to $M1Disag$. Define solution $\lambda' \in I\!N^{|Q|}$ by setting $\lambda'_q = \sum_k \lambda^k_q \ \forall q \in Q$. It is easy to verify that λ' solves M1.

$M1 \overset{LP}{\rightarrow} AgregCovM1$: Let λ be a LP solution to $M1$, then it is solution to AgregCovM1 since the constraints of AgregCovM1 are partial sum of constraints of M1.

$AgregCovM1 \overset{LP}{\rightarrow} DirectExchFlowM1$: Let λ be a LP solution to $AgregCovM1$. For $i = 1,\ldots,n-1$, let $e_{i,i+1} = \sum_q (\sum_{l=1}^{i} x_i^q)\lambda_q - (\sum_{l=1}^{i} d_i)$. Then, (λ, e) solves $DirectExchFlowM1$. Indeed, the constraints of $AgregCovM1$ imply $e_{i,i+1} \geq 0$ while the definition of $e_{i,i+1}$ implies that the constraints of $DirectExchFlowM1$ are satisfied at equality.

$DirectExchFlowM1 \overset{LP}{\rightarrow} ExchFlowM1$: the solution (λ, e) of $DirectExchFlowM1$ augmented with $e_{i,j} = 0$ for $j > i+1$ trivially solves $ExchFlowM1$.

$ExchFlowM1 \overset{LP}{\rightarrow} RestrExchVectM1$: let (λ, e) be a LP solution to $ExchFlowM1$. For each $e_{ij} > 0$ define an exchange vector $(a^e, r^e) \in E_i$

with $r_i^e = 1$ and $a_j^e = \mu_{ij}$ and set the associated variable $\rho_e = e_{ij}$. Then, the solution (λ, ρ) solves $RestrExchVectM1$.

$RestrExchVectM1 \overset{LP}{\rightarrow} ExchVectM1$: the LP solution (λ, ρ) of $RestrExchVectM1$ trivially solves $ExchVectM1$. ∎

Proposition 1 is true whether we work with *proper columns* or not, i.e. we could write $M1(uISP1) \overset{LP}{\rightarrow} AgregCovM1(uISP1)\ldots$ as well as $M1(ISP1) \overset{LP}{\rightarrow} AgregCovM1(ISP1)\ldots$. However, the reverse relations cannot necessarily be established for both bounded and unbounded subproblem. Valerio de Carvalho [37] has proved:

Proposition 2

$$RestrExchVectM1(SP) \overset{LP}{\rightarrow} M1(uSP)$$

In [37] Valerio de Carvalho gives a constructive procedure to transform a LP solution to RestrExchVectM1 into a LP solution to M1, but in the process one might generate *unproper* cutting patterns. He proved the same result in the dual space, showing that the dual solution π to $M1$ is feasible to the dual of $RestrExchVectM1$, i.e. it satisfies all constraints (2.51). Indeed, if one of them is violated then one can construct a cutting pattern (not-necessarily *proper*) with strictly negative reduced cost.

Corollary 1

$$M1(SP) \overset{LP}{\rightarrow} M1(uSP)$$

$$AgregCovM1 \overset{LP}{\rightarrow} M1(uSP)$$

$$DirectExchFlowM1 \overset{LP}{\rightarrow} M1(uSP)$$

$$ExchFlowM1 \overset{LP}{\rightarrow} M1(uSP)$$

But, this is not true for $ExchVectM1$:

Observation 3

$$ExchVectM1 \overset{LP}{\not\rightarrow} M1(uSP)$$

Indeed, as shown by Example 4, cutting waste can be saved by implicitly pasting wide rolls together.

For $AgregCovM1$ and $DirectExchFlowM1$ there is a stronger result not implied by Proposition 2 : the use of direct exchange $e_{i,i+1}$ induces a master formulation equivalent to the standard master with *proper columns*.

Proposition 3

$$DirectExchFlowM1(SP) \overset{LP}{\rightarrow} M1(SP)$$

proof: Let (λ, e) be an optimum LP solution to $DirectExchFlowM1$ with constraints (2.40) set to equality [2]. We proceed to show that the exchanges can be implemented by a way of transforming existing patterns without introducing *unproper* patterns, i.e. after transformation, all cutting patterns q that are used still have the property that $x_i^q \leq d_i \ \ \forall i$.

Let i be the largest item index for which $e_{i-1,i} > 0$ (remember that items are indexed in order that $w_1 \geq w_2 \geq \ldots \geq w_n$). Let $k < i$ be the largest item index before i for which $e_{k-1,k} = 0$. Then, any pieces of j for $j = k, \ldots, i-1$ can be transformed in a piece of i in a partial implementation of exchange $e_{i-1,i}$. Let $S = \{q \in Q : \lambda_q > 0, \ x_{k,i-1}^q = \sum_{j=k}^{i-1} x_j^q > 0\}$ be the set of patterns that are possible candidates for producing extra pieces of i through further cutting larger pieces.

[2] Both problems, with equality (partitioning) or inequality (covering) constraints, admit the same optimal solution.

We need to show that set S contains candidates q with $x_i^q < d_i$ so that after increasing x_i^q the cutting pattern remains *proper*. We show this by contradiction. Assume $x_i^q = d_i \ \forall q \in S$. The master constraint associated with item i in $DirectExchFlowM1$ can be re-written as

$$\sum_{q \in S} x_i^q \, \lambda_q + \sum_{q \in Q \setminus S} x_i^q \, \lambda_q + e_{i-1,i} = d_i$$

because $e_{i,i+1} = 0$. As $x_i^q = d_i \ \forall q \in S$ and $\sum_{q \in Q \setminus S} x_i^q \, \lambda_q + e_{i-1,i} > 0$, this implies that

$$\sum_{q \in S} d_i \, \lambda_q < d_i$$

which in turn implies that

$$\sum_{q \in S} \lambda_q < 1$$

But this is a contradiction since $e_{k-1,k} = 0$ implies that

$$\sum_{q \in S} x_{k,i-1}^q \lambda_q \geq d_{k,i-1}$$

where $d_{k,i-1} = \sum_{j=k}^{i-1} d_j$; and because columns are *proper*

$$\sum_{q \in S} d_{k,i-1} \lambda_q \geq \sum_{q \in S} x_{k,i-1}^q \lambda_q \ ;$$

thus, $\sum_{q \in S} \lambda_q \geq 1$. Thus, we have established the existence of candidate columns $q \in S$ to which a piece of $j \in \{k, \ldots, i-1\}$ can be transformed in a piece of i while keeping *proper* patterns. In practice, one would choose the largest index j such as $\exists q \in S$ with $x_j^q > 0$ and $x_i^q < d_i$. Then one defines a pattern q' as follows: initially set $x^{q'} = x^q$, then redefine $x_j^{q'} = x_j^q - 1$ and $x_i^{q'} = x_i^q + 1$. The associated variable value is set as $\lambda_{q'} = \min\{\lambda_q, e_{i-1,i}\}$, while λ_q is redefined as $\lambda_q := \lambda_q - e_{i-1,i}$ and $e_{i-1,i}$ is redefined as $e_{i-1,i} := e_{i-1,i} - \lambda_q$. The process is reiterated until all exchange variables are brought to zero. ∎

Thus, formulation $DirectExchFlowM1$ does not yield any relaxation compared to $M1$. As $DirectExchFlowM1$ is a relaxation of $AgregCovM1$, the same result holds for $AgregCovM1$.

Corollary 2

$$AgregCovM1(SP) \overset{LP}{\rightarrow} M1(SP)$$

However, once we allow exchanges with multiplicity larger than one, the result is lost:

Observation 4

$$ExchFlowM1(SP) \overset{LP}{\not\rightarrow} M1(SP) \ .$$

As shown by the following example, $ExchFlowM1(SP)$ can be a strict relaxation of $M1(SP)$ because exchanges implicitly amount to using *unproper columns*.

Example 6 *Let* $n = 2, W = 1$ *and*

$w_1 = \frac{2}{3}$	$d_1 = 1$
$w_2 = \frac{1}{3}$	$d_2 = 2$

Consider feasible cutting patterns

$$x^1 = (1, 1)$$

$$x^2 = (0, 2)$$

and exchange $1 \rightarrow 2$ *with multiplicity* $\mu_{1,2} = 2$*, as represented in the following figure 2.6.*

The optimal solution for the linear relaxation of [M1(SP)] is $1\frac{1}{2}$ *with associated solution* $\lambda_1 = 1,\ \lambda_2 = \frac{1}{2}$*. While the optimal LP solution for [ExchFlowM1] is* $1\frac{1}{3}$ *with associated solution* $\lambda_1 = 1\frac{1}{3},\ e_{1,2} = \frac{1}{3}$*.* ∎

Observation 5

$$ExchVectM1 \overset{LP}{\rightarrow} F_1 \ .$$

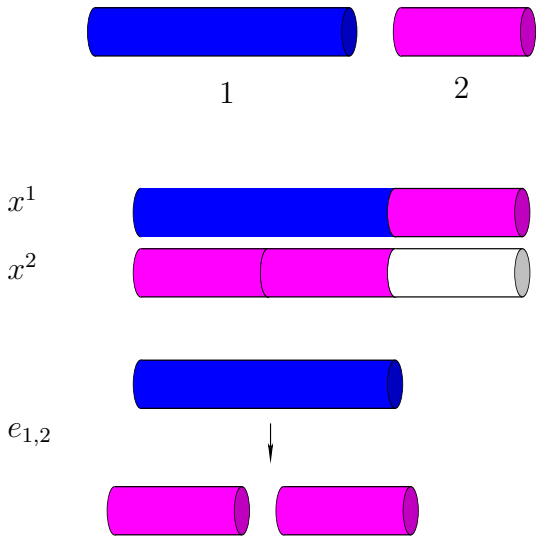


Figure 2.6: Replacing one unit of item 1 by two units of item 2 in cutting pattern 1

Indeed, one can can show that for all LP solutions λ to $ExchVectM1(SP)$ there exits a LP solution x to $F1$ of same cost.

The above results are summarized in Table 2.1 with

$$Z_{LP}^{F1} \leq Z_{LP}^{ExchVectM1} \leq uLD \leq LD \leq Z_{IP}$$

Finally, let us observe that all the above LP transformations do have their equivalent counter part in IP term. Indeed the above transformations preserve integrality of the solution. But, one can say more in terms of IP equivalence.

Proposition 4

$$RestrExchVectM1 \overset{IP}{\rightarrow} M1$$

proof: Let (λ, ρ) be an optimum IP solution to $RestrExchVectM1$ where the master constraints are set as equality constraints. Let us show how it can be transformed in an IP solution λ' to M1 with the same cost. If $\rho = 0$, the result is trivial. Assume therefore $\exists e \in E : \rho_e \geq 1$. From among all those, select

$e \in E_i$ for some i such that $\sum_{e \in \overline{E}_i} a_i^e \ \rho_e = 0$. There must exist such i for otherwise there would be a cyclic series of exchange (which is a contradiction with our assumption that $w_1 > w_2 > \ldots > w_n$). Thus, the master constraints imply $\sum_{q \in Q} x_i^q \lambda_q \geq d_i + 1$. Then, take any pattern q with $\lambda_q \geq 1$ and $x_i^q \geq 1$, define a new pattern q' from q by setting $x_i^{q'} = x_i^q - 1$ and $x_j^{q'} = x_j^q + a_j^e \forall j$ and take it $\lambda_{q'} = \min\{\lambda_q, \rho_e\}$ times. Then reset $\lambda_q := \lambda_q - \lambda_{q'}$, $\rho_e = \rho_e - \lambda_{q'}$ and reiterate until all variables ρ_e are bring to zero. In the process we will not introduced *unproper* pattern because the master constraints set as equality constraint guarantee that columns entries plus exchange addition remain below or equal to item demand. ∎

As a result all stronger master formulations are also IP equivalent to $M1$. But, formulation $ExchVectM1$ is not IP equivalent to $M1$

LP bound	Formulation
Z_{LP}^{F1}	$F1(ISP1)$, $F1(BSP1)$
$Z_{LP}^{ExchVectM1}$	ExchVectM1(SP)
uLD	R1, M1(uISP1), M1(uBSP1), M1(uFSP1), ExchFlowM1(SP), RestrExchVectM1(SP)
LD	M1(ISP1), M1(BSP1), AgregCovM1(SP), DirectExchFlowM1(SP)

Table 2.1: LP Bounds

Observation 6

$$ExchVectM1 \overset{IP}{\not\to} M1(SP)\ .$$

As shown by Example 4, where the IP solution to $ExchVectM1$ is 4 while the IP solution to $M1$ is at least 5.

2.7 REFORMULATIONS OF THE VARIANT WITH INTERVALS ON PRODUCTION

When the production is free to take value within an interval, the waste must be set explicitly as the objective. Hence, the objective value is no longer integer and its LP relaxation cannot be rounded up. The packing constraints (enforcing an upper bound on production) induce a new set of dual variables. The sub-system remains the same than for the standard cutting stock problem. The explicit reformulation resulting from the sub-system formulation [ISP1] is the compact formulation presented in Section 1 [F2], denoted F2(ISP1), while [BSP1] gives rise to the following binary formulation:

$$Z = \min \sum_{k=1}^{K}(y_k\, W - \sum_{i=1}^{n}\sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ijk})$$

$$\text{[F2(BSP1)]} \quad \text{s.t.} \quad \sum_{k=1}^{K}\sum_{j=1}^{n_i} m_{ij}\, x_{ijk} \geq \underline{d_i} \quad \forall i$$

$$\sum_{k=1}^{K}\sum_{j=1}^{n_i} m_{ij}\, x_{ijk} \leq \overline{d_i} \quad \forall i$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ijk} \leq W\, y_k \quad \forall k$$

$$\sum_{j=1}^{n_i} m_{ij}\, x_{ij} \leq u_i \quad \forall i$$

$$y_k \in \{0,1\} \quad \forall k$$

$$x_{ijk} \in \{0,1\} \quad \forall i,\, \forall j,\, \forall k$$

An explicit arc-flow formulation, such as [R1], can be obtained from [uFSP1]. The objective being the minimization of the waste, the waste arcs from set $\{(u, W+1) : u = 0, \ldots, W\}$ weighted by the associated waste incurred define

the objective. The resulting formulation [R2] is

$$\min \quad \sum_{u \in N} (W - u)\, x_{u,W+1}$$

$$[R2] \quad \text{s.t.} \quad \sum_{(u,v)\in A} x_{uv} - \sum_{(v,w)\in A} x_{vw} = 0 \quad \forall v \in N \setminus \{0, W+1\}$$

$$\sum_{v \in N} x_{0v} = \sum_{v \in N} x_{vW+1}$$

$$\sum_{(u,v)\in A(i)} x_{uv} \geq \underline{d_i} \quad \forall i$$

$$\sum_{(u,v)\in A(i)} x_{uv} \leq \overline{d_i} \quad \forall i$$

$$x_{uv} \in I\!N \quad \forall (u,v) \in A$$

The implicit reformulations for the standard cutting stock problem can be adapted for this variant. When the sub-system is defined by [ISP1] or [uISP1], the reformulation gives rise to the following master problem:

$$Z^{M2} = \min \sum_{q \in Q} (W - \sum_{i} (w_i\, x_i^q))\, \lambda_q \tag{2.52}$$

$$[\text{M2}] \quad \text{s.t.} \quad \sum_{q \in Q} x_i^q\, \lambda_q \geq \underline{d_i} \qquad i = 1, \ldots, n \tag{2.53}$$

$$\sum_{q \in Q} x_i^q\, \lambda_q \leq \overline{d_i} \qquad i = 1, \ldots, n \tag{2.54}$$

$$\sum_{q \in Q} \lambda_q \leq K \tag{2.55}$$

$$\lambda_q \in I\!N \qquad \forall q \in Q \tag{2.56}$$

When the subproblem is defined by [BSP1] or [uBSP1] (resp. [uFSP1]), we must replace x_i^q by $\sum_j m_{ij} x_{ij}^q$, (resp. $\sum_{uv:v=u+w_i} x_{uv}^q$).

Let π_i and ν_i be the dual variables associated respectively to the covering (2.53) and packing (2.54) master constraints. Since the objective is to minimize the waste, the reduced cost, $\overline{c}_q$, of a cutting pattern $q \in Q$ is:

$$\overline{c}_q = W - \sum_{i=1}^{n} (w_i + \pi_i - \nu_i)\, x_i^q.$$

Using [ISP1], the resulting pricing subproblem is:

$$\xi(\pi,\nu) = \max\{\sum_{i=1}^{n}(w_i+\pi_i-\nu_i)x_i : \sum_{i=1}^{n} w_i x_i \leq W,\ x_i \leq\ u_i \text{ and } x_i \in\ I\!N \text{ for } i = 1,\ldots n\}$$

The Lagrangian relaxation of packing and covering constraint gives

$$L(\pi,\nu) = \min \sum_{q\in Q}(W - \sum_{i}(w_i + \pi_i - \nu_i)\, x_i^q)\, \lambda_q + \sum_{i=1}^{n} \pi_i\, \underline{d_i} - \sum_{i=1}^{n} \nu_i\, \overline{d_i}$$

$$\sum_{q\in Q} \lambda_q \ \leq\ K$$

$$\lambda_q \in\ I\!N \qquad\qquad \forall q \in Q$$

that leads to Lagrangian bound:

$$L(\pi,\nu) = \sum_{i=1}^{n}(\pi_i\, \underline{d_i} - \nu_i\, \overline{d_i}) + \min\{K\ (W - \xi(\pi,\nu)), 0\}$$

As the objective value is not necessarily integer, these Lagrangian dual bounds cannot be rounded up as for the standard cutting stock problem.

The dual of the master problem is:

$$\max \sum_{i=1}^{n} \underline{d_i}\, \pi_i - \overline{d_i}\, \nu_i - K\, \sigma \qquad (2.57)$$

[D2] s.t. $$\sum_{i=1}^{n} x_i^q\, (w_i + \pi_i - \nu_i) - \sigma \ \leq\ W \qquad \forall q \in Q \qquad (2.58)$$

$$\pi_i,\ \nu_i \ \geq\ 0 \qquad i = 1,\ldots,n$$

$$\sigma \ \geq\ 0$$

The master formulations with exchanges can be adapted, but we have to take into account the waste induced by the replacement of an item by another of larger width in the objective. So the aggregation of the covering and packing constraints cannot be used here, but the master problem can be formulated with the use of exchange variables e_{ik}. Their cost is the waste induced by the exchange of one item by another. The master formulation with exchange flow variables becomes

$$Z^M = \min \sum_{q \in Q} (W - \sum_i (w_i \, x_i^q)) \, \lambda_q + \sum_{(i,j) \in A^+(i)} (w_i - \mu_{ij} \, w_j) \, e_{ij}$$

$[ExchFlowM2]$ s.t.

$$\begin{aligned}
\sum_{q \in Q} x_i^q \, \lambda_q + \sum_{(k,i) \in A^-(i)} \mu_{ki} e_{ki} &\geq \underline{d_i} + \sum_{(i,j) \in A^+(i)} e_{ij} \qquad \forall i \\
\sum_{q \in Q} x_i^q \, \lambda_q + \sum_{(k,i) \in A^-(i)} \mu_{ki} e_{ki} &\leq \overline{d_i} + \sum_{(i,j) \in A^+(i)} e_{ij} \qquad \forall i \\
\sum_{q \in Q} \lambda_q &\leq K \\
\lambda_q &\in I\!N \qquad \forall q \in Q \\
e_{ij} &\in I\!N \qquad \forall (i,j) \in A.
\end{aligned}$$

The dual constraints associated to variables e_{ij} now take the form:

$$(w_i + \pi_i - \nu_i) \geq \left\lfloor \frac{w_i}{w_j} \right\rfloor (w_j + \pi_j - \nu_j) \qquad \forall \, i, (i,j) \in A(i)$$

which means that the value of item i (its coefficient in the column generation subproblem) must be greater or equal to the value that could be obtained by converting item i into μ_{ij} pieces of j.

We can also use a formulation with exchange vectors but the latter now involves a cost. The aggregate master takes the form:

$$\min \sum_{q \in Q} (W - \sum_{i} (w_i \, x_i^q)) \, \lambda_q + \sum_{e \in E} \sum_{i} w_i \, (r_i^e - a_i^e) \, \rho_e \qquad (2.59)$$

[ExchVectM2Disag] s.t.

$$\sum_{q \in Q} x_i^q \, \lambda_q \; + \sum_{e \in E} (a_i^e - r_i^e) \, \rho_e \geq \underline{d_i} \qquad i = 1, \ldots, n \qquad (2.60)$$

$$\sum_{q \in Q} x_i^q \, \lambda_q \; + \sum_{e \in E} (a_i^e - r_i^e) \, \rho_e \leq \overline{d_i} \qquad i = 1, \ldots, n \qquad (2.61)$$

$$\sum_{q \in Q} \lambda_q \; \leq \; K$$

$$\lambda_q \in \mathbb{N} \qquad \forall q \in Q$$

$$\rho_e \in \mathbb{N} \qquad \forall e \in E$$

and it amounts to add in the dual problem the constraints:

$$\sum_{i} (w_i + \pi_i - \nu_i) \, (a_i^e - r_i^e) \;\; \leq \;\; 0 \qquad \forall e \in E$$

The relation between these different formulations remains the same as for the standard cutting stock problem.

2.8 REFORMULATIONS OF THE MULTIPLE WIDTHS CUTTING STOCK PROBLEM

We consider here several wide roll widths, so we define a sub-system X_l for each wide roll type l, $l = 1, ...L$, where L is the number of different widths. A formu-

lation for the subproblem corresponding to the wide roll type l is:

$$\max_{x \in X} \sum_{i=1}^{n} p_i \, x_i$$

$$\begin{aligned} \text{[ISP1l]} \quad \text{where } X = \{ \sum_{i=1}^{n} w_i \, x_i &\leq W_l \\ x_i &\leq u_i \quad i = 1, \ldots n \\ x_i &\in I\!N \quad i = 1, \ldots n \} \end{aligned}$$

Sub-system formulations [BSP1], [FSP1] and [ESP1] are adapted in the same way, and are denoted as [BSP1l], [FSP1l] and [ESP1l].

The arc-flow formulation [R1] becomes:

$$\min \quad \sum_{l=1}^{L} z_l$$

$$[R3] \quad \text{s.t.} \quad \sum_{(u,v) \in A} x_{uv} - \sum_{(v,w) \in A} x_{vw} = \begin{cases} \sum_{l=1}^{L} z_l & \text{if} \quad v = 0 \\ -z_l & \text{for} \quad v = W_l, \forall l \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \sum_{(u,v) \in A(i)} x_{uv} &\geq d_i \quad \forall i \\ z_l &\leq K_l \quad \forall l \\ x_{uv} &\in I\!N \quad \forall (u,v) \in A \\ z_l &\in I\!N \quad \forall l \end{aligned}$$

where z_l corresponds to the number of wide rolls of width W_l used.

The master formulation resulting from the Dantzig-Wolfe decomposition is the same as the disaggregated formulation [M1Disag] for the standard cutting stock problem. Patterns of same width can be aggregated so as to eliminate the symmetry. We assume that wide roll are sorted in non increasing order of their width: $W_1 \leq W_2 \leq \ldots \leq W_L$, then an aggregated master formulation is:

$$Z^{M3} = \min \sum_{q \in Q} \lambda_q$$

$$\text{[M3]} \qquad \text{s.t.} \quad \sum_{q \in Q} x_i^q \, \lambda_q \;\geq\; d_i \qquad\qquad i = 1, \ldots, n$$

$$\sum_{k=1}^{l} \sum_{q \in Q} \delta(W^q \leq W_k) \, \lambda_q \;\leq\; \sum_{k=1}^{l} K_k \qquad\qquad l = 1, \ldots, L$$

$$\lambda_q \in I\!N \qquad\qquad \forall q \in Q$$

where K_l is the number of stock wide rolls of type l.

The dual of the master formulation [M3] is:

$$\max \sum_{i=1}^{n} d_i \, \pi_i - \sum_{l=1}^{L} \sum_{k=1}^{l} K_k \, \sigma_l$$

$$\text{[D3]} \quad \text{s.t.} \sum_{i=1}^{n} x_i^q \, \pi_i - \sum_{l=1}^{L} \sum_{k=1}^{l} \delta(W^q \leq W_k) \, \sigma_l \;\leq\; 1 \qquad\qquad \forall q \in Q$$

$$\pi_i \;\geq\; 0 \qquad\qquad i = 1, \ldots, n$$

$$\sigma_l \;\geq\; 0 \qquad\qquad l = 1, \ldots, L$$

And the Lagrangian's bound obtained by relaxing the covering constraints is:

$$L(\pi) = \min \sum_{q \in Q} (1 - \sum_{i=1}^{n} \pi_i \, x_i^q) \, \lambda_q + \sum_{i=1}^{n} \pi_i \, d_i$$

$$\sum_{k=1}^{l} \sum_{q \in Q} \delta(W^q \leq W_k) \, \lambda_q \; \leq \; \sum_{k=1}^{l} K_k \qquad\qquad l = 1, \ldots, L$$

$$\lambda_q \in \; I\!N \qquad\qquad \forall q \in Q$$

whose solution is

$$L(\pi) = \sum_{i=1}^{n} \pi_i \, d_i + \sum_{l=1}^{L} \; K_l \min\{(1 - \xi^l(\pi)), 0\}$$

Exchanges can be modeled on the same way that for the standard cutting stock problem.

2.9 REFORMULATIONS OF THE VARIANT WITH TECHNICAL RESTRICTIONS

The side constraints resulting from the technical restrictions are set in the subsystem X to define valid cutting patterns. Formulation [ISP1] adding the constraints on the minimum width to be cut and on the maximum cardinality of a cut

set becomes:

$$\max_{x \in X} \sum_{i=1}^{n} p_i \, x_i$$

$$\text{[ISP4]} \quad \text{where } X = \{W_{min} \leq \sum_{i=1}^{n} w_i \, x_i \leq W_{max}$$

$$\sum_{i=1}^{n} x_i \leq C$$

$$x_i \leq u_i \quad i = 1, \ldots n$$

$$x_i \in I\!N \quad i = 1, \ldots n\}$$

The multiple class binary transformation is yet valid, it leads to the following subproblem formulation:

$$\max_{x \in X} \sum_{i=1}^{n} p_{ij} \, x_{ij}$$

$$\text{[BSP4]} \quad \text{where } X = \{W_{min} \leq \sum_{i=1}^{n} \sum_{j=1}^{n_i} w_i \, m_{ij} \, x_{ij} \leq W_{max}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n_i} m_{ij} \, x_{ij} \leq C$$

$$\sum_{j=1}^{n_i} m_{ij} \, x_{ij} \leq u_i \quad i = 1, \ldots n$$

$$x_{ij} \in \{0,1\} \quad i = 1, \ldots n, \; j = 1, \ldots n_i\}$$

The subproblem can be solved by dynamic programming. Hence, the third formulation [uFSP1] can be adapted so as to model the longest path problem underlying the dynamic program. We define the arcs set: $A = \cup_i A(i) \cup \{(u, W+1) : u = W_{min}, \ldots, W_{max}\}$, to specify that the arcs corresponding to the waste should have their head on a node after W_{min}. The cardinality constraint must be added

explicitly to the formulation giving rise to a two resources constraints longest path problem.

The sub-system [ISP4] leads to the compact formulation [F4], and [BSP4] to the binary compact formulation [F3(BSP3)] with the following additional constraints:

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ijk} \quad \geq \quad W_{min}\, y_k \quad \forall k$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i} m_{ij}\, x_{ijk} \quad \leq \quad C \quad \forall k$$

The implicit reformulations remain the same as for the variant with interval on production since the technical constraints are set in the sub-system. The Lagrangian bounds are then also equivalent to those given previously.

The formulation with built-in exchanges do not extend naturally to the present variant with additional technical constraints in the subproblem. Even if we relax the minimum cut width constraint that can mostly be interpreted as a “business rule”, that remains to enforce the cardinality constraint. One to one exchange do satisfy the latter. But the more general exchanges of formulation [ExchFlowM1] or [ExchVectM1] do not extend to this variant.

2.10 REFORMULATION OF THE MINIMIZATION OF SETUPS

Let x_0 be the multiplicity of cutting pattern q, a formulation for the subproblem is:

$$\max_{x \in X} \sum_{i=1}^{n} p_i \, x_i \, x_0$$

$$\text{[ISP5]} \quad \text{where } X = \{ \sum_{i=1}^{n} w_i \, x_i \leq W$$

$$x_0 \, x_i \leq u_i \quad i = 1, \ldots n$$

$$x_i \in I\!N \quad i = 1, \ldots n$$

$$x_0 \in I\!N \}$$

The sub-system X leads to the compact reformulation [F5].

This subproblem can be reformulated as a bilinear binary knapsack problem. The x_i decomposition is the same as for [BSP1], while for x_0 we apply the change of variables:

$$x_0 = \sum_{l} m_l \, x_{0l}$$

with $x_{0l} \in \{0,1\}$, $m_l = 2^{l-1} \ \forall l = 1,\ldots,\lfloor \log_2({x_0}^{max}) \rfloor + 1$ and ${x_0}^{max} = \min\{\max_i d_i, K\}$. The binary reformulation takes the form:

$$\max_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{l} p_{ijl}\, x_{0l}\, x_{ij}$$

[BSP5] where $X = \{ \sum_{i=1}^{n} \sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ij} \leq W$

$$\sum_{j=1}^{n_i} \sum_{l} m_l\, m_{ij}\, x_{ij}\, x_{0l} \leq u_i \quad i = 1,\ldots n$$

$$x_{ij} \in \{0,1\} \quad i = 1,\ldots n \; j = 1,\ldots n_i$$

$$x_{0l} \in \{0,1\} \quad \forall l\}$$

As for [BSP1], this reformulation does not allow to improve the LP dual bound, but the interest is twofold: in our definition of branching schemes and to obtain a linearized formulation. Indeed the non-linearities can be eliminated introducing binary variables $z_{ijl} = x_{ij}\, x_{0l}$. Let $m_{ijl} = m_l\, m_{ij}$, a formulation for the Linearized

Subproblem is:

$$\max_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{l} p_{ijl}\, z_{ijl}$$

[LSP5] where $X = \{$

$$\sum_{i=1}^{n} \sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ij} \leq W$$

$$\sum_{j=1}^{n_i} \sum_{l} m_{ijl}\, z_{ijl} \leq \overline{u}_i \qquad \forall i$$

$$z_{ijl} \leq x_{ij} \qquad \forall i; \forall j; \forall l$$

$$x_{ij} \leq \sum_{l} z_{ijl} \qquad \forall i; \forall j$$

$$z_{ijl} \leq x_{0l} \qquad \forall i\ \forall j\ \forall l$$

$$x_{ij} + x_{0l} - 1 \leq z_{ijl} \qquad \forall i\ \forall j\ \forall l$$

$$x_{ij} \in \{0,1\} \qquad \forall i\ \forall j$$

$$y_i \in \{0,1\} \qquad \forall i$$

$$x_{0l} \in \{0,1\} \qquad \forall l$$

$$z_{ijl} \in \{0,1\} \qquad \forall i\ \forall j\ \forall l\}$$

This sub-system formulation leads to a linearized binary compact formulation

[F5']. For the variant with tolerance on production it takes the form:

$$Z = \min \sum_{k=1}^{K} y_k$$

$$
\begin{aligned}
[\text{F5'}] \qquad \text{s.t.} \quad \sum_{k=1}^{K} \sum_{i} \sum_{j} \sum_{l} m_l \, m_{ij} \, z_{ijlk} &\geq \underline{d_i} \qquad i = 1, \ldots, n \\
\sum_{k=1}^{K} \sum_{i} \sum_{j} \sum_{l} m_l \, m_{ij} \, z_{ijlk} &\leq \overline{d_i} \qquad i = 1, \ldots, n \\
\sum_{l} m_l \, x_{0lk} \, W - \sum_{i} \sum_{j} \sum_{l} m_l \, m_{ij} w_i \, z_{ijlk} &\leq R \\
\sum_{i=1}^{n} \sum_{j=1}^{n_i} m_{ij} \, w_i \, x_{ijk} &\leq W \, y_k \qquad k = 1, \ldots, K \\
\sum_{j} \sum_{l} m_l \, m_{ij} \, z_{ijlk} &\leq d_i \qquad i = 1, \ldots, n \\
z_{ijlk} &\leq x_{ijk} \qquad \forall i; \forall j; \forall l \\
x_{ijk} &\leq \sum_{l} z_{ijlk} \qquad \forall i; \forall j \\
z_{ijlk} &\leq x_{0lk} \qquad \forall i \; \forall j \; \forall l \\
x_{ijk} + x_{0lk} - 1 &\leq z_{ijlk} \qquad \forall i \; \forall j \; \forall l \\
x_{ijk} &\in \{0, 1\} \qquad \forall i, \forall j, \forall k \\
y_k &\in \{0, 1\} \qquad k = 1, \ldots, K \\
z_{ijlk} &\in \{0, 1\} \qquad \forall i, \; \forall j, \; \forall l \, \forall k \\
x_{0lk} &\in \{0, 1\} \qquad \forall l \, \forall k
\end{aligned}
$$

The Dantzig-Wolfe reformulation leads to the following master problem:

$$\min \sum_{q \in Q} \lambda_q$$

$$\text{[M5]} \qquad \text{s.t.} \quad \sum_{q \in Q} x_0^q \, x_i^q \, \lambda_q \;\geq\; \underline{d_i} \qquad i = 1, \ldots, n \qquad (2.62)$$

$$\sum_{q \in Q} x_0^q \, x_i^q \, \lambda_q \;\leq\; \overline{d_i} \qquad i = 1, \ldots, n$$

$$\sum_{q \in Q} (W - \sum_i w_i \, x_i^q) \, x_0^q \, \lambda_q \leq \; R$$

$$\sum_{q \in Q} \lambda_q \leq \; K$$

$$\lambda_q \in \; \{0, 1\} \qquad \forall q \in Q$$

where $x_0^q \in \{1, \ldots, \min_i \{\lfloor \frac{\overline{d_i}}{x_i^q} \rfloor\}$.

Using [ISP5] the reduced cost, $\overline{c}_q$, of pattern q is then

$$\overline{c}_q = 1 - \sum_i (\pi_i - \nu_i) \, x_0^q \, x_i^q + \sigma \, (W - \sum_i w_i \, x_i^q) \, x_0^q$$

and the Lagrangian bound obtained by the relaxation of the covering constraints is:

$$L(\pi) = \min \sum_{q \in Q} (1 - \sum_{i=1}^{n} (\sigma \, w_i + \pi_i - \nu_i) \, x_i^q \, x_0^q + \sigma \, W \, x_0^q) \, \lambda_q + \sum_{i=1}^{n} \pi_i \, \underline{d_i} - \sum_{i=1}^{n} \nu_i \, \overline{d_i} - \sigma \, R$$

$$\sum_{q \in Q} \lambda_q \;\leq\; K$$

$$\lambda_q \in \; \{0, 1\} \quad \forall q \in Q$$

Its solution is:

$$L(\pi) = \sum_{i=1}^{n} \pi_i \, d_i + \min\{K \, (1 - \xi(\pi)), 0\}$$

Exchanges can be modeled as for the standard cutting stock problem.

3

KNAPSACK SUB-PROBLEMS

The integer subproblem formulation [ISP] can be solved using standard solvers, as the classic depth-first-search branch-and-bound algorithm of Horowith and Sahni (see [28]) or a dynamic programming recursion. For our numerical tests, we used a specialized branch-and-bound algorithm for the binary multiple class knapsack problem, operating a binary decomposition of the integer variables x_i. The algorithm is adapted form that of Horowith and Sahni for the standard binary knapsack problem (it takes into account the class capacities). Furthermore, as the binary components of a class have the same ratio profit per weight, the LP bound is computed with a greedy algorithm for the bounded integer knapsack problem.

However, during the branch-and-price procedure, the introduction of branching constraints in the master implies some modifications to the knapsack subproblems, then standard solvers can not be used anymore. The branching rules that we use were presented in section 2.3.7. Branching rule 1 based on column subset

$\hat{Q} = \{q \in Q : x_i^q > 0\}$ requires the introduction of new binary variables in the subproblem, the setup variable $y_i = 1$ if $x_i > 0$ with an associated set-up cost in the objective. Branching rule 2 based on column subset $\hat{Q} = \{q \in Q : x_{ij}^q = 1\}$ requires working with the binary form of the knapsack problem. Then, the subproblem takes the form of a variant of the knapsack problem, the multiple class binary knapsack problem with setup costs:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n_i} p_{ij}\, x_{ij} - \sum_{i=1}^{n} f_i y_i$$

$$\text{[BSUSP1]} \quad \sum_{i=1}^{n} \sum_{j=1}^{n_i} w_i\, m_{ij}\, x_{ij} \leq W$$

$$\sum_{j=1}^{n_i} m_{ij}\, x_{ij} \leq u_i\, y_i \quad i = 1, \ldots n$$

$$x_{ij} \leq y_i \quad i = 1, \ldots n,\ j = 1, \ldots n_i$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \ldots n,\ j = 1, \ldots n_i$$

$$y_i \in \{0, 1\} \quad i = 1, \ldots n$$

The modifications required by branching rules 3 based on column subset $\hat{Q} = \{q \in Q : x_i^q > 0 \text{ and } x_j^q > 0\}$ however, give rise to a subproblem that can no longer be treated with algorithms specialized for knapsack problems.

The above formulation is a generalization of the multiple-class binary knapsack problem studied in [42]. In this chapter, we propose exact solution approaches for this problem. In fact we consider a slightly more general model for which a setup time, $s_i \geq 0$, and a setup cost, $f_i \in I\!R$ are associated with the use of product i. Items are partitioned into classes associated with each product i. A setup time and cost is defined for each class. The item weights in each class are a multiple of a class weight. The total item weight that

can be selected is bounded (we consider explicit lower and upper bounds a_i and b_i for each product i). The objective is to maximize the sum of the profits associated with selected items minus the fixed costs incurred for setting up classes.

The resulting model is called the multiple class binary knapsack problem with setups (MCBKPSU). It takes the following form:

$$\max \quad \sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ij}\, x_{ij} - \sum_{i=1}^{n} f_i y_i \tag{3.1}$$

[MCKPSU] s.t.

$$\sum_{i=1}^{n}(\sum_{j=1}^{n_i} m_{ij}\, w_i\, x_{ij} + s_i\, y_i) \leq W \tag{3.2}$$

$$a_i\, y_i \leq \sum_{j=1}^{n_i} m_{ij}\, x_{ij} \leq b_i\, y_i \qquad \text{for } i = 1, \ldots n \tag{3.3}$$

$$x_{ij} \leq y_i \qquad \text{for } i = 1, \ldots n \text{ and } j = 1, \ldots, n_i \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \quad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, n_i \tag{3.5}$$

$$y_i \in \{0, 1\} \qquad \text{for } i = 1, \ldots, n\ , \tag{3.6}$$

We show the extend to which classical results for the knapsack problem can be generalized to this variant with setups. We give upper bounds that generalized that of Dantzig. We show that the classic branch-and-bound algorithm of Horowith and Sahni extends to this variant in the case $f_i \geq 0$. We provide dynamic programming algorithms for the general case. These results are useful to build good solvers for the particular case of [BSUSP1] (when $s_i = 0$ and $a_i = 0$).

In [32], we further develop the present chapter. We also consider the special case that arises when each class holds a single item, i.e. the standard integer

knapsack problem with setups, and its continuous version. A special case of the integer knapsack problem with setups was studied by Sural et al. [34]: they assume $f_i = 0$ and $w_i = 1$ for all i. They discuss the complexity of special cases to explore the frontier of easiness. They generalize the Dantzig's upper bound to this case and propose a primal heuristic. Both are used for setting up a depth-first search branch-and-bound algorithm that generalizes that of Horowitz and Sahni [16]. Their motivations for studying this model were applications in finance and in machine scheduling.

A variant of model MCBKPSU is considered by Chajakis and Guignard [7] where $m_{ij} = 1 \ \forall ij$, there are no class bounds ($a_i = 0$ and $b_i = \infty$), but the item weights are not restricted to be a multiple of a class weight. The application that motivated their study is the scheduling of parallel unrelated machines with setups where this knapsack problem arises as a subproblem. They propose and test two approaches: either a dynamic program solver or a two-stage approach where the problem is transformed into a standard multiple choice 0-1 knapsack problem and solved either by dynamic programming or branch-and-bound. The transformation consists in defining a "pseudo item" for each dominant feasible solutions within a class. These dominant solutions are the states of a dynamic programming recursion for solving the binary knapsack problem defined on a single class. There is a pseudo-polynomial number of them. They found that, for correlated instances with reasonable knapsack capacity (500), the direct dynamic programming approach is the most efficient. When the number of families or the knapsack capacity increases the two-stage approach using branch-and-bound for the second stage is the most efficient.

3.1 CHARACTERIZATIONS OF OPTIMAL SOLUTIONS FOR THE MULTIPLE-CLASS BINARY KNAPSACK WITH SETUPS

We can make the following observations and assumptions:

Observation 7 *If $a_i = 0$, then we can assume that $p_{ij} \geq 0$ for all j.*

Since, if $a_i = 0$ and $p_{ij} < 0$ for some binary item (i, j), $x_{ij} = 0$ in any optimal solution. ■

Assumption 1 ***(without lost of generality)*** $f_i < \max\{\sum_j p_{ij}\, x_{ij} : \sum_j m_{ij}\, x_{ij} \leq b_i,\ x_{ij} \in \{0,1\}\ \forall j\}$.

Indeed, if $f_i \geq \max\{\sum_j p_{ij}\, x_{ij} : \sum_j m_{ij}\, x_{ij} \leq b_i,\ x_{ij} \in \{0,1\}\ \forall j\}$ for some i, it is optimal to set $x_{ij} = 0\ \forall j$ and $y_i = 0$. ■

Observation 8 *There exists an optimal solution to MCKPSU where for each class i one of the following cases arises:*

$$\begin{array}{ll} (i) & x_{ij} = 0\ \forall j \text{ and } y_i = 0 \\ (ii) & \sum_j p_{ij}\, x_{ij} > f_i \text{ and } y_i = 1\ . \end{array}$$

Indeed, if $\sum_j p_{ij}\, x_{ij} \leq f_i$ and $y_i = 1$, the solution can only improve if one sets $x_{ij} = 0\ \forall j$ and $y_i = 0$. ■

An optimal solution may have $y_i = 1$ while the x_{ij}'s are set to the minimum value that allows to satisfy the class lower bound a_i which could be zero. However,

Observation 9 *when $a_i = 0$ and $f_i \geq 0$, there exists an optimal solution where $y_i = 0$ when $\sum_j x_{ij} = 0$.*

The LP solution to MCKPSU can also be characterized. The following observation derives from the assumption that all items have a weight that is a multiple of the class weight. Hence, the capacity consumption of a class i, i.e. $w_i(\sum_j m_{ij}x_{ij})$, is the same for all solution x_{ij}'s yielding the same total multiplicity $\sum_j m_{ij}\ x_{ij}$. As a result, the optimization within each class can be done independently of the global optimization of the use of the knapsack capacity W.

Observation 10 *Consider solutions to LP relaxation of MCKPSU. Their projection in the subspace* $(x_i = \sum_j m_{ij}\ x_{ij},\ y_i)$ *associated with class* i *are convex combinations of the following extreme points:*

$$\begin{array}{ll}(i) & x_i = 0 \text{ (i.e. } x_{ij} = 0\ \forall j) \text{ and } y_i = 0\\ (ii) & x_i = \sum_j m_{ij}\ x_{ij} = a_i \text{ and } y_i = 1\\ (iii) & x_i = \sum_j m_{ij}\ x_{ij} = b_i \text{ and } y_i = 1\end{array}$$

If the profit per unit of knapsack capacity of extreme solution (ii) *is less than that of* (iii)*, i.e.,*

$$\frac{p_i^a - f_i}{w_i\ a_i + s_i} \leq \frac{p_i^b - f_i}{w_i\ b_i + s_i}$$

where

$$p_i^a = \max\{\sum_j p_{ij}\ x_{ij} : \sum_j m_{ij}\ x_{ij} = a_i,\ x_{ij} \in [0,1]\ \forall j\}$$

and

$$p_i^b = \max\{\sum_j p_{ij}\ x_{ij} : \sum_j m_{ij}\ x_{ij} = b_i,\ x_{ij} \in [0,1]\ \forall j\}\ ,$$

then one only needs to consider solutions that are convex combination of cases (i) *and* (iii)*. The reverse case, i.e.* $\frac{p_i^a - f_i}{w_i\ a_i + s_i} > \frac{p_i^b - f_i}{w_i\ b_i + s_i}$*, can only arise if* $a_i > 0$ *or* $f_i < 0$*. Moreover, when* $\frac{p_i^a - f_i}{w_i\ a_i + s_i} > \frac{p_i^b - f_i}{w_i\ b_i + s_i}$*, the projection of the LP solution in the subspace associated with class* i *will be in the convex hull of cases* (i) *and* (ii) *while its knapsack capacity usage is less or equal to* $(w_i\ a_i + s_i)$*.*

Figure 3.1 illustrates both the case where the slope $\frac{p_i^a - f_i}{w_i\ a_i + s_i} > \frac{p_i^b - f_i}{w_i\ b_i + s_i}$ and vice versa.

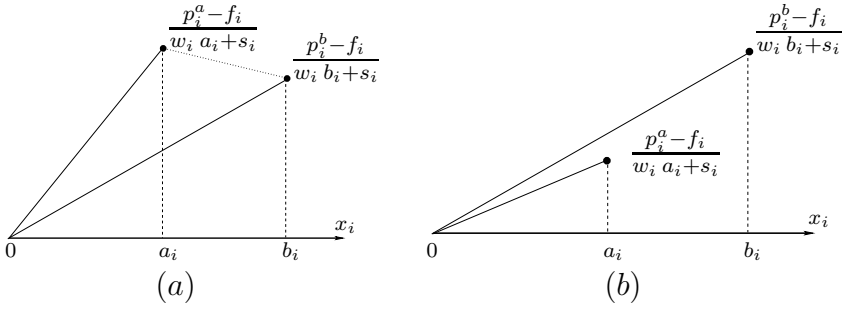


Figure 3.1: Ratio of class i profit per unit of knapsack capacity consumption

3.2 Upper Bound of the multiple-class binary knapsack with setups

We show here that, under some restrictive conditions, the LP bound for MCKPSU can be computed using a greedy algorithm. In the continuous relaxation of MCKPSU, item (i, j) can yield a profit per unit of

$$\text{either} \quad \frac{p_{ij}\,\frac{a_i}{m_{ij}} - f_i}{w_i\, a_i + s_i} \quad \text{or} \quad \frac{p_{ij}\,\frac{b_i}{m_{ij}} - f_i}{w_i\, b_i + s_i}$$

or a convex combination of these two, depending of whether it is contributing to the class effort of targeting extreme solution (ii) or (iii) of Observation 10 or their combination. Case (ii) can be split in two sub-cases, either $a_i > 0$ (let $I^a = \{i : a_i > 0\}$) or $f_i < 0$ (let $I^f = \{i : a_i = 0 \text{ and } f_i < 0\}$). Thus, $I^a \cap I^f = \emptyset$. In the second sub-case, the targeted class solution is $(x_i = 0, y_i = 1)$. We can represent this solution directly using variable $y_i^f = 1$ if $(x_i = 0, y_i = 1)$ and zero otherwise. Similarly, we define $z_{ij}^a = 1$ (defined for $i \in I^a$) if item (i, j) contribution is to achieve extreme solution (ii) of Observation 10 with $a_i > 0$, while $z_{ij}^b = 1$ (defined for $i \in I = \{1, \ldots, n\}$) if item (i, j) contribution is to achieve extreme solution (iii). With these notations, the LP relaxation of MCKPSU can be reformulated as

$$\max \sum_{i \in I^a} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{a_i}\, m_{ij})\, z_{i\,j}^a + \sum_{i \in I} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{b_i}\, m_{ij})\, z_{i\,j}^b - \sum_{i \in I^f} f_i y_i^f \quad (3.7)$$

$$\text{s.t.} \tag{3.8}$$

$$\sum_{i\in I^a}\sum_{j=1}^{n_i}(w_i+\frac{s_i}{a_i})\,m_{ij}z^a_{ij}+\sum_{i\in I}\sum_{j=1}^{n_i}(w_i+\frac{s_i}{b_i})\,m_{ij}z^b_{ij}+\sum_{i\in I^f}s_iy^f_i \;\leq\; W \tag{3.9}$$

$$\sum_{j=1}^{n_i}[\frac{m_{i\,j}}{a_i}\,z^a_{i\,j}+\frac{m_{i\,j}}{b_i}\,z^b_{i\,j}] \;\leq\; 1 \quad \forall i\in I^a \tag{3.10}$$

$$\sum_{j=1}^{n_i}\frac{m_{i\,j}}{b_i}\,z^b_{i\,j}+y^f_i \;\leq\; 1 \quad \forall i\in I^f \tag{3.11}$$

$$\begin{aligned} z^a_{i\,j}+z^b_{i\,j} &\leq 1 \quad \forall i\in I^a,\, j \\ z^a_{i\,j} &\in [0,1]\;\forall i\in I^a,\, j \\ z^b_{i\,j} &\in [0,1]\;\forall i\in I,\, j \\ y^f_i &\in [0,1]\;\forall i\in I^f \end{aligned} \tag{3.12}$$

A solution (z, y^f) translates into a solution for the LP relaxation of MCKPSU as follows:

$$x_{ij}=z^a_{ij}+z^b_{ij} \quad \text{and} \quad y_i=\sum_j(\frac{m_{i\,j}}{a_i}\,z^a_{i\,j}+\frac{m_{i\,j}}{b_i}\,z^b_{i\,j})+y^f_i$$

Constraints (3.10-3.11) are required to enforce $y_i \in [0,1]$. Observe that constraints (3.3) are built in the definition of the change of variables. Indeed, if we replace x and y by their expression in z in (3.3), in the case $a_i > 0$, we obtain:

$$a_i\,\sum_j(z^a_{ij}\,\frac{m_{ij}}{a_i}+z^b_{ij}\,\frac{m_{ij}}{b_i})\leq\sum_j m_{ij}\,(z^a_{ij}+z^b_{ij})\leq b_i\sum_j(z^a_{ij}\,\frac{m_{ij}}{a_i}+z^b_{ij}\,\frac{m_{ij}}{b_i})$$

which is always satisfied because $\frac{a_i}{b_i}\leq 1$ in the left-hand-side and $\frac{b_i}{a_i}\geq 1$ in the right-hand-side. In the case $a_i = 0$, (3.3) is trivially verified. Hence, we have shown that

Proposition 5 *the LP relaxation of MCKPSU is equivalent to the continuous relaxation of binary knapsack problem with class bounds and SOS constraints (3.7-3.12).*

On one hand, it is known that the LP relaxation of binary knapsack problem with SOS constraints admits a greedy solution [17]. On the other hand, Vanderbeck showed in [42] that the LP relaxation of a binary knapsack with class bounds can also be solved using a greedy procedure. But, solving problem (3.7-3.12) requires dealing with both SOS constraints and class bounds. We have not found a greedy procedure to solve this case involving both complexities. Instead, we develop a greedy LP solution for the special case where SOS constraints are redundant.

We make the simplifying assumption that all class i items target a filling up to b_i because this corresponds to a better ratio:

Assumption 2 ***(*restrictive)**

$$\frac{p_{ij}\,\frac{a_i}{m_{ij}} - f_i}{w_i\,a_i + s_i} \;\leq\; \frac{p_{ij}\,\frac{b_i}{m_{ij}} - f_i}{w_i\,b_i + s_i} \quad \forall (i,j)$$

This assumption implies that the aggregate class contribution is in the case illustrated by part (b) of Figure 3.1.

Observation 11 *Under Assumption 2, problem (3.7-3.12) admits a solution where all variables z^a_{ij} and y^f_i have value zero.*

Indeed, if Assumption 2 holds and $z^a_{ij} > 0$, one can modify the solution by setting $z^{a\,\prime}_{ij} = 0$ and $z^{b\,\prime}_{ij} = z^b_{ij} + \frac{(w_i + \frac{s_i}{a_i})\,m_{ij}}{(w_i + \frac{s_i}{b_i})\,m_{ij}}\; z^a_{ij}$. This solution modification is feasible with regard to knapsack constraint (3.9) by construction but also with regard to constraint (3.10) as it can be easily checked. Moreover, the profit value of the modified solution is not less than the original. Similarly, if $y^f_i > 0$, decreasing its value allows to increase some z^b_{ij} value of better profit ratio. ∎

Under Assumption 2 we can give a greedy LP solution to MCBKSU, extending the result of Vanderbeck in [42].

Proposition 6 *If Assumption 2 holds, an optimal solution to the LP relaxation of MCBKSU is given by the following procedure. Sort the items (i, j) in non-increasing order of their ratio:*

$$\frac{(p_{ij} - \frac{f_i}{b_i}\, m_{ij})}{(w_i + \frac{s_i}{b_i})\, m_{ij}} \tag{3.13}$$

Let $m = \sum_i n_i$ and $k = 1, \ldots, m$ be the item indices in that ordering. K^i is the set of items k that belong to class i:

$$K^i = \{k : \ \exists j \in \{1, \ldots, n_i\} \text{ with } k = (i, j)\} \ .$$

For $i \in \{1, \ldots, n\}$, let the critical item for class i be $c_i^b \in K^i$, be such that

$$\sum_{k \in K^i,\, k < c_i^b} m_k \leq b_i \ \text{ but } \sum_{k \in K^i,\, k \leq c_i^b} m_k > b_i \ . \tag{3.14}$$

Let $K^i(l) = \{k \in K^i : k < c_i^b$ and $k < l\}$, $I^b(l) = \{i \ : c_i^b < l\}$, and

$$W(l) = \sum_i \sum_{k \in K^i(l)} (w_i + \frac{s_i}{b_i})\, m_k + \sum_{i \in I^b(l)} (w_i + \frac{s_i}{b_i})\, \frac{(b_i - \sum_{k \in K^i(l)} m_k)}{m_{c_i^b}} \ . \tag{3.15}$$

Then, let the global critical item, $c \in \{1, \ldots, m\}$, be the highest index item such that

$$W(c) \leq W \quad \text{but} \quad W(c) + (w_{i_c} + \frac{s_{i_c}}{b_{i_c}})\, m_c > W \tag{3.16}$$

where i_c refers to the class containing the global critical item (i.e. $c \in K^{i_c}$) and set

$$x_k = 1 \qquad \text{for } k \in K^i(c) \text{ and } i = 1, \ldots, n \tag{3.17}$$

$$x_{c_i^b} = \frac{1}{m_{c_i^b}} (b_i - \sum_{k \in K^i(c)} m_k) \qquad \text{for } i \in I^b(c), \tag{3.18}$$

$$x_c = \frac{1}{w_{i_c} + \frac{s_{i_c}}{b_{i_c}}} (W - W(c)) \qquad \text{if } c \in K^{i_c} \tag{3.19}$$

$$x_k = 0 \qquad \text{otherwise} \tag{3.20}$$

$$y_i = 1 \qquad \text{for } i \in I^b(c) \tag{3.21}$$

$$y_{i_c} = \frac{\sum_{k \in K^{i_c}(c)} m_k + m_c\, x_c}{b_{i_c}} \qquad \text{for } i : c \in K^i \tag{3.22}$$

$$y_i = 0 \qquad \text{otherwise.} \tag{3.23}$$

Proof: Under the assumption made, problem (3.7-3.12) involves only the z_{ij}^b variables. Therefore it admits a greedy solution as proved in [42]. Converting the greedy solution z into the original variables x and y provides the desired result. ■

3.3 A DYNAMIC PROGRAM FOR THE MULTIPLE-CLASS BINARY KNAPSACK WITH SETUPS

A solution by dynamic programming assumes integer data: s_i, w_i, and $W \in I\!N$. Let us first consider the unbounded case where $a_i = 0$ and $b_i \geq \left\lfloor \frac{W-s_i}{w_i} \right\rfloor$ for all i. Then, one can write a dynamic programming recursion where $V^i(C)$ defines the best value that can be achieved using items from class $k = 1, \ldots, i$ with a capacity consumption C and $V^{ij}(C)$ defines the best value that can be achieved using items from class $k = 1, \ldots, i-1$ plus at least one item among the first j items of class i, with a capacity consumption C. The $V^{ij}(C)$ and $V^i(C)$ values can be computed recursively as follows:

$$\begin{aligned} V^{ij}(C) &= \max\{ \quad V^{i,j-1}(C), \;\; V^{i,j-1}(C - w_i\, m_{ij}) + p_{ij}, \\ &\qquad\qquad V^{i-1}(C - w_i\, m_{ij} - s_i) - f_i + p_{ij}\} \qquad (3.24) \\ V^i(C) &= \max\{ \quad V^{i-1}(C),\; V^{i,n_i}(C),\; V^{i-1}(C - s_i) - f_i\} \; . \end{aligned}$$

Such dynamic program requires $O(\sum_i n_i\; W)$ operations. Observe that this complexity $O(\sum_i n_i\, W)$ does not imply that the multiple class problem requires a higher complexity than the integer knapsack problem (for which the unbounded problem can be solved in $O(nW)$). Indeed, the input data file is of length proportional to $\sum_i n_i$ since it includes the description of the profit values p_{ij}.

For the bounded case, one must first solve a knapsack subproblem within each class before solving the overall problem: Let $U^i(M)$ be the optimal value

that can be achieved with class i items using a multiplicity of **exactly** M units. $U^i(M)$ can be computed by dynamic programming: Initially, $U^{i,j}(0) = 0\ \forall j$ and $U^{i,0}(M) = -\infty$ for $M = 1, \ldots, b_i$; then, one sets $U^{ij}(M) = U^{i,j-1}(M)$ for $M = 1, \ldots, m_{ij} - 1$ and one computes

$$U^{ij}(M) = \max\{U^{i,j-1}(M), U^{i,j-1}(M - m_{ij}) + p_{ij}\} \tag{3.25}$$

$M = m_{ij}, \ldots, b_i$ and for $j = 1, \ldots, n_i$. Then

$$U^i(M) = U^{i,n_i}(M)\ \ \forall M\ . \tag{3.26}$$

These computations requires $O(n_i\ b_i)$ operations for each class i. Therefore the overall complexity for computing all the $U^i(M)$ is $O(\sum_i n_i b_i)$ (which is bounded by $O(\sum_i n_i W)$ as $b_i \leq \lfloor \frac{W}{w_i} \rfloor$). As an aside, observe that when $m_{ij} = 2^{j-1}\ \forall i, j$, a given multiplicity M can only be obtained from a single combination of 0-1 items (i, j) and $U^i(M)$ can be computed directly, although this does not change the computational complexity. From $U^i(M)$ values, one can compute $V^i(C)$, the best value that can be achieved with items of class 1 up to i and capacity C:

$$V^i(C) = \max\{\underbrace{V^{i-1}(C)}_{y_i=0}, \underbrace{\max_{a_i \leq M \leq b_i} \{V^{i-1}(C - w_i\ M - s_i) + U^i(M) - f_i\}}_{y_i=1}\}\ . \tag{3.27}$$

This requires $O(nW \max_i\{b_i\})$ operations (which is bounded by $O(nW^2)$ but can be much smaller than $O(nW^2)$ in practice).

When an integer knapsack problem is transformed into a binary multiple class knapsack problem one can treat the class bounds a_i and b_i implicitly and use the dynamic recursion (3.24) for the unbounded case to benefit from the lower complexity $O(\sum_i n_i\ W)$. Indeed, in such case, the profit is defined for the class and not for the 0-1 items, therefore we can assume $a_i = 0$ for all i (quantity a_i can be incorporated to the fixed cost and weight). To eliminate the upper

bound b_i one just needs to amend the 0-1 transformation defined by (2.1): set $n_i = \lfloor \log_2 b_i \rfloor + 1$ and $m_{ij} = 2^{j-1}$ for $j = 1, \ldots, n_i - 1$ but $m_{i,n_i} = b_i - \sum_{j=1}^{n_i-1} m_{ij}$.

3.4 PRIMAL HEURISTICS FOR THE MULTIPLE-CLASS BINARY KNAPSACK WITH SETUPS

In the rest of this Section, we make the following assumption:

Assumption 3 ***(*restrictive)** $f_i \geq 0$ *for all* i .

Furthermore we further assume $a_i = 0$ for all i for simplicity.

Primal heuristics can be developed based on decomposing the problem into knapsack subproblems for each class. We assume that classes are sorted by non decreasing ratio

$$\frac{\tilde{p}_i - f_i}{w_i\, b_i + s_i}$$

where $\tilde{p}_i = \max\{\sum_j p_{ij}\ x_{ij} : \sum_j m_{ij}\ x_{ij} = b_i,\ x_{ij} \in \{0,1\}\ \forall j\}$, or its LP relaxation value or an estimate obtained by a greedy algorithm; and the knapsack is filled with these classes in this order up to reaching capacity. For the critical class, one solves $\max\{\sum_{j=1}^{n_i} p_{ij} x_{ij} : \sum_j m_{ij}\, x_{ij} \leq \lfloor \frac{C}{w_i} \rfloor, x_{ij} \in \{0,1\} \forall j\}$ (where C is the residual capacity) either exactly or with a greedy procedure.

The alternative approach is to base the heuristic on the greedy ordering of the items, (3.13), that was used for the LP solution. It allows to account more accurately for difference of profit ratio within the same classes (cases where $p_{ij} \neq \frac{\tilde{p}_i}{m_{ij}}$ and the difference is important). On the other hand, this second approach is quite myopic with regards to the setup cost, f_i, and the setup capacity usage,

s_i. Indeed, the non-increasing order of ratio (3.13) is a greedy approach for a relaxation whose formulation is

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{b_i} m_{ij})\, z_{ij} \tag{3.28}$$

$$\text{s.t.} \sum_{i=1}^{n} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{b_i})\, m_{ij}\, z_{ij} \leq W \tag{3.29}$$

$$\sum_{j} m_{ij}\, z_{ij} \leq b_i \qquad \forall i \tag{3.30}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,\, j\ . \tag{3.31}$$

Problem MCKPSU (under Assumption 3 and with $a_i = 0\ \forall i$) and problem (3.28-3.31) admit the same LP solution has shown above but not the same integer solution. A partial feasible integer solution to (3.28-3.31) can be transformed into a feasible for MCKPSU by rounding up the implicit y variables if the residual capacity allows it.

The latter approach is used in the primal heuristic of Table 3.1. We implicitly set z variables to 1 in greedy order and we keep tract of the setup fraction that has not yet been accounted for. We use the following notations: i_k denotes the class index of item k, C denotes the remaining knapsack capacity, C_i the residual upper bound on class i items, S the reserved knapsack capacity for setups, and F the setup cost to be withdrew from the current profit Z. On exiting the algorithm, a primal solution is given by vectors x and y, whose value is $Z - F$. S and F can be understood as corrections needed to transform the current solution to relaxation (3.28-3.31) into a solution for MCKPSU.

In the application for the column generation subproblem [BSUSP1] (where costs p_{ij} have been modified as a result of branching (typically $p_{ij} \neq \frac{\tilde{p}_i}{m_{ij}}$), while $s_i = 0\ \forall i$ and f_i takes non zero value only when we branch on a class), the second approach based on setting individual items is likely to be more effective than the

Table 3.1: Primal heuristic for MCBKSU when $f_i \geq 0$ and $a_i = 0\ \forall i$

Step A : **Initializations:** Renumber the m items (i, j) in decreasing order of their ratio (3.13), breaking tight in favor of the item $k = (i, j)$ with the largest weight $w_k = (w_i + \frac{s_i}{b_i})\, m_{ij}$.
Let $m_k = m_{ij}$, and $p_k = p_{ij} - \frac{si}{b_i} f_i$ for $k = 1, \ldots m$.
Let $C = W$, $C_i = b_i\ \forall i$, $S = 0$, $F = 0$, $Z = 0$, $x = y = 0$, $k = 1$.

Step B: **Setting items to one:** While $(w_{i_k}\, m_k + s_i\, (1 - y_{i_k}) \leq C - S)$ and $(m_k \leq C_{i_k})$ and $(k \leq m)$, do
if $(y_{i_k} == 0)$, let $y_{i_k} = 1$, S+= s_{i_k}, F+= f_{i_k};
$x_k = 1$, C−= w_k, S−= $\frac{s_{i_k}}{b_{i_k}}\, m_k$, C_{i_k}−= m_k, Z+= p_k, F−= $\frac{s_{i_k}}{b_{i_k}} f_{i_k}$, k++.

Step C: If $(k > m)$, STOP.

Step D: **Setting item to zero:** /* $((w_{i_k}\, m_k + s_i\, (1 - y_{i_k}) > C - S)$ or $(m_k > C_{i_k})$ */
Let k++.

Step E: If $(k > m)$, STOP. Else, goto Step B.

former based on setting classes.

3.5 BRANCH-AND-BOUND FOR THE MULTIPLE-CLASS BINARY KNAPSACK WITH SETUPS

We propose an extension of the depth-first-search branch-and-bound algorithm of Horowitz and Sahni [16] to MCBKSU under Assumption 3 with $a_i = 0$ for all i. The algorithm of Table 3.2 can be understood as a branch-and-bound procedure for problem (3.28-3.31) that has been adapted to MCBKSU by making corrections to ensure primal feasibility (as in the above primal heuristic). The interest using relaxation (3.28-3.31) implicitly is to have a greedy ordering of items that serves both primal and dual procedure as required for Horowitz and Sahni's algorithm.

Table 3.2: Branch-and-Bound for MCBKSU $f_i \geq 0$ and $a_i = 0\ \forall i$

Initialization: Sort items in decreasing order of their ratio (3.13). Let $m_k = m_{ij}$,
$w_k = (w_i + \frac{s_i}{b_i})\ m_{ij}$, and $p_k = p_{ij} - \frac{s_i}{b_i} f_i\ \forall k$, $w_{\min} = \min_k\{w_{i_k}\ m_k\}$,
$C = W$, $C_i = b_i\ \forall i$, $S = 0$, $F = 0$, $Z = 0$, $x = y = 0$, $INC = 0$, $k = 1$.

Compute UB: Let $U = Z$, $K = C$, $K_i = C_i\ \forall i$, and let $l = k$.

Step A: While $(s_{i_l}\ (1 - y_{i_k}) \leq C - S)$ and $(w_l \leq K)$ and $(m_l \leq C_{i_l})$ and $(l \leq m)$,
do $U \mathrel{+}= p_l$, $K \mathrel{-}= w_l$, $l = l + 1$.

Step B: If $(s_{i_l}\ (1 - y_{i_k}) > C - S)$, do $l = l + 1$ and goto Step A.

Step C: If $(l > m)$, goto Test Pruning.

Step D: If $(m_l > C_{i_l})$ and $(w_l\ \frac{C_{i_l}}{m_l} \leq K)$, do
$U \mathrel{+}= p_l\ \frac{C_{i_l}}{m_l}$, $K \mathrel{-}= w_l\ \frac{C_{i_l}}{m_l}$, $l = l + 1$, go to Step A.

Step E: If $(m_l > C_{i_l})$ and $(w_l\ \frac{C_{i_l}}{m_l} > K)$, do
$U \mathrel{+}= p_l\ \frac{K}{w_l} \frac{m_l}{C_{i_l}}$, go to Test Pruning.

Step F: /* $(w_l > K)$ */ $U \mathrel{+}= p_l\ \frac{K}{w_l}$.

Test Pruning: if $(U \leq INC)$, goto Backtracking.

Forward Move: While $(w_{i_k}\ m_k + s_{i_k}\ (1 - y_{i_k}) \leq C - S)$ and $(m_k \leq C_{i_k})$ and $(k \leq m)$, do
if $(y_{i_k} == 0)$, let $y_{i_k} = 1$, $S \mathrel{+}= s_{i_k}$, $F \mathrel{+}= f_{i_k}$;
$x_k = 1$, $C \mathrel{-}= w_k$, $S \mathrel{-}= \frac{s_{i_k}}{b_{i_k}}\ m_k$, $C_{i_k} \mathrel{-}= m_k$, $Z \mathrel{+}= p_k$, $F \mathrel{-}= \frac{s_{i_k}}{b_{i_k}} f_{i_k}$, $k{+}{+}$.
If $(k > m)$ or $(C - S < w_{\min})$, /* leaf node */ goto Record Incumb.

Set item to 0: /* $((w_{i_k}\ m_k + s_i\ (1 - y_{i_k}) > C - S)$ or $(m_k > C_{i_k})$ */
Let $k{+}{+}$. If $(k > m)$, goto Record Incumb. Else, goto Compute UB.

Record Incumb: If $(Z - F > INC)$, then $INC = Z - F$ and record (x, y).

Pre-backtrack: If $(k > m)$, do {$k{-}{-}$; if $(x_k == 1)$, **WithdrawItem(**k**)**; }

Backtracking: Do $k{-}{-}$, while $(x_k == 0)$ and $(k \geq 1)$.
If $(k == 0)$, STOP.
/* $(x_k == 1)$ */ **WithdrawItem(**k**)**, $k = k + 1$.
Go to Compute UB.

Table 3.3: subroutine **WithdrawItem(k)** of the Branch-and-Bound for MCBKSU

Let $x_k = 0$, $C \mathrel{+}= w_k$, $S \mathrel{+}= \frac{s_{i_k}}{b_{i_k}} m_k$,
$C_{i_k} \mathrel{+}= m_k$, $Z \mathrel{-}= p_k$, $F \mathrel{+}= \frac{s_{i_k}}{b_{i_k}} f_{i_k}$.
If $(C_{i_k} == b_i)$, $y_{i_k} = 0$, $S \mathrel{-}= s_{i_k}$, $F \mathrel{-}= f_{i_k}$.

The "Compute UB" step is implemented so as to compute the upper bound of Proposition 6 for the residual problem. In "Forward Moves", we implicitly set z_k's to one in formulation (3.28-3.31) which involves implicitly taking a fraction of y_{i_k}; but simultaneously we construct a primal solution (x, y) for MCBKSU, we reserve the extra capacity S, and we account for the extra fixed cost F that results from rounding up the fractional setup involved in the z solution. This is repeated while there remain some knapsack capacity to insert further items, i.e. while $C - S \geq w_{\min}$, where $w_{\min}$ is the smallest item weight and some class capacity. Otherwise, the next item is set to zero and the dual bound must be computed. For the dual bound computation, we only account for the knapsack capacity and class bound capacity used by fixing the z variables. However, if the remaining primal capacity $C - S$ is not sufficient to open a new class, the items of this class are ignored during the UB computation. A leaf node is reached when the knapsack is filled or there are no more items to consider. In the latter case, as the branch $z_n = 1$ has been explored, the branch $z_n = 0$ does not need to be explored as it is dominated. "Backtracking" must insure that the class setup is set to zero when the last positive item of that class is set to zero. This is done in the WithdrawItem(k) subroutine of Table 3.3.

Both dynamic program and branch and bound have been tested as subproblem solvers. When $f_i < 0$, we must use the dynamic program, in other case the branch and bound algorithm performs better, even when we use stabilization methods, as

we shall see.

4

COMPARING IP COLUMN GENERATION STRATEGIES

The column generation procedure suffers from several drawbacks illustrated in figure (4.1). The first one is the *heading-in effect*: at the beginning of the procedure, poor dual informations lead to bad Lagrangian bounds, so the gap between primal and dual bounds is important. Furthermore the procedure is known to have a slow convergence, usually called the *tailing-off effect*: when approaching the optimal solution lots of iterations are needed to prove optimality. There are also *degeneracy* problems in the primal: new columns added in the restricted master does not improve its value that remains constant during several iterations. The *jumpy behavior* corresponds to oscillations of the intermediate Lagrangian bounds, they do not converge monotonically to the optimal value. This phenomenon is due to the instability of the dual variables values from one iteration to the following.

Several techniques can be used to reduce these drawbacks. Some of these techniques are presented and compared in this chapter. In the first part of this chapter, we study different initializations that may help in reducing the heading-in effect. Then, we compare several methods to stabilize the column generation procedure. Follows the numerical comparison of various strategies for introducing columns in the master (using exact versus heuristic oracles, adding a single column at each iteration versus multiple columns). Finally we study the efficiency of the branching scheme.

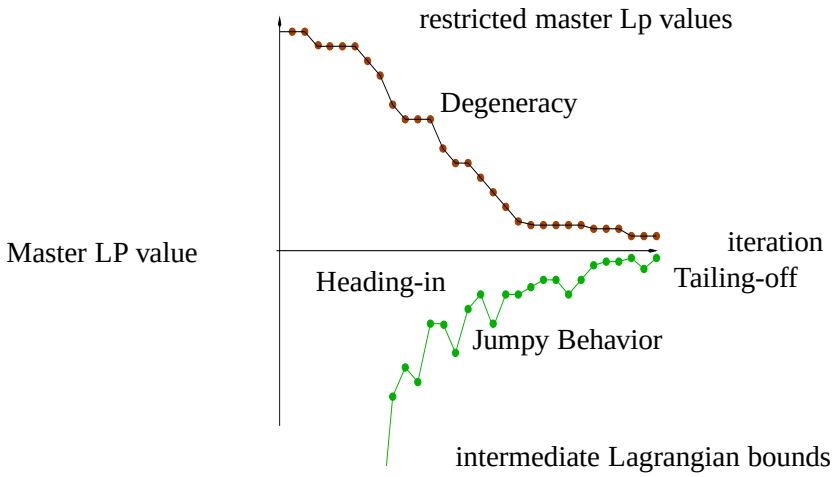


Figure 4.1: Drawbacks of the column generation procedure

4.1 FRAMEWORK FOR COMPUTATIONAL TESTS: DATA SETS AND TABLE OF RESULTS

We have made comparative tests on real and randomly generated instances. For the standard cutting stock problem, we use 9 real data containing 7 to 33 items and we generated 20 random instances with 50 items, an average demand of 100, the width of wide roll is 10 000 and the items widths are uniformly generated in the interval [500, 5000]. We also used 20 random instances used by Belov and

Scheithauer in [4]. For these, the items widths are uniformly distributed in [100, 7000] and the demand in [1, 100]. For the variant with tolerance on production we use the same instances with an interval on items demands ranging between $[\lfloor d_i * 0.95 \rfloor, \lceil d_i * 1.05 \rceil]$.

For the bin-packing problem we have used only random data coming from the OR library [2]. When an instance involves items with the same width, we aggregated their demand; hence they define cutting stock instances with low average demand. For each type of data, we have made tests on 20 instances. BP-250, BP-500 and BP-1000 refer to instances with 250, 500 and 1000 items respectively whose sizes are integer and uniformly distributed in [20, 100] while the size of the bin is 150. BP-t120, BP-t249 and BP-t501 design so-called triplets instances with 120, 249 and 501 items whose sizes are real and range from 25 to 49.9 while the bin capacity is 100. The triplets instances are generated from an optimum solution where there are exactly three items in each bin and they fill the bin capacity exactly. These are known to be hard to solve for standard column generation algorithms, but their solution using the compact formulation is trivial.

The instances used to test the multiple width cutting stock problem comes from [3]. We choose 20 random instances with 4 roll types whose widths are in [5 000, 10 000] and the numbers of rolls available in each type are uniformly distributed in [625, 2 500], and 100 items whose widths were generated in [1,7 500] and demand in [1, 100] (with an average demand of 50).

In each comparative table, we report the following measures:

- **ITER** is the number of iterations of the column generation procedure to solve the linear master problem,
- **Col** is the number of columns generated during the procedure,

- t_{Oracle} is the time in seconds taken by the oracle to solve the subproblems,
- t_{Master} is the time in seconds to solve the restricted master problem,
- t_{Total} is the total time (Oracle + Master).

All implementations have been done using the environment of *BaPCod*, a generic branch-and-price Code [43]. The oracle used to solve the subproblems is a specialized procedure presented in chapter 3, and the restricted linear master problems are solved with Xpress^{MP} [31].

4.2 INITIALIZATIONS

The column generation procedure used to solve the linear relaxation of the Dantzig-Wolfe reformulation must be initialized with a primal feasible solution. This solution can be obtained with the use of a heuristic procedure or one can choose to introduce artificial columns in the master. To control the heading-in effect of the column generation procedure different initializations have been tested. We consider the use of columns of an initial heuristic solution to the master and/or the use of artificial columns. To obtain a heuristic solution a **first fit decreasing** algorithm is used. For artificial columns we have tested two strategies: adding just one **global artificial column** or one artificial column for each master constraint called **local artificial columns**.

4.2.1 INITIALIZATION WITH A HEURISTIC SOLUTION

A heuristic solution is constructed by applying a first fit decreasing algorithm. Items are sorted in order of non-increasing width. Then, each item is placed by searching into the list of existing patterns the first one where it fits. When none can be found a new pattern is initialized with the current item.

For the problem with interval on production we compute a first fit heuristic solution by the same procedure but considering the lower bound on the demand. Then we try to fill the existing patterns with the remaining demand, $d_i = \overline{d}_i - \underline{d}_i$: if an item i cannot be placed, one passes to the following item without creating new patterns.
When there are multiple widths roll types we begin to fill the larger wide roll, and when the upper bound is reached we fill the smaller ones.

4.2.2 INITIALIZATION WITH ARTIFICIAL COLUMNS

A basic initialization is to introduce a **single artificial column** at the beginning of the procedure to initialize the restricted master problem. The cost of this artificial column is increased if the column is still in the solution on completion of the column generation algorithm, then the procedure is re-iterated. However if after a fixed number of iterations this column remains in the solution the problem is considered as unfeasible. For the variant with multiple widths we introduce one artificial column for each subproblem or one global artificial column.

Let γ be the artificial column, $\tilde{c}$ its cost and $\tilde{a}_i$ its coefficient in constraint i. The master formulation with artificial columns takes the form:

$$Z^M = \min \sum_{q \in Q} c_q \, \lambda_q + \tilde{c} \, \gamma$$

$$\text{s.t.} \quad \sum_{q \in Q} x_i^q \, \lambda_q + \tilde{a}_i \, \gamma \;\geq\; d_i \qquad\qquad i = 1, \ldots, n$$

$$\sum_{q \in Q} \lambda_q \;\leq\; K$$

$$\lambda_q \in I\!N \qquad\qquad \forall q \in Q$$

$$\gamma \geq\; 0$$

whose dual is:

$$\begin{aligned} & \max \sum_{i=1}^{n} d_i \, \pi_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i^q \, \pi_i \;\leq\; c_q && \forall q \in Q \\ & \sum_{i=1}^{n} \tilde{a}_i \, \pi_i \;\leq\; \tilde{c} \\ & \pi_i \geq 0 && \forall i \end{aligned} \tag{4.1}$$

The values $\tilde{c}$ and $\tilde{a}$ are presented in Table 4.1. For the cost $\tilde{c}$ of this artificial column we use an estimate of the order of magnitude of the objective. In this way, dual constraints (4.1) takes the form: $\sum_{i=1}^{n} d_i \, \pi_i \;\leq\; \tilde{c}$ and then π is chosen such that the dual objective does not exceed our cost estimate.

Problem	constr. coeff. ($\tilde{a_i}$)	cost ($\tilde{c}$)
CSP	d_i	$\left\lceil \frac{\sum_{i=1}^{n} w_i \, d_i}{W} \right\rceil$
BP	1	$\left\lceil \frac{\sum_{i=1}^{n} w_i}{W} \right\rceil$
CSP with tol. on prod.	d_i	$\frac{\left\lceil \frac{\sum_{i=1}^{n} w_i \, d_i}{W} \right\rceil - \frac{\sum_{i=1}^{n} w_i \, d_i}{W}}{W}$
multi Width CSP	d_i	$\left\lceil \frac{\sum_{i=1}^{n} w_i \, d_i}{\max_k W_k} \right\rceil$

Table 4.1: Constraints coefficient and cost of the single artificial column

For the variants with restrictive constraints and the minimization of setups the cost and constraints coefficients are the same than for the standard cutting stock

problem.

Instead of a single artificial column one can use **local artificial columns**, one for each master constraint. Let γ_i be the artificial column associated to the covering constraint i and $\tilde{c}_i$ its cost, the master [M1] becomes:

$$Z^M = \min \sum_{q \in Q} c_q \, \lambda_q + \sum_{i=1}^{n} \tilde{c}_i \, \gamma_i$$

$$\begin{aligned}
[M1LocArtCol] \quad \text{s.t.} \quad & \sum_{q \in Q} x_i^q \, \lambda_q + \gamma_i \;\geq\; d_i && i = 1, \ldots, n \\
& \sum_{q \in Q} \lambda_q \;\leq\; K \\
& \lambda_q \in I\!N && \forall \, q \in Q \\
& \gamma_i \;\geq\; 0 && i = 1, \ldots, n
\end{aligned} \tag{4.2}$$

In a dual point of view it amounts to add dual constraints:

$$\max \sum_{i=1}^{n} d_i \, \pi_i$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{i=1}^{n} x_i^q \, \pi_i \;\leq\; c_q && \forall q \in Q \\
& \pi_i \;\leq\; \tilde{c}_i && \forall i \\
& \pi_i \geq\; 0 && \forall i
\end{aligned}$$

We see clearly that the costs of artificial columns define upper bounds on the associated dual variables. So these values should be an estimate of the optimal dual values.

For the standard cutting problem the LP solution value of [F1] (1.1) is known

to be:

$$Z_{LP} = \frac{\sum_{i=1}^{n} w_i d_i}{W}$$

which is obtained with the solution $x_{ik} = \frac{d_i}{K}$ and $y_k = \frac{\frac{\sum_{i=1}^{n} w_i d_i}{W}}{K}$. Indeed, the optimality can be proved by showing that a feasible dual solution of the same cost can be found. If we note π_i and ν_k the dual variables associated to the constraints of [F1], its dual is:

$$\begin{aligned}
&\max \sum_{i=1}^{n} d_i \, \pi_i \\
&\text{s.t.} \quad x_{ik} \, \pi_i \quad \geq \quad w_i \, \nu_k \qquad\qquad \forall i, \; k \\
&\sum_{k=1}^{K} W \, \nu_k \quad \geq \quad 1 \\
&\pi_i, \nu_k \quad \geq \quad 0 \qquad\qquad \forall i, \; k
\end{aligned}$$

The solution $\pi_i = \frac{w_i}{W}, \; \forall i$ and $\nu_k = \frac{1}{W}, \; \forall k$ is feasible and its objective value is equal to Z_{LP}.

Hence, when there are no interval on production, we initialize the artificial columns cost with:

$$\tilde{c_i} = \frac{w_i}{W} \, \alpha_{init} \;\; \forall i \tag{4.3}$$

where α_{init} is a parameter set to $\alpha_{init} = 1.2$ in our computational tests.

For the problem that consists in minimizing the waste, when there are intervals on the demand, we add n other artificial columns $\gamma'_i, \; \forall i$, in the packing constraints. To obtain the costs values we use the economic interpretation of the dual variables: the benefit to relaxing the covering constraint should be proportional to the width of the associated item and inversely proportional for the packing constraints (the smaller items are more easily used to fill the gaps in cutting patterns

that would otherwise be counted as waste). This gives rise to the following values:

$$\tilde{c_i} = \frac{R}{\sum w_i d_i} * w_i \quad \text{and} \quad \tilde{c_i'} = \frac{R}{\sum \frac{W}{w_i} d_i} * \frac{W}{w_i}, \tag{4.4}$$

where R is an estimate upper bound on the optimal waste.

Furthermore, if artificial columns are in the primal solution on completion of the column generation process their cost is updated increasing it by a factor $\alpha = 1.5$.

4.2.3 COMPARATIVE TESTS

The following tables are comparative tests between the different initialization methods. For each class of problems we give the average results obtained for each method. The column **Problem** refers to the class of problems while **Init. Mode** corresponds to the method used:

- **1 art. col.** is the initialization with a single artificial column (one for each sub problem for the multiple widths problem),
- **1 global art. col.** is the initialization with a global artificial column for the multiple widths problem,
- **local art. col.** is the initialization with one artificial column per master constraints,
- **FFD** is the first fit decreasing algorithm.

If artificial columns remains in the solution their cost is increased by the factor $\alpha = 1.5$ and the column generation procedure is continued. For the FFD the total time includes the time used to solve it.

Problem	Init. Mode	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
real instances	1 art. col.	42	43	0.51	0.05	0.68
	local art. col.	29	29	3.75	0.02	3.86
	1 art. col. + FFD	24	50	0.14	0.04	**0.34**
	local art. col. + FFD	**23**	48	0.62	0.02	0.77
random instances of 50 items	1 art. col.	236	237	5.02	0.62	6.41
	local art. col.	142	142	4.31	0.27	**5.04**
	1 art. col. + FFD	159	235	4.75	0.55	7.74
	local art. col. + FFD	**136**	211	4.91	0.28	6.96
random instances of 150 items	1 art. col.	727	727	78.32	23.00	106.13
	local art. col.	476	476	59.34	2.55	64.93
	1 art. col. + FFD	475	645	44.72	17.12	72.74
	local art. col. + FFD	**347**	515	35.51	2.03	**44.39**

Table 4.2: Standard Cutting Stock

Problem	Init. Mode	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
120 items	1 art. col.	339	340	7.41	0.94	9.97
	local art. col.	**118**	118	3.12	0.29	**4.14**
	1 art. col. + FFD	277	324	8.23	0.87	10.77
	local art. col. + FFD	127	173	3.97	0.34	5.35
249 items	1 art. col.	650	651	58.38	3.62	66.14
	local art. col.	**191**	191	13.80	0.73	**16.14**
	1 art. col. + FFD	518	612	76.02	3.16	83.58
	local art. col. + FFD	198	292	17.29	0.81	20.67
501 items	1 art. col.	950	951	545.41	8.73	562.05
	local art. col.	246	246	43.41	1.25	**47.59**
	1 art. col. + FFD	859	923	416.06	4.20	430.03
	local art. col. + FFD	298	467	126.79	1.93	142.01

Table 4.3: Bin-Packing triplets instances

Problem	Init. Mode	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
120 items	1 art. col.	273	273	8.18	0.63	10.03
	local art. col.	112	112	3.63	0.27	4.48
	1 art. col. + FFD	106	168	0.76	0.27	2.26
	local art. col. + FFD	**48**	109	0.32	0.11	**1.08**
500 items	1 art. col.	268	268	7.15	0.80	9.23
	local art. col.	115	115	3.43	0.26	4.33
	1 art. col. + FFD	82	162	0.57	0.23	2.17
	local art. col. + FFD	**38**	117	0.26	0.09	**1.06**
1000 items	1 art. col.	258	258	7.82	0.90	10.02
	local art. col.	145	145	4.64	0.30	5.68
	1 art. col. + FFD	72	167	0.49	0.20	2.30
	local art. col. + FFD	**35**	129	0.25	0.09	**1.18**

Table 4.4: Bin-Packing

Problem	Init. Mode	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
real instances	1 art. col.	43	43	2.59	0.04	**2.83**
	local art. col	30	26	9.09	0.03	9.26
	1 art.col. + FFD	30	56	6.17	0.05	6.51
	local art. col. + FFD	**29**	51	8.12	0.03	8.42
random instances of 50 items	1 art. col.	140	140	8.14	0.25	**9.36**
	local art. col	129	115	26.35	0.30	27.33
	1 art.col. + FFD	**119**	194	7.72	0.27	12.05
	local art. col. + FFD	124	186	25.70	0.29	29.41
random instances of 150 items	1 art. col.	525	522	147.75	3.18	**158.79**
	local art. col	520	479	447.80	3.62	456.87
	1 art.col. + FFD	514	617	138.58	3.17	164.12
	local art. col. + FFD	**473**	617	418.34	3.45	451.26

Table 4.5: CSP - Interval on production (5%)

Problem	Init. Mode	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
real instances	1 art. col.	131	122	3.50	0.31	4.53
	1 global art. col.	125	114	3.62	0.30	4.46
	local art. col	96	84	13.13	0.22	13.82
	1 art.col. + FFD	79	98	2.17	0.04	2.65
	1 global art. col. + FFD	80	96	2.21	0.02	2.59
	local art. col. + FFD	**78**	93	2.35	0.03	2.75

Table 4.6: Multiple Widths Cutting Stock Problem

The least number of iterations is obtained with the FFD heuristic combined with the use of local artificial columns for almost all variants except the triplets instances for which it is obtained with the use of n artificial columns: in the case of the triplets instances the FFD heuristic typically provides a poor primal solution. Compared to the use of one artificial column, using n local artificial columns reduces the number of iterations in a consequent way for all variants, this reduction is due to its stabilization effect in the column generation procedure. However, in the model with interval on production the best computational time is obtained with the use of one artificial column because oracles take more time in the stabilized version. Overall, we note that the average time spent in oracles per iteration is lower when using one artificial column because subproblems are then easier to solve when their costs are well balanced.

4.3 STABILIZATION METHODS

To prevent problems of convergence issued from the instability of the dual values a lot of stabilization techniques have been developed. Some of these techniques are reviewed in this section and compared experimentally. We consider four types of stabilization methods:

(i) bounding the dual values,

(ii) smoothing the dual values,

(iii) penalizing deviation of the dual solution from a stability center.

We shall consider the stabilization effect of the use of the formulations with built-in exchanges that were presented in 2.5 in the next section.

4.3.1 THE DYNAMIC BOXSTEP METHOD

Boxstep method is used to bound the dual variables around a stability center $\hat{\pi}$. This method was introduced by Marsten in [27]: the Lagrangian dual is solved forcing π to lie in an l_∞ box around $\hat{\pi}$. The problem to solve at iteration k is then:

$$\max_{\pi \geq 0} \{ L^k(\pi) \ : \ \|\pi - \hat{\pi}\|_\infty \leq \delta \} \tag{4.5}$$

where L^k is defined in (2.19).

On completion of the Kelley's cutting plane procedure (see section 2.3.5), if the current solution strictly lies in the box, optimality is proved and the Kelley's cutting plane process terminates. Otherwise, the stability center is moved setting $\hat{\pi}$ to the value of the dual vector giving the best dual bound and the procedure is reiterated.

The LP form of (4.5) is:

$$\begin{aligned}
&\max \sum_{i=1}^{n} d_i \, \pi_i - K \, \sigma \\
&\text{s.t.} \quad x_i^q \, \pi_i - \sigma \ \leq \ 1 && \forall q \in Q \\
&\qquad\qquad \pi_i \ \leq \ \hat{\pi}_i + \delta && \forall i \\
&\qquad\qquad -\pi_i \ \leq \ -\hat{\pi}_i + \delta && \forall i \\
&\qquad\qquad \pi_i, \sigma \ \geq \ 0 && \forall i
\end{aligned}$$

whose dual is the primal view of the dynamic box step method:

$$\max \sum_{q \in Q} \lambda_q + \sum_{i=1}^{n} ((\hat{\pi}_i + \delta)\, y_i^+ + (-\hat{\pi}_i + \delta)\, y_i^-)$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{q \in Q} x_i^q\, \lambda_q + y_i^+ - y_i^- &\leq d_i && \forall i \\
\sum_{q \in Q} \lambda_q &\leq K \\
\lambda_q &\geq 0 && \forall q \in Q \\
y_i^+, y_i^- &\geq 0 && \forall i
\end{aligned}$$

Thus, the use of local artificial columns in the primal master problem, as in formulation [M1LocArtCol], acts as a Boxstep method using one-side boxes, where y_i^- is fixed to 0, $y_i^+ = \gamma_i$ and $\delta + \hat{\pi}_i = \tilde{c}_i$.

The principle of the dynamic Boxstep method is to bound dynamically these dual variables: each time we find a new dual bound $L^k(\pi) > LB$, with $LB = \max_{l=1,\ldots,k+1}\{L^k(\pi^l)\}$, we redefine our stability center to be the current dual solution π^{best} that gives rise to the best dual bound so far.

For the comparative numerical tests we use a simplified implementation of the dynamic box step method: we only consider upper bounds on π_i which we define as $\alpha\, \pi^{best}$ where α is a constant set to 1.5 and we update it at each improvement of the Lagrangian bound. Thus, in practice, we simply update the costs of local artificial columns according to the above scheme.

4.3.2 THE BUNDLE METHOD

Non-linear stabilization methods can be used to solve the dual Lagrangian problem (2.18).

Bundle methods go back to [25]. In their latest form, initiated in [26], [21] and fully described in [15, Chap. XV], it consists in approximating the dual function by adding a quadratic term (an euclidean norm) that penalizes the variation of the dual variables from a stability center. Let us adopt the dual view of section 2.3.5 to see how this penalization can be implemented in the cutting plane algorithm. Let $\hat{\pi}$ be the stability center, the dual function to solve takes the form:

$$\max_{\pi \geq 0} L^k(\pi) - \frac{1}{2t}\|\pi - \hat{\pi}\|^2 \tag{4.6}$$

The restricted master problem (4.6) is solved by the quadratic programming solver of K.C. Kiwiel [22], [23]. The initialization is done heuristically computing $\hat{\pi_i} = \tilde{c_i} \ \ \forall i$ and we start with an empty initial set of columns. The algorithm also needs an initial value for t, and this is obtained from an estimate of the optimal value of the original problem. The algorithm stops when the largest cut violation is "sufficiently small", i.e. when

$$\xi(\pi^k) = \theta^k(\pi^k) - \theta(\pi^k) \leqslant \varepsilon\,. \tag{4.7}$$

(see in [15, §II.2.2(c)] for more explanations).

4.3.3 SMOOTHING METHODS

In this section we present two smoothing methods: the first one has been developed by Neame in [30] and the second one by Wentges in [44]. The principle of smoothing methods is to take a convex combination of the current dual solution,

π^k and, either the solution obtained at the previous iteration (in the method by Neame), or the dual solution that provided the best Lagrangian bound, π^{best}, seen as a stability center (in the method by Wentges). Thus, after each solution of the restricted master, the dual values, before being sent to the subproblem, are smoothed.

The principle of the **Neame's procedure** is to solve the Lagrangian subproblem with the dual vector

$$\pi^k = (1-\alpha)\pi^{k-1} + \alpha\pi^{RM} \tag{4.8}$$

where π^{RM} is the optimal dual vector given by the last restricted master solution, and π^{k-1} is the previous dual vector used at the last iteration, $0 < \alpha < 1$ is a constant. When the column generated has a negative reduced cost, it is added to the restricted master and one goes on to the following iteration, else, if the dual vector is too distant from π^{RM} this vector is recomputed by giving more weight to π^{RM}, increasing the parameter α. If none of these situations occurs the subproblem is solved using the dual vector π^{RM}. It has then the effect of clearing the memory of the preceding vectors. The algorithm stops only if, in this case, the column generated has a non-negative reduced cost, then the master solution is optimal.

Algorithm 1 (Neame's algorithm)

STEP 0. *Select an initial set of columns. Choose $\varepsilon > 0$, $0 < \alpha < 2$, compute π^{RM} by solving the restricted master problem and set $\pi^0 = \pi^{RM}$.*

STEP 1.A $i = 0$.

STEP 1.B *Compute a dual optimal solution π^{RM} of the restricted master problem.*

STEP 1.C *Compute $\pi^{i+1} = (1-\alpha)\pi^i + \alpha\pi^{RM}$. Solve the Lagrangian subproblem using π^{i+1} to optimality. If the resulting column has a negative reduced cost let $\pi^0 = \pi^{i+1}$, add the column and goto STEP 1-a)*

Else, if $||\pi^{i+1} - \pi^{RM}|| > \varepsilon$*, and* $i < i_{max}$*, then* $i = i + 1$ *and goto STEP 1-c)*
Else solve the Lagrangian subproblem with π^{RM}*, if the column has a non-negative reduced cost the current solution is optimal (stopping criteria 1).*
Else $\pi^0 = \pi^{RM}$*, add new column and goto STEP 1-a).*

For our numerical tests, we use a simplified implementation where $i_{max} = 1$ and $\alpha = 0.7$ is constant.

In the **Wentges' procedure** the subproblem is solved taking

$$\pi^k = (1 - \alpha)\pi^{best} + \alpha\pi^{RM} \tag{4.9}$$

where π^{best} is the dual vector giving the best Lagrangian bound value, and α is decreased at each iteration, its depends on the iteration and the number of times the Lagrangian bound has been improved (the more one advances in the algorithm and one improves the dual bound and the more one gives weight to the π^{best}). In our simplified implementation α remains constant and is set to value 0.7. Furthermore, if the oracle does not give a negative reduced cost column for the smoothed dual solution π^k, we reset $\pi^k = \pi^{RM}$ and call again the oracle as in the Neame method.

4.3.4 COMPARATIVE TESTS

We present results that are averages on the instances classes presented in section 4.1. For the dynamic box step method, we use the formulation with n artificial columns. For the Bundle method, the master is initialized with one artificial column.

	Method	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
real inst.	1 art. col.	42	43	0.48	0.03	0.51
	1 art. col. + wentges	44	43	0.68	0.05	0.73
	1 art. col. + neame	48	45	0.68	0.05	0.73
	n art. col.	25	25	3.49	0.03	3.52
	n art. col. + wentges	25	25	3.72	0.01	3.73
	n art. col. + neame	**23**	22	2.33	0.02	2.35
	dynamicBoxStep	25	25	3.61	0.03	3.64
	Bundle	34	26	21.03	0.03	21.06
rand. inst. 50 items	1 art. col.	256	257	3.60	1.08	4.68
	1 art. col. + wentges	230	231	3.45	1.04	4.49
	1 art. col. + neame	230	231	3.34	1.06	4.40
	n art. col.	130	128	2.31	0.22	2.53
	n art. col. + wentges	130	128	2.78	0.22	3.00
	n art. col. + neame	**125**	123	2.01	0.32	2.33
	dynamicBoxStep	132	128	2.64	0.22	2.86
	Bundle	163	114	32.74	0.39	33.13
rand. inst. 150 items	1 art. col.	727	727	40.16	23.00	63.16
	1 art. col. + wentges	643	644	22.08	18.27	40.35
	1 art. col. + neame	644	646	21.44	18.26	39.70
	n art. col.	476	476	34.25	2.55	36.80
	n art. col. + wentges	474	473	30.62	2.36	32.98
	n art. col. + neame	**443**	441	19.27	3.14	22.41
	dynamicBoxStep	481	474	35.40	2.51	37.91
	Bundle	668	388	671.52	14.00	685.52

Table 4.7: CSP Stab

	Method	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
250 items	1 art. col.	273	273	0.44	0.58	1.02
	1 art. col. + wentges	240	241	0.36	0.74	1.10
	1 art. col. + neame	226	227	0.33	0.70	1.03
	n art. col.	110	108	0.34	0.21	0.55
	n art. col. + wentges	110	108	0.33	0.21	**0.54**
	n art. col. + neame	**103**	101	0.30	0.33	0.63
	dynamicBoxStep	110	108	0.33	0.22	0.55
	Bundle	112	106	0.56	1.05	1.61
500 items	1 art. col.	269	269	0.35	0.77	1.12
	1 art. col. + wentges	243	243	0.30	0.86	1.16
	1 art. col. + neame	226	226	0.29	0.77	1.06
	n art. col.	117	115	0.43	0.24	0.67
	n art. col. + wentges	117	115	0.43	0.24	0.67
	n art. col. + neame	108	106	0.37	0.36	0.73
	dynamicBoxStep	117	115	0.43	0.23	**0.66**
	Bundle	**99**	99	0.29	0.99	1.28
1000 items	1 art. col.	258	258	0.33	0.90	1.23
	1 art. col. + wentges	235	235	0.26	0.96	1.22
	1 art. col. + neame	217	217	0.24	0.79	1.03
	n art. col.	121	119	0.49	0.27	0.76
	n art. col. + wentges	121	119	0.48	0.26	**0.74**
	n art. col. + neame	109	107	0.42	0.37	0.79
	dynamicBoxStep	121	119	0.49	0.25	**0.74**
	Bundle	**93**	94	0.15	0.94	1.09

Table 4.8: Bin-Packing aggreg - Stab

	Method	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
120 items	1 art. col.	340	340	5.75	0.97	6.72
	1 art. col. + wentges	326	326	5.73	1.29	7.02
	1 art. col. + neame	325	325	8.75	1.23	9.98
	n art. col.	123	121	2.56	0.28	**2.84**
	n art. col. + wentges	123	121	2.59	0.33	2.92
	n art. col. + neame	135	133	3.22	0.50	3.72
	dynamicBoxStep	123	121	2.58	0.29	2.87
	Bundle	**80**	81	2.36	0.92	3.28
249 items	1 art. col.	651	651	53.39	3.66	57.05
	1 art. col. + wentges	592	592	38.04	4.03	42.07
	1 art. col. + neame	562	562	65.42	3.80	69.22
	n art. col.	191	189	10.52	0.75	11.27
	n art. col. + wentges	191	189	10.51	0.73	11.24
	n art. col. + neame	203	201	11.83	1.17	13.00
	dynamicBoxStep	191	189	10.49	0.73	11.22
	Bundle	**138**	138	7.99	1.54	**9.53**
501 items	1 art. col.	956	956	479.73	8.71	488.44
	1 art. col. + wentges	886	886	302.57	9.71	312.28
	1 art. col. + neame	788	788	397.97	7.84	405.81
	n art. col.	253	251	27.44	1.30	28.74
	n art. col. + wentges	253	251	27.22	1.30	28.52
	n art. col. + neame	264	262	27.38	2.08	29.46
	dynamicBoxStep	253	251	27.50	1.31	28.81
	Bundle	**191**	190	24.32	2.67	**26.99**

Table 4.9: Bin-Packing aggreg - Stab - triplets instances

	Method	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
real inst.	1 art. col.	43	43	2.79	0.04	**2.83**
	1 art. col. + wentges	41	41	4.25	0.06	4.31
	1 art. col. + neame	44	44	4.53	0.08	4.61
	n art. col.	30	26	12.48	0.03	12.51
	n art. col. + wentges	29	26	10.96	0.05	11.01
	n art. col. + neame	**27**	23	12.01	0.04	12.05
	dynamicBoxStep	95	46	132.72	0.13	132.85
	Bundle	62	33	17.17	0.78	17.95
rand. inst. 50 items	1 art. col.	140	140	7.64	0.28	**7.92**
	1 art. col. + wentges	**135**	134	8.76	0.45	9.21
	1 art. col. + neame	138	137	8.57	0.44	9.01
	n art. col.	139	121	30.32	0.32	30.64
	n art. col. + wentges	139	121	40.22	0.44	40.66
	n art. col. + neame	136	118	36.66	0.53	37.19
	dynamicBoxStep	141	140	8.28	0.25	9.56
	Bundle	193	80	52.14	1.43	53.57
rand. inst. 150 items	1 art. col.	521	518	147.85	3.05	**150.90**
	1 art. col. + wentges	**505**	502	180.02	4.87	184.89
	1 art. col. + neame	522	519	164.79	5.21	170.00
	n art. col.	545	499	567.51	3.80	565.90
	n art. col. + wentges	539	493	722.68	5.54	728.22
	n art. col. + neame	537	492	661.14	5.95	667.09
	dynamicBoxStep	491	466	227.13	1.77	228.90
	Bundle	550	241	1026.27	11.41	1037.68

Table 4.10: CSP tolerance on production Stab

Method	ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
1 art. col.	395	1058	7.46	4.62	12.08
1 art. col. + wentges	345	906	5.97	4.07	10.14
1 art. col. + neame	336	893	6.23	3.86	10.09
n art. col.	186	504	5.89	0.67	6.56
n art. col. + wentges	180	485	5.29	0.74	6.03
n art. col. + neame	**165**	433	4.99	0.82	5.81
dynamicBoxStep	270	567	11.48	1.12	12.6
Bundle	377	436	14.71	5.26	19.97

Table 4.11: CSP Multiple Widths Stab

From a general point of view, the use of local artificial columns has a real stabilization effect on all instances, except for the variant with tolerance on production.
The smoothing methods (Wentges and Neame methods) have broadly the same effect, however Neame's method seems to be a little more effective, and it is accentuated when it is applied to the version with local artificial columns, compared to the version with a single artificial column. However for the triplets instances of the Bin Packing problem, smoothing methods have no stabilization effect when combined with the use of local artificial columns.

The dynamic Boxstep method has no effect because we start with an optimal dual solution, and so the stability center does not move. The Bundle method must be compared to the single artificial column version. It has a stabilization effect for all the variants, the number of iterations decreased, however the time spent in the subproblem increased, except for the Bin Packing where the bundle method takes less time and less iterations compared to all the other stabilization methods.

For the variant with tolerance on production none of the stabilization methods have a real impact on the column generation procedure. The number of iterations is roughly the same while the time increases. The resolution of this variant is less unstable because there is less degeneracy.

4.4 FORMULATIONS WITH EXCHANGES

The master formulations with exchanges built-in presented in 2.5 are stabilization methods, as they amount to add dual cuts. We have compared the following formulations:

- **M1** that refers to the standard master formulation. We initialize it with a global artificial column.

- The formulation **AgregCovM1** that correspond to the aggregation of the covering constraints.

- For the exchange flow formulation we tested two versions: the first one is **DirectExchFlowM1** that corresponds to the simple exchanges and the second **ExchFlowM1** refers to whole feasible exchanges between two piece types.

- We experiment**RestrExchVectM1** where exchange vectors are restricted to those where one item is replaced by a set of smaller items. At each iteration of the column generation procedure we solve two subproblems, the column generation subproblem and a second pricing subproblem that allows to generate a restricted exchange vector (with $|r| = 1$).

- The method of Carvalho where cuts are added a priori to the problem by generating the corresponding columns before starting the column generation process has been tested. For this, two types of cuts are used, referred as **CarvalhoCuts**:

$$\pi_{i+1} \leq \pi_i \qquad \text{for } i = 1, \ldots n-1 \tag{4.10}$$

$$\pi_j + \pi_k \leq \pi_i \quad \begin{cases} \text{for } i = 1, \ldots n-2, \\ \text{for the first identified pair of items (j,k)} \\ \text{such that } i < j < k \text{ and } w_j + w_k \leq w_i \end{cases} \tag{4.11}$$

The **Simple CarvalhoCuts** refers to the use of only the first type of cuts. For the second type of cuts we generate only one cut for all i.

4.4.1 COMPARATIVE TESTS

	Method	DB	ITER	Col	t_{Oracle}	t_{Master}	$t_{Or.+Mast.}$
	M1	81.22	42	43	0.48	0.03	0.51
real	AgregCovM1	81.22	35	35	0.26	0.02	0.28
	DirectExchFlowM1	81.22	40	40	0.61	0.04	0.65
	Simple CarvalhoCuts	81.22	53	55	0.11	0.04	**0.15**
inst.	ExchFlowM1	81.22	**35**	36	0.55	0.03	0.58
	RestrExchVectM1	81.22	41	80	3.47	0.04	3.51
	CarvalhoCuts	81.22	46	121	0.50	0.05	0.55
	M1	1381.10	256	257	3.60	1.08	**4.68**
rand.	AgregCovM1	1381.10	207	208	5.73	1.23	6.96
50	DirectExchFlowM1	1381.10	211	212	5.60	0.75	6.35
items	Simple CarvalhoCuts	1381.10	267	270	5.33	1.18	6.51
	ExchFlowM1	1381.10	**188**	189	5.27	0.79	6.06
	RestrExchVectM1	1381.10	218	404	108.25	1.18	109.43
	CarvalhoCuts	1381.10	223	1077	4.28	1.83	6.11
				cuts generation = 1.50			
	M1	2558.35	727	727	40.16	23.00	63.16
rand.	AgregCovM1	2558.35	660	661	55.81	119.13	174.94
150	DirectExchFlowM1	2558.35	700	700	36.51	21.31	**57.82**
items	Simple CarvalhoCuts	2558.35	779	781	52.57	23.46	76.03
	ExchFlowM1	2558.35	**557**	557	32.45	25.56	58.01
	RestrExchVectM1				TOO LONG		
	CarvalhoCuts	2558.35	681	10584	49.72	85.57	135.29

Table 4.12: CSP Exchanges init. with one art. col

The use of alternative master reformulations allows to reduce the number of iterations on almost all instances. The formulation ExchFlowM1 is the most effective on the standard cutting stock problem. However, on the triplets instances, it is less effective because the item widths are roughly the same and so

	Method	DB	ITER	Col	t_{Oracle}	t_{Master}	$t_{Or.+Mast.}$
	M1	101.6	273	273	0.44	0.58	**1.02**
250	AgregCovM1	101.6	226	226	1.10	1.24	**2.34**
	DirectExchFlowM1	101.6	237	237	0.99	0.66	1.65
	Simple CarvalhoCuts	101.6	295	298	0.90	0.76	1.66
items	ExchFlowM1	101.6	**201**	201	0.69	0.67	1.36
	RestrExchVectM1	101.6	207	415	40.37	0.78	41.15
	CarvalhoCuts	101.6	221	1850	0.46	1.10	1.56
	M1	201.2	269	269	0.35	0.77	**1.12**
500	AgregCovM1	201.2	236	236	1.43	1.54	**2.97**
	DirectExchFlowM1	201.2	251	251	1.13	0.81	1.94
	Simple CarvalhoCuts	201.2	277	279	0.74	0.75	1.49
items	ExchFlowM1	201.2	222	222	0.91	0.90	1.81
	RestrExchVectM1	201.2	210	420	30.87	1.02	31.89
	CarvalhoCuts	201.2	**208**	1999	0.38	1.43	1.81
	M1	400.5	258	258	0.33	0.90	**1.23**
1000	AgregCovM1	400.5	240	240	1.49	1.72	**3.21**
	DirectExchFlowM1	400.5	238	238	1.17	0.91	2.08
	Simple CarvalhoCuts	400.5	309	311	1.04	1.07	2.11
items	ExchFlowM1	400.5	**198**	198	0.67	0.97	1.64
	RestrExchVectM1	400.5	206	413	26.73	1.20	27.93
	CarvalhoCuts	400.5	234	2029	0.52	1.82	2.34

Table 4.13: Bin-Packing aggreg - Exchanges init. with one art. col.

	Method	DB	ITER	Col	t_{Oracle}	t_{Master}	$t_{Or.+Mast.}$
	M1	40	340	340	5.75	0.97	6.72
120	AgregCovM1	40	**231**	231	27.62	1.59	29.21
	DirectExchFlowM1	40	277	277	42.59	0.76	43.35
	Simple CarvalhoCuts	40	374	376	27.71	0.98	28.69
items	ExchFlowM1	40	277	277	42.66	0.80	43.46
	RestrExchVectM1	40	290	537	268.88	1.22	8.93
	M1	83	651	651	53.39	3.66	57.05
249	AgregCovM1	83	**366**	366	232.72	7.20	239.92
	DirectExchFlowM1	83	429	430	441.41	2.13	443.54
	Simple CarvalhoCuts	83	561	563	309.68	2.75	312.43
items	ExchFlowM1	83	429	430	440.01	2.11	442.12
	RestrExchVectM1	83	465	887	1247.81	6.18	1253.99
	M1	167	956	956	479.73	8.71	488.44
501	AgregCovM1	167	**469**	465	671.48	22.26	693.74
	DirectExchFlowM1	167	623	623	2154.48	4.64	2159.12
	Simple CarvalhoCuts	167	735	737	2639.65	7.02	2646.67
items	ExchFlowM1	167	622	622	2096.58	4.66	2101.24
	RestrExchVectM1	TOO LONG					

Table 4.14: Exchanges init. with one art. col. Bin-Packing triplets aggreg

	Method	DB	ITER	Col	t_{Oracle}	t_{Master}	$t_{Or.+Mast.}$
	M1	1893.03	43	43	2.54	0.04	2.58
real	DirectExchFlowM1	1893.03	39	40	2.18	0.05	**2.23**
	Simple CarvalhoCuts	1893.03	50	59	3.37	0.05	3.42
inst.	ExchFlowM1	1893.03	39	39	4.23	0.05	4.28
	RestrExchVectM1	1893.03	**36**	62	5.94	0.05	5.99
	CarvalhoCuts	1893.03	42	123	2.53	0.06	2.59
	M1	3499.79	140	140	7.71	0.28	7.99
random	DirectExchFlowM1	3499.79	133	133	7.50	0.33	7.83
	Simple CarvalhoCuts	3499.79	159	192	14.69	0.29	14.98
instances	ExchFlowM1	3499.79	**130**	130	7.48	0.33	**7.81**
50 items	RestrExchVectM1	3499.79	144	212	135.37	0.31	135.68
	CarvalhoCuts	3499.79	157	1045	15.18	0.35	15.53
				cuts generation = 2.50			
	M1	15778.3	521	518	147.85	3.05	150.90
random	DirectExchFlowM1	15778.3	506	503	161.65	5.10	166.75
instances	ExchFlowM1	15778.3	**491**	488	162.83	5.45	168.28
150 items	RestrExchVectM1			TOO LONG			
	CarvalhoCuts	15778.2	582	10675	342.23	5.83	348.06

Table 4.15: CSP tol On Prod Exchanges init. with one art. col

Method	DB	ITER	Col	t_{Oracle}	t_{Master}	$t_{Or.+Mast.}$
M1	2048.3	395	1058	7.46	4.62	**12.08**
AgregCovM1	2048.3	421	1071	12.60	19.96	32.56
DirectExchFlowM1	2048.3	429	1075	11.16	6.50	17.66
Simple CarvalhoCuts	2048.3	425	905	12.87	3.73	16.60
ExchFlowM1	2048.3	**339**	779	9.74	6.12	15.86
CarvalhoCuts	2048.3	348	5188	10.94	8.28	19.22

Table 4.16: CSP Multiple Widths Exchanges init. with one art. col

only simple (one-to-one) exchanges can arise. We note that the subproblems are harder to solve when using the formulation AgregCovM1, perhaps because the dual values are more correlated.

The use of formulation ExchVectM1 is very expensive, because to generate exchange vectors at each iteration we have to solve a second subproblem (it is solved using Xpress because we did not develop a specific solver for this sub problem). When using Carvalho cuts, some time is spent in generating the cuts a priori before the procedure, and on the instances with a lot of items, more time is spent in solving the restricted master problems because there are lots of extra columns.
We note that in practice we obtain the same dual bounds, even using formulations that are relaxations of the standard column generation formulation.

4.5 STRATEGIES FOR COLUMN GENERATION

In the classical column generation procedure, one column is generated at each iteration by an exact oracle. Alternatively, one could use a heuristic oracle while it provides a column with negative reduced cost and call the exact oracle only when it fails. However a drawback of this method is that the dual bound must be computed with an exact subproblem solution, so we have also tested this strategy but generating an exact column solution every k iterations (in our tests we use k = 2 or 10). Our heuristic oracle is the standard greedy algorithm for the knapsack problem.

We also experimented with a different strategy aiming at diversifying the search. Our intuition being that all columns with best reduced tend to concern the same subset of items. Thus, we implemented the following scheme:

(i) Solve the subproblem exactly and compute how many time this column can feasibly be taken in the CSP solution,
(ii) update the bound on subproblem variables as if you were taking that selected column so many time,
(iii) re-solve the subproblem after updating to obtain the next selected column,
(iv) if it has negative reduced cost, re-iterate.
In practice we impose an upper bound on the number of passes. (In our experiments we compare generating up to 3 columns and up to 10 columns per iteration.)

Other strategies could be to record all columns with negative reduced cost that are encountered in the oracle while solving the subproblem (for instance when the oracle is a dynamic programming solver or a branch-and bound). When there are many columns that could be returned, one can apply a selection criteria such as taking those with minimum negative reduced cost. Moreover, using a dynamic programming recursion would allow to generate all the columns with an optimal cost, memorizing all the paths leading to a dominant solution. While with a branch-and-bound procedure only columns that arise as incumbents at some stages can be generated. We have not implemented these strategies.

In tables 4.17 and 4.18, respectively for the standard cutting stock problem and the variant with tolerance on production, we compare these strategies on the instances presented in section 4.1. The reported results are averages.

		ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
	1 exact col.	29	30	4.28	0.03	5.19
real	3 col/it.	10	41	3.60	0.01	3.64
inst.	10 col/it.	**7**	74	4.01	0.00	4.06
	1 col. heur.	62	62	0.61	0.09	**1.37**
	1 col.Heur-exact every 2it.	45	45	1.63	0.05	2.23
	1 col.Heur-exact every 10it.	61	61	0.72	0.07	1.47
	1 exact col.	146	146	6.88	0.30	10.93
50	3 col/it.	42	165	6.96	0.80	11.17
items	10 col/it.	**21**	200	9.40	0.50	13.59
	1 col. heur.	251	251	5.28	0.50	9.95
	1 col.Heur-exact every 2it.	201	201	5.55	0.39	**9.56**
	1 col.Heur-exact every 10it.	240	240	5.53	0.51	9.94
150	1 exact col.	492	496	353.45	2.93	380.86
items	1 col.Heur-exact every 10it.	834	837	272.76	5.36	314.72

Table 4.17: CSP Several columns generated per iteration

		ITER	Col	t_{Oracle}	t_{Master}	t_{Total}
	1 col/it.	28	28	4.30	0.02	4.37
real	3 col/it.	15	38	8.33	0.01	8.40
inst.	10 col/it.	**11**	65	10.96	0.01	11.06
	1 col/it.	142	120	32.80	0.29	33.56
50	3 col/it.	65	134	54.20	0.18	54.73
items	10 col/it.	**48**	222	93.11	0.17	94.39

Table 4.18: CSP with tolerance on prod. - Several columns generated per iteration

The generation of several columns per iteration allows to reduce considerably the number of column generation iterations, however lots of these columns are not used. At the opposite the generation of heuristic columns increases the number of iterations because the dual bound is updated only when this heuristic column has a positive reduced cost and when an exact column is generated. That is why we have tested the generation of an exact column after a fixed number of iterations. A heuristic solver allows to reduce the time spent in oracles (time in oracles per iteration decrease), however, this does not translate into a reduction in the total time as the number of iterations is much larger.

4.6 PRIMAL HEURISTICS

Integer primal solutions can be derived either a priori (for instance using the FFD heuristic described in Section 4.2.1) or using the column generation framework. A classical column generation based heuristic is to solve to integrality the master program restricted to a subset of columns (by branch-and-bound). This can potentially be done at any time during branch-and-price. However we just report experimental results where this exact solution of the restricted master is called at the end of the column generation procedure at the root node of the branch-and-price tree. It is to be observed that the restricted set of columns can be insufficient to give rise to an integer feasible solution. Below we describe 2 column generation based heuristics using dynamic column generation: a greedy procedure and a rounding procedure. Then, we compare the quality of the primal solution obtained with these procedures on random instances.

4.6.1 GREEDY ALGORITHM

In the greedy heuristic, we iteratively generated a column by solving the subproblem with heuristic dual prices. The generated column is taken in the solution as

many time as feasible and the process reiterates until the residual master problem becomes trivial (all orders demands are satisfied). This procedure can be called at the outset of the algorithm using dual price set a priori (as defined in (4.3) for the standard problem and in (4.4) for the variant with interval on production). Alternatively, the greedy procedure can be called at any time using the current master dual solution. In our numerical comparison we test the greedy procedure with a priori dual value (this procedure is denoted greedy1) and with the dual solution obtained at the end of the column generation procedure at the root node of the b-a-p tree (this procedure is denoted greedy2). The cumulative use of these two procedures will be denoted greedy.

4.6.2 ROUNDING HEURISTIC

The rounding heuristic consists in deriving an integer solution from the linear solution of the master problem. It can be seen as doing a heuristic dive into a branch-and-price tree. However the underlying branching scheme is not necessarily the one that would be used for solving the problem exactly: the aim is to get quickly to an integer solution and we do not have to worry about backtracking. Again, it can be used at any time, but in our tests we used it only at the end of the column generation procedure at the root node of the b-a-p tree. To be specific, the last LP solution is rounded down and the column q which have the largest value λ_q is iteratively selected and taken in the partial solution by rounding it up. After each round-up we apply a column generation procedure to re-optimize the linear master problem associated to the residual problem obtained by removing the fixed columns and updating the right hand side of the master constraints.

4.6.3 COMPARATIVE TESTS

These four methods have been tested on instances described in section 4.1. In the following tables, we mention the instances tested, the method used: **FFD** refers

to the first fit decreasing algorithm, **greedy1**, **greedy2** and **greedy** to the greedy heuristics, **rounding** to the rounding heuristic and **RMIP** refers to the solution of the integer restricted master problem. However the RMIP used alone does not allow to find an integer solution in most cases, so we test it with the FFD algorithm in **FFD + RMIP**. The tables compare then the primal and dual bounds and timers in seconds. The number given between brackets alongside the heuristic name corresponds to the number of instances for which an integer solution was found. Then, the average primal bound is that obtained over the instances for which we found a primal bound.

	Method	PB	DB	ITER	Col	t_{Oracle}	t_{Master}	t_{Heur}	$t_{Or.+Mast.}$	t_{Total}
	FFD	83.56	81.22	34	50	0.09	0.03	0.11	0.11	1.27
real	greedy	81.67	81.22	21	52	2.63	0.02	4.75	2.65	7.57
inst.	rounding	**81.22**	81.22	123	122	2.03	0.09	4.85	2.12	6.88
	RMIP	$+\infty$	81.22	30	28	4.42	0.03		4.45	5.54
	FFD + RMIP	83.56	81.22	16	39	0.73	0.01	0.11	0.74	1.63
	FFD	1405.90	1380.8	186	235	5.36	0.61	2.49	5.97	18.34
50	greedy	1393.95	1380.8	172	248	8.65	0.55	16.91	9.00	46.60
	greedy1	1447.40	1380.8	105	170	4.19	0.21	2.38	6.59	7.34
	greedy2	1397.60	1380.8	142	177	6.07	0.27	6.05	6.34	11.47
items	rounding	**1381.15**	1380.8	345	338	8.53	0.82	12.10	9.35	30.00
	FFD	2574.1	2557.65	539	639	239.71	16.86	10.88	256.57	333.65
150	greedy	**2626.3**	2557.65	622	763	378.16	18.30	318.65	396.46	1040.72
items	rounding	**2626.3**	2557.65	622	763	433.86	18.68	326.79	452.54	1109.98

Table 4.19: CSP Primal Heuristics

	Method	PB	DB	ITER	Col	t_{Oracle}	t_{Master}	t_{Heur}	$t_{Or.+Mast.}$	t_{Total}
	FFD	103.10	101.6	50	111	0.12	0.12	0.54	0.24	4.60
250	greedy	110.55	101.6	62	174	0.31	0.15	6.13	0.46	17.36
items	rounding	**101.60**	101.6	235	211	0.53	0.45	2.52	0.98	4.04
	FFD + RMIP	103.10	101.6	50	111	0.13	0.13	0.55	0.26	4.90
	FFD	203.9	201.2	39	118	0.05	0.09	0.80	0.11	4.51
500	greedy	211.4	201.2	53	188	0.25	0.13	7.84	0.38	21.15
items	rounding	**201.2**	201.2	259	235	0.55	0.50	3.22	1.05	4.53
	FFD + RMIP	203.9	201.2	39	118	0.05	0.09	0.82	0.11	4.82
	FFD	405.40	400.55	37	130	0.06	0.09	1.16	0.15	4.92
1000	greedy	414.95	400.55	51	194	0.25	0.12	8.69	0.37	23.35
items	rounding	**400.55**	400.55	261	237	0.56	0.49	3.04	1.05	4.58
	FFD + RMIP	405.40	400.55	37	130	0.05	0.09	1.19	0.11	5.14

Table 4.20: Bin Packing Primal Heuristics

We can conclude that in all cases the best primal bounds are obtained with the rounding heuristic, furthermore in almost all instances it allows to find the optimal

	Method	PB	DB	ITER	Col	t_{Oracle}	t_{Master}	t_{Heur}	$t_{Or.+Mast.}$	t_{Total}
	FFD	45.8	40	179	224	1.44	0.53	0.40	1.97	13.26
120	greedy	41.0	40	122	191	2.50	0.34	3.99	2.84	20.06
items	rounding	**40.6**	40	219	200	1.80	0.48	6.92	2.28	16.18
	RMIP	$+\infty$	40	125	123	1.01	0.34		1.35	9.69
	FFD + RMIP	45.8	40	190	235	2.01	0.61	0.41	2.62	15.12
	FFD	95.0	83	593	617	16.49	3.27	1.26	19.76	71.77
249	greedy	84.0	83	156	297	9.83	0.73	12.89	10.56	65.44
items	rounding	**83.9**	83	372	334	8.91	1.24	24.64	10.15	47.98
	FFD + RMIP	95.0	83	339	431	8.26	1.84	1.27	10.10	44.66
	FFD	190.0	167	298	467	126.79	1.93	1.95	128.72	142.01
501	greedy	168.2	167	214	473	25.79	1.52	34.37	27.31	177.58
items	rounding	**167.8**	167	490	438	30.35	2.26	32.61	32.61	103.40

Table 4.21: Bin Packing triplets Primal Heuristics

	Method	PB	DB	ITER	Col	t_{Oracle}	t_{Master}	t_{Heur}	$t_{Or.+Mast.}$	t_{Total}
	FFD	94308.7	1893.03	38	56	5.44	0.04	0.11	5.48	6.84
real	greedy	42806.2	1893.03	18	50	13.23	0.02	7.24	13.25	18.97
inst.	rounding	**2033.8**	1893.03	63	61	11.38	0.06	7.71	11.44	14.20
	RMIP (7)	3017.1	1893.03	43	43	2.79	0.02		2.81	3.89
	FFD + RMIP	2396.6	1893.03	38	56	3.34	0.03	0.05	3.37	4.37
	FFD	251776	3499.85	142	193	8.62	0.28	2.58	8.90	19.91
50	greedy	176285	3499.87	111	180	13.58	0.24	36.74	13.82	77.04
items	rounding	**3544.4**	3499.84	179	162	13.91	0.33	10.41	14.24	27.22
	FFD + RMIP	11204.9	3499.79	143	194	4.79	0.16		4.95	252.11
	FFD	182514.1	15778.2	524	627	934.97	3.14	11.08	938.11	1021.8
150	greedy (15)	429525.1	15778.2	454	599	887.21	1.99	217.91	888.20	1250.22
items	rounding	**15843.8**	15778.2	652	609	1424.83	3.82	241.68	1428.65	1569.95

Table 4.22: CSP with tolerance on production - Primal Heuristics

1+2 Method	PB	DB	ITER	Col	t_{Oracle}	t_{Master}	t_{Heur}	$t_{Or.+Mast.}$	t_{Total}
FFD	2237.6	2048.25	167	562	6.10	0.67	3.31	6.77	37.74
greedy (3)	2226.7	2048.25	142	559	11.22	0.58	23.52	12.10	79.33
rounding	**2048.3**	2048.25	256	571	8.39	0.80	12.63	9.19	24.84
FFD + RMIP	2237.6	2048.25	167	562	6.15	0.67		6.82	39.05

Table 4.23: CSP Multiple Widths - Primal Heuristics

solution. The first fit decreasing (FFD) algorithm allows to obtain a valid upper bound on all instances, but it can be far from the dual bound, especially on the variant with interval on production. For this variant solving the restricted master integer problem (RMIP) exactly in addition to this heuristic reduces considerably its value. We note that the set of columns generated at the root node is in general not sufficient to obtain an integer solution (the RMIP can not be solved). Greedy2 gives rise to better primal bounds than greedy1, but the combination of greedy1 and greedy2 seems to be even better.

From a computational time point of view the FFD heuristic is fastest (except on the triplets instances), indeed the columns are generated heuristically so it does not require the use of solvers for the master and the oracle. The greedy, rounding and RMIP are generic algorithms so they could be used on others applications, while the FFD heuristic is specific to cutting and bin packing problems, however it could be improved for some variants.

4.7 BRANCHING

The choice of the branching scheme is essential for the efficiency of the branch-and-price procedure. The aim of branching is twofold: improving the dual bound and enforcing integrality. These two objectives can be contradictory. As the dual bound for the global problem is the worse bound over all active nodes, it is important to use a balanced branching scheme that constraints equally all descendant nodes: i.e. the subset $\hat{Q}$ defining the partition must not be too restrictive, because otherwise, the branch corresponding to the restrictive branching constraint will probably lead to an improved bound, while the other branch will barely be constrained leading to a subproblem with almost the same bound as the parent node.

In this section, we study and compare the branching rules presented in section 2.3.7. We first test each branching rule separately, our aim is to compare the dual bounds improvements that can be obtained from that rule alone. We also see whether some rules are more likely to quickly give rise to primal integer solutions. Based on these experiments, we try to combine these independent branching rules. Priorities on the variables allows to define the order of generation of the branching constraints and can influence the balance of the tree. They are detailed below. The tree search strategies are best bound first or depth first. The best bound strategy consists in giving priority to the improvement of the dual bound, the next node to treat being the node associated to the best dual bond. The depth first strategy on the other hand aims at quickly finding primal feasible solutions. As the dual bounds at the root node are optimal for the standard cutting stock problem, the depth first strategy seems to be more appropriate, so we use it for our numerical tests.

4.7.1 NUMERICAL TESTS

For the numerical tests on branching we used smaller instances because of the computational time required. The tests have been done on 5 classes of random instances with 20 items, whose average demand is 50, and the wide roll width is 1000. The classes are characterized by the items widths which were randomly generated in the following intervals:

- class 1 (c1): $w_i \in [0, 7500]$
- class 2 (c2): $w_i \in [0, 5000]$
- class 3 (c3): $w_i \in [0, 2500]$
- class 4 (c4): $w_i \in [500, 5000]$
- class 5 (c5): $w_i \in [500, 2500]$

In tables 4.24 and 4.25, we compare the two following rules:

$$\text{[br1]} \qquad \sum_{q:x_{ij}=1} \lambda_q \in I\!N$$

$$\text{[br2]} \qquad \sum_{q:x_i>0} \lambda_q \in I\!N$$

The highest priority was given to the variable with the largest weight $w_i\ m_{ij}$ for [br1], and w_i for [br2]. The last branching rule defined in section 2.3.7 is noted [br3]:

$$\text{[br3]} \qquad \sum_{q:y_i=y'_i=1} \lambda_q \in I\!N$$

It consists in branching on the number of columns that involve two items i and l. [br3] has proved to be too restrictive to be interesting (as it can be seen on the first class tests). We also test the combination of the rules 1 and 2 on the classes for which the use of a single rule was not sufficient to obtain the integer optimal solution: we apply [br1] while a branching constraint is found to separate the current fractional solution and we use [br2] otherwise.

For the numerical experiments we have limited the number of nodes treated to 1000, so the procedure stops:

- (i) when the maximum number of nodes to treat is reached,
- (ii) when the used branching rule becomes insufficient to cut the current fractional solution,
- (iii) or when the integer optimal solution has been found.

The master is initialized with local artificial columns and we use the first fit decreasing algorithm to start we an initial integer solution.

Table 4.24 present average results over 5 random instances for the standard cutting stock problem while table 4.25 reports the average results over all classes for the variant with tolerance on production. The columns give:

- Rule = The branching rule tested.
- Nodes = number of node processed during the branch-and-price procedure.
- RtDB = dual bound at the root node.
- RtInc = incumbent at the root node.
- DB = best dual bound at the end of the branch-and-price procedure.
- Inc = best incumbent at the end of the branch-and-price procedure.
- Mast = number of master LP that have been solved.
- Sp = number of subproblems that have been solved.
- TSp = total time spent at solve the subproblems, in seconds.
- TMast = total time spent at solving restricted master problems, in seconds.
- Total = total time in seconds.

The number between brackets in the column Rule is the number of optimal solutions over 5 instances. For the variant with tolerance on production, the number of instances solved to optimality with a single branching rule was under 10% for each branching rule.

The rule [br1] was sufficient to find an optimal integer solution in almost all instances of the standard cutting stock problem. Moreover, even when the use of [br2] allows to find the optimal solution, the number of nodes generated is much more important than using [br1] and the computational times per node is larger. Optimal integer solutions are obtained for all instances on the standard cutting stock problem using first [br1] and then [br2].

For the variant with interval on production the use of a single branching rule is insufficient to cut all fractional solutions. We must use a combination of

Rule		Nodes	RtDb	RtInc	DB	Inc	Mast	Sp	Tsp	TMast	Total
c1	[br1] (5)	14	374.8	378.6	374.8	374.8	46	17	1.62	0.03	2.27
c1	[br2] (5)	6	374.8	378.6	374.8	374.8	33	18	8.36	0.02	9.39
c1	[br3] (3)	432	374.8	378.6	376	374.8	836	223	215.67	1.84	304.16
c2	[br1] (5)	106	262	266.4	262	262.0	285	62	14.87	0.25	18.39
c2	[br2] (4)	409	262	266.4	262	262.4	797	180	620.17	2.29	687.56
c3	[br1] (4)	467	122.8	123.4	122.8	123.0	777	111	44.54	1.38	69.65
c3	[br2] (3)	600	122.8	123.4	122.8	123.4	1043	124	199.51	1.92	247.40
c3	[br1] + [br2] (5)	472	122.8	123.4	122.8	122.8	1137	150	165.97	2.74	242.91
c4	[br1] (4)	222	247.2	253	247.2	247.4	494	73	68.53	0.51	78.09
c4	[br2] (2)	804	247.2	253	247.2	251.2	1381	145	114.72	2.73	181.81
c4	[br1] + [br2] (5)	191	247.2	253	247.2	247.2	440	76	76.75	0.58	91.11
c5	[br1] (4)	466	141.8	145.2	141.8	142.0	1099	137	139.43	1.28	163.86
c5	[br2] (0)	1000	141.8	145.2	141.8	145.2	1880	213	546.37	19.56	663.52
c5	[br1] + [br2] (5)	1143	141.8	145.2	141.8	141.8	2356	144	181.79	6.53	388.47

Table 4.24: Branching rules

Rule	Nodes	RtDb	RtInc	DB	Inc	Mast	Sp	Tsp	TMast	Total
[br1]	802	36231.3	88488	36231.3	40612.2	1236	554	6177.68	2.47	6328.12
[br2]	809	36231.3	88488	36231.3	47792.1	1423	489	2953.16	3.21	3066.22

Table 4.25: Branching rules - Tol. On Prod.

these rules. The computational experiments show that branching on the binary components ([br1]) gives rise to a better incumbent solution thought the trees sizes are equivalent. However the computational time spent in the subproblems is twice as large.

5

THE INDUSTRIAL CUTTING STOCK PROBLEM

Industrial Cutting Stock Problem (ICSP) combine all the difficulties of the variants discussed in Chapter 1: tolerance on production, multiple stock pieces, bi-criteria optimization (minimizing waste and the number of different patterns used). Moreover, there are typically further technical restrictions, some of which concern the definition of a feasible cutting pattern (and hence must be taken into account in the column generation subproblem), while others are global constraint (such as sequencing issue) that must be formulated in the master program.

Among the technical constraints generally encountered in the paper industry we can cite:

1. An upper bound on the waste resulting from cutting. This bound translates into a minimal cut width on the cutting pattern. It can be modeled in the subproblem as (1.13).

2. A maximum number of cuts that can be made in a pattern, as the winder has typically a fixed number of knives. It gives rise to the cardinality constraint (1.15) that goes in the subproblem.

3. A maximum number of different order types in a pattern. This constraint comes from storage problems: after the cutting process the reels are put on pallets. Moreover, dealing with many types of reels cause more waste of time in handling. Such restriction takes the form:

$$\sum_{i=1}^{n} y_{ik} \quad \leq \quad T \qquad k = 1, \ldots, K$$

where $y_{ik} = 1$ if some piece of type i is cut in roll k, i.e. if $x_{ik} > 0$ (such constraint goes in the subproblem).

4. In order to avoid short batch run length, a minimum number of use (N) for each pattern may be imposed. In formulation [F5] (see (1.16)), this constraint takes the form

$$z_k \quad \geq \quad N\, y_k \qquad k = 1, \ldots, K$$

While in the column generation reformulation [M5] (see (2.62)), it can be modeled in the subproblem, since we augmented it with multiplicity variable x_0: it takes the form

$$x_0 \quad \geq \quad N \qquad k = 1, \ldots, K$$

5. A minimum $\underline{M}$ and maximum $\overline{M}$ use of stock pieces. It is a global constraint that arise typically when there are multiple kind of stock pieces (this may be a way to enforce the use of some stock pieces in priority). In the compact formulation [F3] (1.8), it can be formulated as:

$$\underline{M} \quad \leq \quad \sum_{k \in S} y_k \quad \leq \quad \overline{M}$$

for some subset S of stock pieces. In the column generation reformulation, it takes the form:

$$\underline{M} \quad \leq \quad \sum_{k \in S} \sum_{K} \sum_{q \in Q^k} \lambda_q \quad \leq \quad \overline{M}$$

Lee, in [24], proposes a unified bilinear model that corresponds to the formulation [F5] with the same additional technical constraints as ours (minimal width and number of cuts) and whose objective is the waste minimization. This model allows to generate the patterns *in situ* (some details on this approach were given in Section 2.4). He starts with an initial set of patterns generated heuristically, some of them are fixed, he solves the partial linearized model whose solution allows to fix optimal patterns. After solving the linearized model, a local search is used to determine what patterns can be dropped or re-generated.

Our aim in this chapter is to study the extents to which such real-life problems can be approached with a branch-and-price algorithm that relies on a commercial mip solver for master and subproblem solutions. In this purpose we use a prototype generic branch-and-price code, *BaPCod* that is developed locally. The code automatically applies the Dantzig-Wolfe decomposition based on the original formulation and the user indications of what constraints must be dualized. This "black-box" approach frees the user from having to define the form taken by a column and its reduced cost and the form of the Lagrangian dual bound. All further modifications resulting from branching or adding cuts in the master for instance can be taken into account automatically too.

The interest of such approach is that it is quite flexible in incorporating specific technical constraints: any additional constraint need just be formulated correctly and the user must say if the associated constraint goes in the master or in the subproblem. The drawback is that computing times can be much larger than those

of a specialized branch-and-price algorithm that relies on an efficient subproblem solver. (However, when a specialized subproblem solver is available, it can replace the call of the commercial MIP solver in *BaPCod*). In practice, it appears that the size of the industrial instances that we received is within the reach of the generic code based on MIP solver. The strategies of implementations that we experimented within Chapter 4, have been built into *BaPCod* and are therefore available for our study of ICS problems.

5.1 THE CUTTING PROBLEM AT THE PAPER MILL CONDAT

We consider here a specific variant of ICSP such as encountered at the paper mill Condat (Dordogne, France). The primer objective is to minimize the waste while a second objective is to minimize the number of different cutting patterns. There is a tolerance on the production level. Technical constraints on cutting patterns are minimum cut width and maximum number of cuts (knifes) (1 and 2).

We implemented an hierarchical optimization approach. In a first model (ICPM1), we minimize waste under demand satisfaction constraints. Then, under a second model (ICPM2), we minimize the number of setups under demand satisfaction and waste bound constraint. We also consider a third model (ICPM3) to examine the different trade-off between waste minimization and setup minimization. We minimize waste under demand satisfaction constraints and a bound on the number of setups. By enumerating the discrete value of the latter, we can obtain all pareto optimal solutions.

The first model, which we call [ICSPM1], is to solve [M2] associated to the subproblem [BSP4]. The optimal integer solution corresponds to the minimal waste solution. The minimum waste value, $waste^*$ is used as an input for the

second model.

For the second model, [ICSPM2], the subproblem is formulation [LSP5] augmented with a constraint on the waste (as a single pattern on its own cannot produce a waste larger than the total authorized waste) and technical constraints on the minimum cut width of cutting patterns and maximum number of cuts:

$$\begin{aligned} W\, x_{0l}\, m_l - \sum_{i=1}^{n}\sum_{j=1}^{n_i} z_{ijl}\, m_{ijl}\, w_i &\leq waste^* \\ \sum_{i=1}^{n}\sum_{j=1}^{n_i} w_i\, m_{ij}\, x_{ij} &\geq W_{min} \\ \sum_{i=1}^{n}\sum_{j=1}^{n_i} m_{ij}\, x_{ij} &\leq C \end{aligned}$$

The master problem is the linearized version of [M5] with an additional constraint on the waste, i.e.,

$$\min \sum_{q\in Q} \lambda_q$$

$$ICSPM2 \qquad \text{s.t.} \quad \sum_{q\in Q}\sum_{l}\sum_{j} m_{ijl} z^q_{ijl}\, \lambda_q \geq \underline{d_i} \qquad i = 1,\ldots,n$$

$$\sum_{q\in Q}\sum_{l}\sum_{j} m_{ijl} z^q_{ijl}\, \lambda_q \leq \overline{d_i} \qquad i = 1,\ldots,n$$

$$\sum_{q\in Q}(W\, x^q_{0l}\, m_l - \sum_{i=1}^{n}\sum_{j=1}^{n_i} z^q_{ijl}\, m_{ijl}\, w_i)\, \lambda_q \leq waste^*$$

$$\sum_{q\in Q} \lambda_q \leq U$$

$$\lambda_q \in \{0,1\} \qquad \forall q \in Q$$

where U is a valid upper bound on the number of different cutting patterns. In practice, the value of this bound is set to the number of different patterns used in

the solution given by the first model, (ICPM1).

The third model, [ICSPM3], takes the form:

$$\min \quad \sum_{q\in Q}(W\, x_{0l}^q\, m_l - \sum_{i=1}^{n}\sum_{j=1}^{n_i} z_{ijl}^q\, m_{ijl}\, w_i)\, \lambda_q$$

$$\text{s.t.} \quad \sum_{q\in Q}\sum_{l}\sum_{j} m_{ijl} z_{ijl}^q\, \lambda_q \;\geq\; \underline{d_i} \qquad i = 1,\ldots,n$$

$$\sum_{q\in Q}\sum_{l}\sum_{j} m_{ijl} z_{ijl}^q\, \lambda_q \;\leq\; \overline{d_i} \qquad i = 1,\ldots,n$$

$$\sum_{q\in Q} \lambda_q \;\leq\; U$$

$$\lambda_q \in \{0,1\} \qquad \forall q \in Q$$

where U is the parameter on which we iterate to obtain the curve of pareto optimal solutions: starting with the U obtained in the solution of model [ICSPM1] whose waste is equal to $waste^*$, the value U is decreased iteratively until there is no more feasible solution. The two extreme solutions of the pareto optimal curve are the minimum waste solution involving the smallest number of different patterns and the solution with the minimum number of patterns that can be achieved (which has typically a waste greater than $waste^*$).

To enforce integrality, we used the branching constraints defined in Section 2.3.7. Moreover, we use a new branching rule, enforcing $\sum_{q\in\hat{Q}} \lambda_q \in I\!N$ for $\hat{Q} = \{q \in Q : x_{0l}^q = 1\}$, i.e. the number of columns of multiplicity x_{0l} must be integer. These branching rules are sufficient to solve our experimental instances to optimality.

We used the two first models to solve the industrial data coming from Condat.

They were implemented in BaPCod ([43]) that uses XPressMP for solving the linear master programs and the subproblems. For the convexity constraint we compute an upper bound (K) on the number of wide rolls as:

$$\frac{\sum_i \overline{d_i}\, w_i}{W}\ 1.5.$$

The maximum multiplicity of the cutting patterns is set to $\min\{\max_i \overline{d_i}, K\}$. To obtain an incumbent solution for $ICSPM1$, we use the first fit decreasing algorithm adapted to this model and the rounding heuristic at a depth of 2, while $ICSPM2$ is initialized with the solution of $ICSPM1$. The masters are initialized with a single artificial column (4.2.2) as we have shown in chapter 4 that it was the initialization mode that performs better on the variant with tolerance on production. All branching schemes described in chapter 2 are useful to obtain integer optimal solution. The algorithm stops when the number of generated nodes exceeds 10000 or at optimality.

The instances used are real-life problem from the paper factory Condat. Their size is representative of the hardest instances that arise in practice at Condat: The average number of orders is 8 and there widths vary between 42 and 166 while the average demand is 37. Each set of data are composed of:

1. the number of items to cut,
2. the minimum and maximum widths of a cutting patterns,
3. the number of knives of the winder, defining the capacity of a pattern,
4. the width and associated demand in each order type,
5. lower and upper bounds on production for each order.

The names of data files are icsp$k0h$, where h is the number of the data file while $k = 0, ...3$ has the following meaning:

$k = 1$**:** the instance represents an order form such as that it was provided to logistics.

$k = 2$**:** same as above but a tolerance on production was introduced.

$k = 3$**:** the aim production levels represent what was really produced by the factory (as opposed to what was on the order form), a tolerance on production level is set at 2 %.

$k = 0$**:** the required production levels represent what was really produced by the factory.

The optional orders are treated as standard orders with lower bound on production set to zero.

The numerical results are presented in tables 5.1 and 5.2. Each table reports:

- N. = name of the instance.
- RtLpVal = value of the last restricted LP master problem at the root node .
- RtDB = dual bound at the root node.
- RtInc = incumbent at the root node.
- DB = best dual bound at the end of the branch-and-price procedure.
- Inc = best incumbent at the end of the branch and-price procedure.
- Nodes = number of nodes processed during the branch-and-price procedure.
- TSp = total time spent at solve the subproblems, in seconds.(Note that it represents the bulk of the computing time. Recall that we use a commercial mip solver. Computing times would be much lower with a customized solver).

- TMast = total time spent at solving restricted master problems, in seconds.
- TRh = time spent on the rounding heuristic, in seconds.
- Total = total time in seconds.

The first line corresponds to the solution of the first model while the second line to the second one.

N.	RtLpVal	RtDb	RtInc	DB	Inc	Nodes	TSp	TMast	TRh	Total
0001	1149	1149	1149	1149	1149	1	0.03	0.01	0.00	0.10
0001	2.15	3	4	3	3	31	9.41	0.01	0.00	11.79
1001	1109.8	1109.8	1121	1121	1121	378	2.34	0.22	1.18	6.75
1001	2.11	3	3	3	3	1	0.99	0.00	0.00	1.15
2001	1109.8	1109.8	1121	1121	1121	362	2.25	0.26	1.68	6.94
2001	2.10	3	4	3	3	61	15.25	0.14	0.00	17.89
3001	1132.2	1132.2	1149	1147	1147	403	2.35	0.37	1.04	7.02
3001	2.10	3	3	3	3	1	1.11	0.01	0.00	1.28
0003	25.72	25.72	123.5	123.5	123.5	413	8.85	0.62	5.92	21.14
0003	2.68	3	6	4	4	403	28.79	0.67	0.00	44.08
1003	9.79	9.79	187.5	57.5	57.5	448	15.29	0.82	7.67	30.38
1003	2.38	3	4	4	4	27	7.63	0.03	0.00	8.74
2003	5.5	5.5	27.5	27.5	27.5	82	7.91	0.09	7.60	11.73
2003	2.75	3	4	4	4	3	1.79	0.00	0.00	2.03
3003	20.58	20.58	111	33.5	33.5	55	2.29	0.10	2.24	4.43
3003	2.93	3	5	5	5	7	3.46	0.00	0.00	3.90
0004	25.3	25.3	46	46	46	5393	245.06	7.81	20.48	579.38
0004	3.21	4	8	5	5	41	15.39	0.11	0.00	17.84
1004	120.77	120.77	233	129	129	526	124.53	0.83	65.63	156.38
1004	3.31	4	7	5	5	491	337.59	1.04	0.00	374.61
2004	27.54	27.54	54	30	30	757	151.24	1.11	55.64	192.35
2004	3.52	4	8	5	5	255	166.58	0.70	0.00	183.95
3004	23.8	23.8	31	28	28	368	43.06	0.50	18.95	58.54
3004	3.77	4	7	5	5	28	18.66	0.05	0.00	20.56

Table 5.1: B&P - numerical results (1/2)

N.	RtLpVal	RtDb	RtInc	DB	Inc	Nodes	TSp	TMast	TRh	Total
0005	274.39	274.39	314	314	314	2373	218.50	3.81	17.68	415.49
0005	3.66	4	5	5	5	97	78.90	0.17	0.00	86.07
3005	248.54	248.54	260	252	252	79	9.53	0.21	4.90	13.27
3005	3.82	4	6	5	5	154	78.42	0.23	0.00	87.49
0006	25	25	69	69	69	252	9.27	0.42	3.58	17.74
0006	2.89	3	5	3	3	31	8.85	0.09	0.00	10.51
3006	23.33	23.33	110	25	25	102	9.63	0.31	8.79	15.24
3006	2.63	3	4	4	4	13	5.85	0.02	0.00	6.59
0008	0	0	0	0	0	31	1.69	0.00	1.00	2.07
0008	2.91	3	9	5	5	1558	7165.36	3.11	0.00	7312.82
0009	1444	1444	1444	1444	1444	1	0.85	0.00	0.00	1.26
0009	5.44	6	7	6	6	33	189.36	0.13	0.00	199.29
0010	20	20	20	20	20	1	0.01	0.00	0.00	0.04
0010	1.39	2	5	2	2	21	2.56	0.00	0.00	3.15
1010	19.8	19.8	20	20	20	2856	7.98	2.39	0.38	64.26
1010	1.39	2	5	2	2	21	2.88	0.02	0.00	3.63
2010	19.8	19.8	20	20	20	3158	8.51	2.56	1.68	71.99
2010	1.39	2	5	2	2	21	2.71	0.04	0.00	3.40
3010	19.4	19.4	20	20	20	6238	23.10	6.42	2.40	279.16
3010	1.42	2	3	2	2	23	2.78	0.02	0.00	3.45
0011 (*)	187.88	187.88	325	196.171	325	9799	3112.70	16.54	47.30	5143.33
0011	3.28	4	8	4	4	4996	3441.15	20.14	0.00	6275.81
2011	40.5	40.5	41	41	41	48	55.14	0.09	17.62	57.83
2011	2.76	3	8	4	4	1093	2290.48	2.68	0.00	2440.30
3011 (*)	175.47	175.47	200	180.04	194	9939	1246.91	11.61	24.90	2516.04
3011	2.74	3	9	5	5	5000	6020.21	13.65	0.00	7742.08

Table 5.2: B&P - numerical results (2/2)
(*) refers to instances that were not solved to optimality (number of limited nodes exceeded).

In Table 5.3, we give compare the solutions used in the factory Condat to ours. In the column **waste** the number between bracket is the percentage of waste over the production, and in the column **number of setups** we give the number of different cutting patterns over the total number of wide rolls used. The symbol ***** referred to optimal solutions, while **UB** is the lowest upper bound for the instances where we do not have the optimal solution. We only have the factory solution value for instance of type 1 and 0. For some instances, marked with **(+)**, the factory solution is not feasible (hence they can have a waste lower than our feasible optimal solution).

For some instances in table 5.2 we have not the order form provided to logistics, so we used only what was really produced as an input. We note that most of these industrial instances were solved to optimality, in terms of waste and number of different cutting patterns. Model $ICPM1$ take often more time than $ICPM2$ because in many cases, the branch-and-price tree grows larger for $ICPM1$. However the subproblems are harder to solve for $ICPM2$, thus times per node are more important for this model.

A more specialized code should have better performance. On one hand the columns generated for solving the first model could be used in the second model (this can be implemented by redefining the associated subproblem solution). Indeed, we note that on most instances the number of distinct cutting patterns given by the optimal solution of the first model is near (sometimes equal) to its optimal value, a lot of time is spent in generating columns that were already used in the solution of $ICPM1$. Moreover, the computing time could be improved developing a specific solver (as a dynamic program). For the subproblem associated to $ICPM2$ we could iterate on the multiplicity x_0 of a cutting pattern: for each value $x_0 = 1, \ldots, x_0^{max}$ we determine the reduced cost of the associated cutting

instance	waste		number of setups	
	factory	B&P	factory	B&P
1001	1149 (4.71 %)	1121 * (4.68 %)	3 / 49	3 *⁄ 48
2001		1121 *		3 * / 48
3001		1147 *(4.70 %)		3 * / 49
0001	1149	1149 *	3 / 49	3 * / 49
1003	123.5	57.5 *	4 / 15	4* / 13
2003		27.5 *		4 *⁄ 13
3003		33.5 * (0.47 %)		5* / 16
0003	123.5 (1.87 %)	123.5 *	4 / 15	4* / 15
1004	46 (+)	129 *	5 / 18	5 * / 18
2004		30 *		5 * / 18
3004		28 *		5 * / 19
0004	46 (0.74 %)	46 *	5 / 18	5 * / 18
3005		252 * (1.37 %)		5 * / 37
0005	314 (1.70 %)	314 *	5 / 37	5 * / 37
3006		25 * (0.95 %)		4 *⁄ 6
0006	69	69 *	3 / 6	3 * / 6
0008	345	0 *	7 / 32	5 * / 32
0009	1444	1444 *	6 / 98	6 * / 98
1010	90	20 *	2 / 9	2*⁄ 9
2010		20 *		2*⁄ 9
3010		20 *		2* / 10
0010	90 (2.28 %)	20 * (0.0050 %)	2 / 9	2* / 9
2011		41 *	5 / 27	4* / 33
3011		194 (UB)		5* / 28
0011	325 (2.40 %)	325 (UB)	5 / 27	4* / 27

Table 5.3: Comparisons with factory solutions

pattern by solving a standard knapsack problem.

5.2 AN APPLICATION WITH MINIMAL RUN LENGTH

The application described here comes from another industrial problem, where the technical constraints are different from the above models. We still consider a tolerance on production, but now the only technical constraint is a minimum number of runs set to $N = 2$. The model used to solve this application is [ICPM3] associated to the subproblem formulation [LSP5] to which we add a constraint on the minimal multiplicity of a cutting pattern:

$$\sum_l m_l \, x_{0l} \quad \geq \quad N$$

Numerical tests were done on two instances provided by Greycon with 17 items. Their characteristics are summarized in the following table:

	W	w_i	$\underline{d_i}$	$\overline{d_i}$
$inst.1$	3200	$[400, 870]$	32	36
$inst.2$	1344	$[352, 722]$	12	11

We obtained an estimate of the minimal number of setups required by solving the model [ICSPM1], then we decreased its value while minimizing the waste. The pareto curve associated to the first instance is represented in figure 5.1. The optimal waste, whose value is 0, is obtained with at most 8 cutting patterns, while when the number of distinct cutting patterns is restricted to be at most $7, 6, 5$ or 4 the waste increased to the value 220, and there is no feasible solution when this number takes a value inferior to 3.

The solutions are represented in figures 5.2 to 5.4. The two first figures (5.2 and 5.2) are associated to the first instance. Figure 5.2 corresponds to the optimal

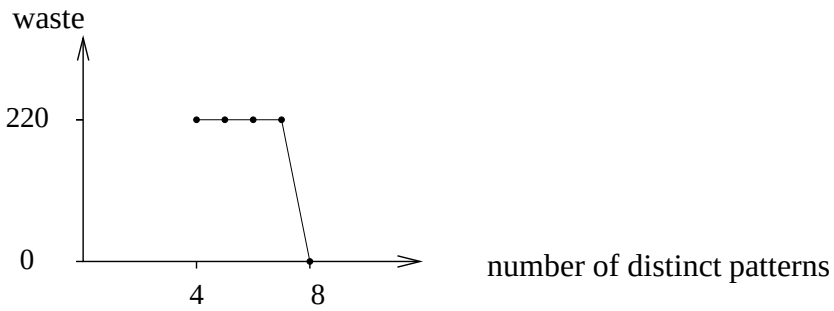
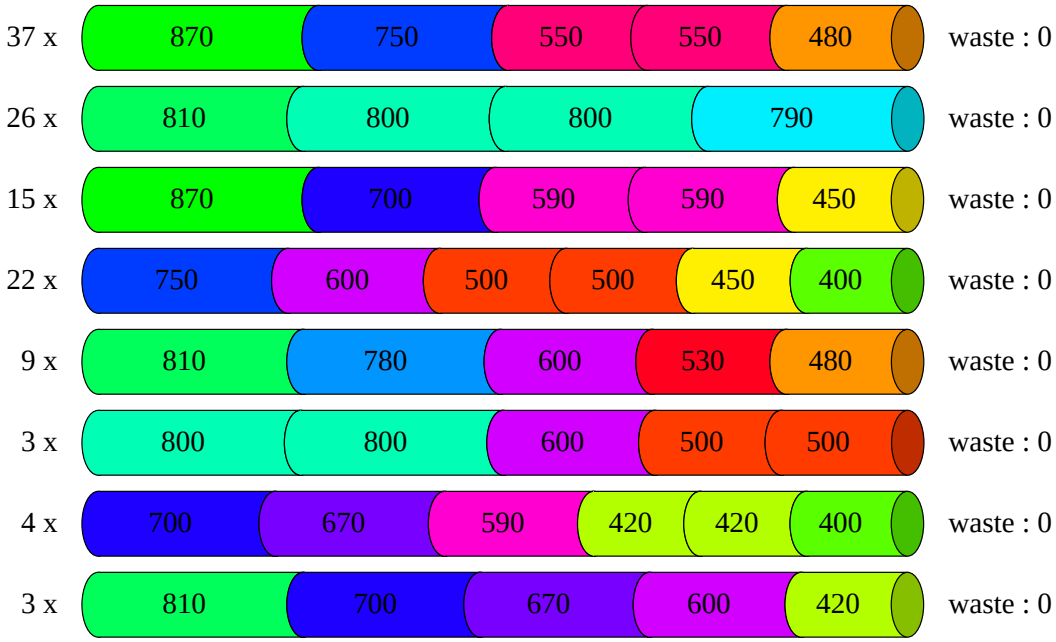


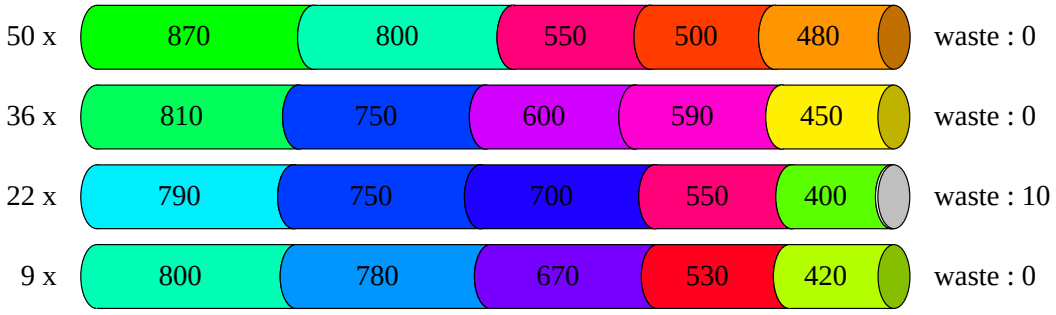
Figure 5.1: pareto optimality curve for instance 1

waste solution while 5.2 corresponds to the solution with the minimum number of setups. Figure 5.4 represents a solution for the second instance. This solution was the best compromise that can be obtained in terms of number of setups and waste. The second instance was harder to solve because of the items widths that are large relative to the wide roll width.



Number of distinct cutting patterns : 8
Number of rolls : 119
Total waste : 0

Figure 5.2: instance 1 - optimal waste



Number of distinct cutting patterns : 4
Number of rolls : 117
Total waste : 220

Figure 5.3: instance 1 - optimal number of distinct cutting patterns

Number of distinct cutting patterns : 17
Number of rolls : 88
Total waste : 5090

Figure 5.4: instance 2

Conclusion

Cette thèse donne une revue compréhensive des différentes formulations et approches de résolution associées pour le problème de découpe (CSP) et ses variantes. Les relations théoriques d'équivalence ou de dominance qui existent entre ces formulations ont été établies, ainsi que des comparaisons sur des questions pratiques telles que la symétrie dans la représentation des solutions et les schémas de branchement qu'elles induisent. Nous nous sommes alors concentrés sur l'algorithme de "branch-and-price". Nous avons développé des algorithmes exacts spécialisés pour les sous-problèmes modifiés de sac à dos qui interviennent au cours du parcours de l'arbre de "branch-and-price". Notre étude numérique complète des stratégies d'implémentation a montré quelles stratégies ont un réel impact sur l'initialisation, la stabilisation, les branchements, et dans l'obtention de solutions primales. Enfin, nous avons démontré qu'en utilisant une implémentation basique des stratégies importantes, on peut résoudre des problèmes industriels.

Le chapitre 1 replace le problème de découpe 1D dans son contexte et donne la formulation de contraintes et d'objectifs additionnels qui peuvent apparaître dans les variantes du problème. Des reformulations sont présentées au chapitre 2. Nous les présentons comme résultant d'un changement de variable dans le sous-problème de sac à dos. Les problèmes de découpe ont typiquement différentes solutions optimales. De plus, certaines formulations admettent différentes représentations d'une solution donnée. Notre discussion montre

que le modèle de Gilmore-Gomory évite cette dernière symétrie, alors que ce n'est pas le cas de la formulation en terme de flots. La formulation compacte de Kantorovich est encore plus mauvaise (car il y a un nombre exponentiel de permutations d'index donnant la même solution). Le chapitre 2 classe également les diverses formulations en classes d'équivalence en termes de leurs solutions LP et IP.

Nous avons présenté, en particulier, des formulations originales avec échanges intégrés. Nous montrons qu'elles ont un impact sur la restriction des solutions duales (de la même façon que l'inclusion de colonnes artificielles implique des contraintes duales) et par conséquent elles ont un effet de stabilisation sur une approche de génération de colonnes. Nous présentons le concept de vecteurs d'échanges et nous avons généralisé le travail de Carvalho sur les coupes duales. Cette étude théorique est achevée par des tests numériques. Nous avons montré au chapitre 4 que bien que la théorie prévoie une borne duale plus faible pour certaines de ces reformulations avec échanges, elles semblent en fait donner la même borne duale en pratique. Le meilleur effet de stabilisation est obtenu avec la formulation avec variables d'échanges de flot. Elle semble être le meilleur compromis en termes de complexité de la structure d'échange.

Les comparaisons numériques des stratégies d'implémentations du "branch-and-price" du chapitre 4 sont menées sur les variantes du CSP afin d'identifier des stratégies robustes. (Les travaux précédents présentés dans la littérature portent seulement sur les problèmes de Bin-Packing (BPP) ou de CSP standard). On observe que le BPP pur souffre typiquement davantage de la dégénérescence que le CSP standard, lui-même étant plus dégénéré que la variante avec tolérance sur la production. En effet, les exemples de BPP tendent à impliquer un grand nombre d'articles dont les largeurs peuvent être proches les unes des autres (ils

sont bien plus dégénérés si des articles identiques ne sont pas agrégés). Alors les techniques de stabilisation ont un plus grand impact. D'autre part, quand on permet la tolérance sur la production, l'objectif est de mesurer la perte et par conséquent il y a peu de solutions de même coût.

Nos résultats montrent qu'une initialisation appropriée a un impact significatif sur le nombre d'itérations de la procédure de génération de colonnes. La meilleure stratégie est l'initialisation avec les colonnes artificielles locales, si on a une bonne évaluation de la solution duale à priori (ce qui est le cas pour BPP et CSP). Les colonnes d'une solution heuristique gloutonne aident aussi. En termes de techniques de stabilisation, on peut noter que l'utilisation des contraintes de recouvrement au lieu des contraintes de partitionnement est déjà une forme de stabilisation. Nos comparaisons montrent que le boxstep dynamique n'aide pas, étant donnée notre bonne initialisation. Lisser le vecteur dual avec celui obtenu à l'itération précédente (Neame) fonctionne mieux que lisser avec le vecteur dual donnant la meilleur borne duale (Wentges). La méthode des faisceaux est efficace en terme de réduction du nombre d'itérations (particulièrement sur les problèmes les plus dégénérés comme les triplets du BPP) ; cependant, elle est plus chère en temps pour résoudre les sous-problèmes. On note que le facteur de vitesse (ou le nombre d'itérations) rapporté dans la littérature sur des techniques de stabilisation est obtenu en le comparant à une approche faible. Lorsque ces techniques sont appliquées avec une bonne initialisation leur impact est moins important.

Notre expérimentation de différentes stratégies de génération de colonnes ne fait pas clairement apparaître de stratégie gagnante : comme prévu la solution heuristique du sous-problème donne moins de temps par itération mais plus d'itérations; on observe l'effet inverse en générant plusieurs colonnes par

itération avec notre stratégie de diversification. Les heuristiques primales basées sur une approche de génération de colonnes fonctionnent bien, en particulier l'heuristique d'arrondi qui donne des solutions presque optimales (optimales dans presque tous les cas).

Des solveurs pour résoudre le sous-problème ont été développés au chapitre 3. Nous avons montré comment les résultats classiques pour le problème de sac à dos peuvent être généralisés au problème de sac à dos multi-classe avec des "setups". Nous donnons des bornes supérieures qui généralisent celles de Dantzig. Nous avons montré que l'algorithme de "branch-and-bound" classique de Horowith et Sahni se prolonge à ces variantes et nous fournissons des algorithmes de programmation dynamique. La contribution principale de ce chapitre est un schéma d'énumération intelligent pour l'algorithme de "branch-and-bound" spécialisé. Elle exploite les caractéristiques des solutions optimales pour le modèle. On doit étendre l'approche standard pour avoir un schéma d'énumération qui est glouton en terme de bornes duales et primales. L'intérêt de considérer un modèle multi-classe 0-1 plutôt que la transformation 0-1 standard ou un modèle de sac à dos en nombre entier ont été développés dans [42].

Dans le chapitre 5, nous examinons des contraintes et objectifs supplémentaires qui interviennent dans les problèmes de découpe réels : nous avons montré comment les formuler et nous avons fourni des modèles, bi-critère pour une optimisation hiérarchique, ou pour donner toutes les solutions pareto optimales. Ce chapitre prouve qu'un code générique de "branch-and-price", basé simplement sur un solveur MIP commercial pour traiter les formulations mathématiques, peut résoudre des exemples industriels. L'intérêt de cette approche est sa flexibilité pour manipuler de nouvelles contraintes spécifiques. Si on s'intéresse alors au développement d'un solveur spécifique pour le sous-problème, on peut facilement

améliorer les temps calcul.

Conclusion

This thesis gives a comprehensive view of the scope of formulations and related solution approaches for the cutting stock problem (CSP) and its variants. We have established theoretically the relative strength of each formulation and compared them on practical issues such as the symmetry in the representation of solutions and the branching scheme to which they lead. We then focused on the branch-and-price algorithm. We developed specialized exact algorithms for the modified knapsack subproblems that arise in the course of the branch-and-price tree. Our thorough numerical testing of implementation strategies has showed what strategies have a real impact on initialization, stabilization, branching, and in producing primal solutions. Finally, we demonstrated that using a basic implementation of the important strategies, one can solve industrial problems.

Chapter 1 places the one-dimensional cutting stock problem in context and provides formulation of additional constraints and objectives that may arise in variants of the problem. Reformulations are presented in Chapter 2. We take the view of presenting them as arising from a variable change in the knapsack subproblem. Cutting stock problems typically admit different optimal solutions. What is more, some formulations allow for different representations of a given solution. Our discussion showed that the Gilmore-Gomory model avoids the latter symmetry, while the arc flow formulation does not. The compact formulation of Kantorovich is even worse (as there is an exponential number of index permutations leading to the same solution). Chapter 2 also sorts the various

formulations into equivalent classes in terms of their LP and IP solutions. This classification builds on known results and completes them with new results.

In particular, we introduced original formulations with built-in exchanges. They are shown to have an impact in terms of constraining dual solutions (in the same way as including artificial columns implies dual constraints) and hence they have a stabilization effect on a column generation approach. We introduce the concept of exchange vectors and generalized the work of Carvalho on dual cuts. This theoretical study is completed by numerical tests. We showed in Chapter 4 that although theory predicts a weaker dual bound for some of these reformulations with exchanges, they seem to lead to the same dual bound in practice. The best stabilization effect is obtained with the exchange-flow formulation which seems to strike the right trade-off in terms of the complexity of the exchange structure.

Chapter 4's numerical comparisons of implementations strategies for branch-and-price are carried across the CSP variants to identify robust strategies. (Previous works reported in the literature concerned only Bin Packing Problems (BPP) or standard CSP). It is to be observed that the pure BPP typically suffers more from degeneracy than the standard CSP, itself being more degenerate than the variant with tolerance on production. Indeed, BPP instances tend to involve a large number of items whose width can be close to another (they are even more degenerate if identical items are not aggregated). Then stabilization techniques have a larger impact. On the other hand, when tolerance on production is allowed, the objective must be to measure waste and hence there are few solutions with the same cost.

Our results show that proper initialization has a significant impact on the

number of iterations of the column generation procedure. The best strategy is the initialization with local artificial columns, provided one has a good estimation of the dual solution a priori (which is the case for BPP and CSP). Columns from a greedy heuristic solution do help too. In terms of stabilization techniques, let us first note that using covering constraints instead of partitioning is already a stabilization. Our comparisons show that dynamic boxstep does not help given our good initialization. Smoothing the dual vector with those obtained at the previous iteration (Neame) works better than the smoothing with the dual vector giving the best dual bound (Wentges). The Bundle method is effective in reducing the number of iterations (specially on the most degenerate problems as the BPP triplet problems); however, it is more expensive in solving the subproblems. It is to be noticed that the speeding factor (or number of iterations) reported in the literature on stabilization techniques are obtained by comparing a poor approach. Once these techniques are applied beyond a good initialization and/or in combination, their impact is less important.

Our experimentation with different column generation strategies do not exhibit a clear winner strategy: as expected solving the subproblem heuristically leads to less time per iteration but more iterations; the opposite effect is observed when generating several columns per iteration with our diversification strategy. The primal heuristics based on a column generation approach are shown to perform well, in particular the rounding heuristic that gives close to optimal solutions (optimal in almost all cases).

The subproblem solvers were developed in Chapter 3. There, we have shown the extend to which classical results for the knapsack problem can be generalized to the multiple-class knapsack problem with setups. We gave upper bounds that generalized that of Dantzig. We showed that the classic branch-and-bound

algorithm of Horowith and Sahni extends to these variants and we provided dynamic programming algorithms. The main contribution of this chapter is an intelligent enumeration scheme for the specialized branch-and-bound algorithm. It exploits the characterization of optimal solutions for the model. One had to stretch the standard approach to have an enumeration scheme that is greedy for both primal and dual bounds. Another contribution of this chapter is to reset the boundary of knapsack problem variants that admit a greedy LP solution: after multiple choice and variant with class bounds, we now extend this to the case with setups. The interest of considering a multiple class 0-1 model rather than the standard 0-1 transformation or an integer knapsack model was developed in [42].

In Chapter 5, we offered a review of extra constraints and objectives that arise in real-life cutting stock problems: we showed how to formulate them and we provided bi-criteria models for a hierarchical optimization or to generate all pareto optimal solutions. This chapter shows that a generic branch-and-price code, that simply relies on a commercial MIP solver for dealing with the mathematical programming formulation, is able to handle industrial instances. The interest of this approach is its flexibility in handling new specific constraints. Then, if one cares to develop a specific subproblem solver, one can easily improve on the computing times.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] J.E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. http://www.brunel.ac.uk/depts/ma/research/jeb/info.html.

[3] G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, 141(2):274–294, 2002.

[4] G. Belov and G. Scheithauer. The Number of Setups (Different Patterns) in One-Dimensional Stock Cutting. Preprint MATH-NM-15-2003, TU Dresden, 2003.

[5] H. Ben Amor. Résolution du problème de découpe par générations de colonnes. Master's thesis, Ecole polytechnique de Montréal, Canada, 1997.

[6] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of Bundle and Classical Column Generation. Rapport de recherche INRIA, 5453, 2005.

[7] E. D. Chajakis and M. Guignard. Exact Algorithms for the setup knapsack problem. *INFOR*, 32(3):124–142, 1994.

[8] M. H. Correira, J.F. Oliveira, and J. S. Ferreira. Reel and sheet cutting at a paper mill. *Computers and Operations Research*, 31:1223–1243, 2004.

[9] G.B. Dantzig and P. Wolfe. Decomposition Principle for Linear Programs. *Operations Research*, 8:101–111, 1960.

[10] G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column Generation*, chapter 1. Springer, 2005.

[11] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[12] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.

[13] M.L. Fisher. The Lagrangian relaxation method for solving integer programming problem. *Management Science*, 27(1):1–18, 1981.

[14] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[15] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer Verlag, Heidelberg, 1993.

[16] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of ACM*, 21:277–292, 1974.

[17] E.L. Johnson and M.W. Padberg. A Note on the Knapsack Problem With Special Ordered Sets. *Operations Research Letters*, 1(1):18–22, 1981.

[18] M.P. Johnson, C. Rennick, and E. Zak. Skiving addition to the cutting stock problem in the paper industry. *SIAM Review*, 39(3):472–483, 1997.

[19] L. V. Kantorovich. Mathematical methods of organazing and planning production. *Management Science*, 6:363–422, 1960.

[20] J.E. Kelley. The cutting plane method for solving convex programs. *J. Soc. Indust. Appl. Math.*, 8:703–712, 1960.

[21] K.C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.

[22] K.C. Kiwiel. A dual method for certain positive semidefinite quadratic programming problems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):175–186, 1989.

[23] K.C. Kiwiel. A Cholesky dual method for proximal piecewise linear programming. *Numerische Mathematik*, 68:325–340, 1994.

[24] J. Lee. In *Situ* Column Generation for a Cutting-Stock Problem. Technical report, IBM Research Report, 2005.

[25] C. Lemaréchal. An algorithm for minimizing convex functions. *Information Processing '74*, pages 552–556, 1974.

[26] C. Lemaréchal. Nonsmooth optimization and descent methods. Technical report, IIASA, 1978.

[27] R. E. Marsten, W.W. Hogan, and J. W. Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

[28] S. Martello and P. Toth. *Knapsack problems*. John Wiley & Sons, 1990.

[29] R. K. Martin. Generating Alternative Mixed-Integer Programming Models Using Variable Redefinition. *Operations Research*, 35(6):820–831, 1987.

[30] P. J. Neame. *Nonsmooth Dual Methods in Integer Programming*. PhD thesis, University of Melbourne, March 1999.

[31] Dash Optimization. Xpress-MP: User guide and Reference Manual, Release 12. Technical report, http://www.dashoptimization.com, 2001.

[32] N. Perrot and F. Vanderbeck. Knapsack Problems with Setups. Working Paper no U-04.03, Laboratoire de Mathématiques Appliquées Bordeaux (MAB), Université Bordeaux 1., 2004.

[33] G. Scheithauer, J. Terno, A. Müller, and G. Belov. Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm. *JORS*, 52:1390–1401, 2001.

[34] H. Sural, L. N. Van Wassenhave, and C. N. Potts. The bounded knapsack problem with setups. Technical report, INSEAD working paper series - 97-71-TM, 1997.

[35] J.M. Valério de Carvalho. Exact solution of Cutting Stock Problems using column generation and branch-and-bound. *International Transactions in Operational Research*, 5(1):35–44, 1998.

[36] J.M. Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operation Research*, 86:629–659, 1999.

[37] J.M. Valério de Carvalho. Using extra dual cuts to accelerate convergence in column generation. To appear in INFORMS Journal on Computing., 2000.

[38] J.M. Valério de Carvalho. LP models for bin packing and cutting stock problems. *European Journal Of Operational Research*, 141:253–273, 2002.

[39] Pamela H. Vance. Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem. *Computational Optimization and Applications*, 9:211–228, 1998.

[40] F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Math. Program.*, A(86):565–594, July 1999.

[41] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.

[42] F. Vanderbeck. Extending Dantzig's Bound to the Bounded Multi-Class Binary Knapsack Problem. *Mathematical Programming*, 94(1):125–136, 2002.

[43] F. Vanderbeck. Dantzig-wolfe re-formulation or how to exploit simultaneaously original formulation and column generation re-formulation. Working paper U-03.24, Univ. Bordeaux 1, Talence, France, 2003.

[44] P. Wentges. Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming. *Int. Trans. Oper. Res.*, 4(2):151–162, 1997.

[45] G. Wäscher and T. Gau. Heuristics for the one-dimensional cutting stock problem: A computational study. *OR Spektrum*, 18:131–144, 1996.

[46] G. Wäscher, H. Haubne, and H. Schumann. An improved Typology for C&P Problems. Working Paper No. 24/2004 , Faculty of Economics and Management, Otto von Guericke University Magdeburg., 2004.