Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

Automatica e Ricerca Operativa

Ciclo XXVIII

Settore concorsuale di afferenza: 01/A6 - RICERCA OPERATIVA

Settore scientifico disciplinare: MAT/09 - RICERCA OPERATIVA

Models and Solutions of Resource Allocation Problems based on Integer Linear and Nonlinear Programming

Presentata da: Dimitri Thomopulos

Coordinatore Dottorato
Prof. Daniele Vigo

Relatore
Prof. Enrico Malaguti
Correlatore
Prof. Andrea Lodi

Esame finale anno 2016

Contents

Keywords

- *Two-dimensional Cutting Problems*
- *Guillotine Knapsack Problems*
- *Chance Constrained Programming*
- *Exact Algorithms*
- *Integer Linear Programming*
- *Nonlinear Programming*
- *Pricing*
- *Stochastic Programming*
- *Computational Experiments*

Ai miei Genitori, mio Fratello, i miei Amici, i miei Nonni e in particolar modo a mio Nonno Dimitris

Acknowledgements

I want to express my deeply-felt thanks to my thesis advisors, Professors Enrico Malaguti and Andrea Lodi, for their warm encouragement, thoughtful guidance.

I also want to express my gratitude to Dr. Valentina Cacchiani for her continuous support and sound advice.

Lastly, I wish to thank Professor Daniele Vigo and Professor Giacomo Nannicini.

Chapter 1

Introduction

1.1 On Resource Allocation Problems

Resource Allocation is the generic problem of assigning available resources to users in the best possible way. Usually the resources are limited, thus sets of activities compete for these resources establishing explicit and implicit dependencies, which may be subject to uncertainty. The resulting decision problem on the best use of the resources can be be decomposed in several problems that have been intensively studied in the field of operations research in the last few decades. Examples of such problems are:

- **Scheduling Problem.** Scheduling is the process of deciding, controlling and optimizing works and activities in a production process or manufacturing process. Scheduling is used to optimally allocate scarce resources to activities, and includes problems like the allocation of plant and machinery resources, the organization of human resources, and the control of production processes and materials purchase (see, e.g., Jordan [1996]).
- **Knapsack Problem.** A large variety of resource allocation problems can be cast in the framework of a knapsack problem. The aim of this problem, given a set of items, each with a weight and a profit, is to maximize the total obtained value, determining the number of each item to include in a knapsack, or more generally in a collection, so that the total weight is less than or equal to a given limit. The main idea is to consider the capacity of the knapsack as the available amount of resource and the item types as activities to which this resource can be allocated (see, e.g., Martello and Toth [1990]).
- **Cutting Stock Problem.** CSP is a general resource allocation problem where the objective is to cut pieces of stock material into pieces of specified sizes, minimizing the wasted material. Focusing more on the resources it can be defined

as the decision of subdividing a given amount of a resource into a number of predetermined allocations so that the left-over amount is minimized (see, e.g., Gilmore and Gomory [1961]).

- **Power Generation Scheduling Problem.** Power Generation Scheduling is required in order to find the optimum allocation of energy such that the annual operating cost of a power system is minimized, or the obtained profit is maximized (see, e.g., Bertsekas et al. [1983]). This problem is also strictly related to the Unit Commitment Problem or Pre-Dispatch Problem, and it has been subject to considerable discussion in the power system literature.

There is a large literature dealing with these problems and several solving techniques have been proposed. One of the most widely used modeling and solution techniques is Mathematical Programming (MP). This is one of the most effective and it can be conveniently defined as a mathematical representation aimed at programming, i.e., planning the best possible allocation of scarce resources. Many real-world and theoretical resource allocation problems may be modeled in this general framework, which involves searching for the optimal settings of decision variables satisfying all the occurring constraints, while optimizing the objective function. These constraints represent the conditions, like financial, technological and organizational conditions, which occur in the problem.
Mathematical Programming is a significantly large discipline and several subfields have been explored specifically. Among them we can mention:

- **Linear Programming**, LP in short, is a modeling technique for achieving the best objective function in a mathematical model characterized by linear functions. Hence, the objective function is linear and the constraints are linear inequalities (see, e.g., Luenberger and Ye [2008]).
  The first Linear Programming formulation of a problem was given by Kantorovich in 1939, who also proposed a method for solving it (see, e.g., Schrijver [1986]). He developed it during World War II motivated by combinatorial applications, in particular transportation and transshipment. About the same time as Kantorovich, also Koopmans introduced LPs, formulating classical economic problems as Linear Programs. In addition, during 1946-1947, Dantzig independently developed the General Linear Programming formulation. He was working for the United States Air Force, where one of his tasks was to develop mathematical models that could be used to formulate practical planning and scheduling problems. In 1947, Dantzig also invented the Simplex Method that is the most widely applied algorithm for solving Linear Problems. In fact the journal Computing in Science and Engineering listed it as one of the top 10 algorithms of the twentieth century.

A convenient expression of LP is

$$(LP\ )\max f(x) \tag{1.1}$$
$$g(x) \leq 0 \tag{1.2}$$
$$x \in \mathbb{R}^n_+ \tag{1.3}$$

- **Integer Programming** IP in short, refers to the class of constrained optimization problems in which the variables are required to be integers. In many settings the term refers to Integer Linear Programming (ILP), in which the objective function is linear and the constraints are linear inequalities (see, e.g., Jünger et al. [2010]).
  A generic ILP is conveniently expressed as

  $$(ILP\ )\max f(x) \tag{1.4}$$
  $$g(x) \leq 0 \tag{1.5}$$
  $$x \in \mathbb{Z}^n \tag{1.6}$$

  0-1 Linear Programming is a possible special case of ILP that involves only binary variables, i.e. variables that are restricted to be either 0 or 1. A generalization is Mixed-Integer Linear Programming (MILP) that involves problems in which only some of the variables x, are constrained to be integers, while other variables are allowed to be non-integers.

  Linear and Integer Linear Programming are presented, e.g., in Bertsimas and Tsitsiklis [1997], Papadimitriou and Steiglitz [1998].

- **Nonlinear Programming**, NLP in short, is a framework for modeling problems defined by constraints, over a set of real variables, along with an objective function to be maximized or minimized, where some of the constraints or the objective function are nonlinear (see, e.g., Luenberger and Ye [2008]).
  A NLP is conveniently expressed as

  $$(NLP\ )\max f(x) \tag{1.7}$$
  $$g(x) \leq 0 \tag{1.8}$$
  $$x \in X \subseteq \mathbb{R}^n \tag{1.9}$$

  where $g(x)$ is a nonlinear function.
  Similarly to the ILP approach it is possible to consider a variant of NLP where some of the variables x are constrained to be integers, while other variables are allowed to be non-integers. This is the Mixed-Integer Nonlinear Programming (MINLP) approach.

  Nonlinear Programming is presented, e.g., in Bertsekas [1999].

- **Stochastic Programming.** Stochastic Programs are mathematical programs for modeling problems that involve uncertainty. Uncertainty is usually characterized by a probability distribution on the model parameters. Some of the data incorporated into the objective function or the constraints are uncertain, and if some of the data are random, then also the solutions and the optimal objective values are themselves random. In order to deal with uncertainties in optimization, the available stochastic information is integrated into the problem formulation.

  One possible way for representing these problems is using recursive models, that take one decision now and minimize the expected costs (or utilities) of the consequences of that decision. Let us assume to have a set of decisions to be taken without full information on some random events represented by a vector x. These decisions are called first-stage decisions. Later, full information is received on the realization of some random vector ξ. Then, some corrective action y can be taken. These decisions are called second-stage decisions. This is known in literature as the Two-Stage Stochastic Program (see, e.g., Birge and Louveaux [1997]). A convenient expression of Two-Stage Stochastic Program is

$$(\mathit{2S}-\mathit{SP}\,)\max f(x)+E[Q(x,\xi)] \tag{1.10}$$

$$g(x)\leq 0 \tag{1.11}$$

$$x\in X \tag{1.12}$$

  where Q(x,ξ) is the minimized expected cost.

  In some cases, it may be more appropriate to try to find a decision that ensures that the probability of meeting a set of certain constraints is above a certain level. This is the case of Chance-Constrained Programming, first introduced by Charnes et al. [1958].

  A generic formulation of Chance-Constrained Programming problem is

$$(\mathit{CCP}\,)\max f(x,\xi) \tag{1.13}$$

$$g(x,\xi)\leq 0 \tag{1.14}$$

$$P(g(x,\xi)\geq 0)\geq p \tag{1.15}$$

$$x\in X \tag{1.16}$$

$$p\in[0,1] \tag{1.17}$$

1.2 Thesis Contribution

In this thesis we deal with some variants of these approaches used for solving the considered resource allocation problems. In the first part we focus on the Integer Programming approach for solving Two-Dimensional Guillotine Cutting Problems. We propose a new framework to model general guillotine restrictions through a Mixed-Integer Linear Formulation. In the second part we present a Branch-and-Cut algorithm for a class of Nonlinear Chance Constrained Mathematical Optimization Problems with a finite number of scenarios. We apply this algorithm to a specific Power Generation Scheduling Problem, i.e., the Mid-Term Hydro Scheduling Problem, for which we propose a chance-constrained formulation. Thus, we propose a sophisticated algorithmic combination of the three approaches Integer Programming, Nonlinear Programming and Stochastic Programming specifically tailored for solving resource allocation problems with a practical impact. Table 1.1 summarizes the combined approaches.

	LP	*ILP*	*NLP*	*CCP*
Two-Dimensional Guillotine Cutting Problem	yes	yes	no	no
Mid-Term Hydro Scheduling Problem	no	yes	yes	yes

Table 1.1: Used approaches

1.3 Applied Motivations

Making decisions on issues with important consequences has become a highly complex problem due to the many competing forces under which the world is operating today. Operations Research and Mathematical Programming techniques have been used more and more to support managerial decisions. In this thesis we consider Two-Dimensional Guillotine Cutting Problems and the Mid-Term Hydro Scheduling Problem. They belong to some of the current and crucial class of problems in which Mathematical Programming is applied.

The study of efficient and innovative methods for solving Cutting Problems is required by the economic impact of such problems. They are of great relevance in metal, wood, paper or glass industries, but also in loading, transportation, telecommunications and resource allocation in general (see, e.g., Bennell et al. [2013], Burke et al. [2006], Iori et al. [2007], Lodi et al. [2011], Malaguti et al. [2014], Vanderbeck [2001]). Depending on the industry, special features for the cuts may be required. Using *guillotine* cuts is one of the most common of them. It applies to wood cutting, and in particular to glass cutting. Glass Alliance Europe [2015] estimated that in 2014 the EU-28 glass production reached a volume of more than 33 million tonnes, a slight increase of 2% compared with 2013, involving 180,000 employees. This production level still maintains the EU as the largest glass producer in the world with a market share of around 33% of the total world market. Several industrial sectors directly depend on the production of glass. Indeed, some of the most important customers of glass industries come from the car industry, the construction sector, domestic and leisure industries. Therefore, even small savings in costs may have a relevant global impact.

The other crucial problem studied in this thesis belongs to the family of the Power Generation Scheduling Problems. Electric power today plays an exceedingly important role in the development of several economic sectors, but also in the most common aspects of modern life. Indeed, the modern economy is completely dependent on the electric power, which is one of the basic inputs of several activities. With the ever increasing per capita energy consumption and exponentially rising population, also the need for power stations, transmission lines and networks is increasing. Consequently Power Generation Scheduling Problems are becoming more and more complex and relevant. The International Energy Agency [2015] estimated that in 2013 the world energy consumption was 13,541 Mtoe, or 1.6 $\times 10^{11}$ MWh. Hydroelectricity is the most widely used form of renewable energy, accounting for 3.9 $\times 10^{9}$ MWh of production in 2013, with the generation costs that fall into a range of 50 to 100 USD/MWh. Not surprisingly Power Generation Scheduling Problems deal with resources worth millions of dollars. Thus, it is fundamental to optimize the activities that concern power production.

1.4 Thesis Methodological Outline

In §1.2 we presented the practical contents of the thesis, instead in this Section we present the methodological contents and contributions. As we mentioned before, in this thesis we deal with two problems of resource allocation solved through a Mixed-Integer Linear Programming approach and a Mixed-Integer Nonlinear Chance Constraint Programming approach, which are combinations of the approaches described in §1.1:

- *Two-Dimensional Guillotine Cutting Problem*,
- *Mid-Term Hydro Scheduling Problem.*

1.4.1 Two-Dimensional Guillotine Cutting Problem

Cutting Problems are combinatorial optimization problems, which occur in several real-world applications of industry and production. Due to the complexity and extensive nature of these problems, several different formulations and approaches have been proposed in literature (see, e.g., Bennell et al. [2013], Burke et al. [2006], Dyckhoff [1981], Gilmore and Gomory [1961], Iori et al. [2007], Lodi et al. [2011], Malaguti et al. [2014], Vanderbeck [2001]). The countless existing variants differ in terms of dimension, application field and special requirements. Two Dimensional Cutting Problems are subsets of these possible variants, which concern the best method to obtain a set of small (rectangular) *items* from one or more (rectangular) larger *panels*. Depending on the industry, special features for the cuts may be required; a very common one which applies to glass and wood cutting is to have *guillotine* cuts.

(i) In guillotine cutting, items are obtained from panels through cuts that are parallel to the sides of the panel and cross the panel from one side to the other;

(ii) Cuts can be performed in *stages*, where each stage consists of a set of parallel guillotine cuts on the shapes obtained in the previous stages. If the maximum number of stages is not allowed to exceed a value n, the problem is called $n-stage$. Otherwise, if there is no such restriction the problem is called $non-stage$;

(iii) Each cut removes a so-called *strip* from a panel. If during the cut sequence, the width of each cut strip equals the width of the widest item obtained from the strip, then the cut is denoted as *restricted*.

In Chapter 2, we propose a framework to model general guillotine restrictions in two-dimensional cutting problems formulated as Mixed-Integer Linear Programs (MILP). The modeling framework requires a pseudo-polynomial number of variables and constraints, which can be effectively enumerated for medium-size instances. Our modeling of general guillotine cuts is the first one that, once it is implemented within a state-of-the-art MIP solver, can tackle instances of challenging size. Our objective is to propose a way of modeling general guillotine cuts via Mixed Integer Linear Programs (MILP), i.e., *we do not limit the number of stages* (restriction (ii)), *nor impose the cuts to be restricted* (restriction (iii)). We only ask the cuts to be guillotine ones (restriction (i)). We mainly concentrate our analysis on the Guillotine Two Dimensional Knapsack Problem (G2KP), for which a model, and an exact procedure able to significantly improve the computational performance, are given. We also show how the modeling of general guillotine cuts can be extended to other relevant problems such as the Guillotine Two Dimensional Cutting Stock Problem (G2CSP) and the Guillotine Strip Packing Problem (GSPP). Finally, we conclude the Chapter discussing an extensive set of computational experiments on G2KP and GSPP benchmark instances from the literature.

1.4.2 Mid-Term Hydro Scheduling Problem

Mathematical Programming is an invaluable approach for optimal decision-making that was initially developed in a deterministic setting. However, early studies on problems with probabilistic (i.e., nondeterministic) constraints have appeared since the late 50s, see, e.g., Charnes et al. [1958], Prekopa [1970]. In a problem with probabilistic constraints the formulation involves a (vector-valued) random variable that parameterizes the feasible region of the problem; the decisionmaker specifies a probability α, and the solution to the problem must maximize a given objective function subject to being inside the feasible region for a set of realizations of the random variable that occurs with probability at least $1 - \alpha$. The interpretation is that a solution that does not belong to the feasible region is undesirable, and we want this event to happen with small probability α. This type of problem is a Chance Constrained Mathematical Programming problem. For convenience, we reformulated the Chance Constrained Mathematical Program (1.13)-(1.17) as

$$(CCP)\ \max f(x,\xi) \tag{1.18}$$
$$g(x,\xi) \leq 0 \tag{1.19}$$
$$P(g(x,\xi) \geq 0) \geq 1 - \alpha \tag{1.20}$$
$$x \in X \tag{1.21}$$

The main application of Chance-Constrained Programming studied in this thesis is the Mid-Term Hydro Scheduling Problem. A central problem in power generation systems is that of optimally planning resource utilization in the mid and long term and in the presence of uncertainty. Hydro power production networks usually consist of several reservoir systems, often interconnected, which are operated on a yearly basis: it is common to have seasonal cycles for demand and inflows, which can be out of phase by a few months, i.e. inflow peaks typically precede demand peaks by a few months. The Mid-Term Hydro Scheduling Problem refers to the problem of planning production over a period of several months. To be effective, such planning must take into account uncertainty affecting rainfall and energy demand, as well as the complex and nonlinear power production functions. A commonly used approach in practice is to rely on deterministic optimization tools and on the experience of domain experts to deal with the uncertainty, because of the sheer difficulty of incorporating uncertainty into the model. Many deterministic approaches can be found in the literature, see, e.g., Carneiro et al. [1990]. More recently, methodologies that can take into account the uncertainty in the model have appeared, see, e.g., Carpentier et al. [2012], but these are still rare compared to the deterministic ones.

In Chapter 3, we present a Branch-and-Cut algorithm for a class of Nonlinear Chance Constrained Mathematical Optimization Problems with a finite number of scenarios. This class corresponds to the problems that can be reformulated as Deterministic Convex Mixed-Integer Nonlinear Programming problems, but the size of the reformulation is large and quickly becomes impractical as the number of scenarios grows. The Branch-and-Cut algorithm is based on an implicit Benders decomposition scheme (see, e.g., Geoffrion [1972]), where we generate cuts as outer approximation cuts from the projection of the feasible region on suitable subspaces. The size of the master problem in our scheme is much smaller than the deterministic reformulation of the chance-constrained problem. We apply the Branch-and-Cut algorithm to the Mid-Term Hydro Scheduling Problem, for which we propose a chance-constrained formulation. We are not aware of previous work that employs a chance-constrained formulation for the mid-term hydro scheduling problem, although there has been work on the related unit commitment problem, see, e.g., van Ackooij [2014], Wang et al. [2012]. Even in the case of unit commitment, chance-constrained optimization approaches are the least commonly used in the literature, due to their difficulty [Tahanan et al., 2015, Sect. 4.4]. A computational study using data from ten hydro plants in Greece shows that the proposed methodology solves instances orders of magnitude faster than applying a general-purpose solver for Convex Mixed-Integer Nonlinear Problems to the deterministic reformulation, and scales much better with the number of scenarios. Our numerical experiments show that introducing a small amount of flexibility in the formulation, by allowing constraints to be violated with a joint probability $\leq 5\%$, increases the expected profit by 6.1%.

Chapter 2

Two-Dimensional Guillotine Cutting Problem

[1]

In this chapter we propose a framework to model general guillotine restrictions in two-dimensional cutting problems formulated as Mixed Integer Linear Programs (MIP). The modeling framework requires a pseudo-polynomial number of variables and constraints, which can be effectively enumerated for medium-size instances. Our modeling of general guillotine cuts is the first one that, once it is implemented within a state-of-the-art MIP solver, can tackle instances of challenging size. We mainly concentrate our analysis on the Guillotine Two Dimensional Knapsack Problem (G2KP), for which a model, and an exact procedure able to significantly improve the computational performance, are given. We also show how the modeling of general guillotine cuts can be extended to other relevant problems such as the Guillotine Two Dimensional Cutting Stock Problem (G2CSP) and the Guillotine Strip Packing Problem (GSPP). Finally, we conclude the Chapter discussing an extensive set of computational experiments on G2KP and GSPP benchmark instances from the literature.

2.1 Introduction

Two dimensional cutting problems are about obtaining a set of small (rectangular) *items* from one or more (rectangular) larger *panels*. Cutting problems are of great relevance in metal, wood, paper or glass industries, but also in loading, transportation, telecommunications and resource allocation in general (see, e.g., Bennell et al. [2013], Burke et al. [2006], Iori et al. [2007], Lodi et al. [2011], Malaguti et al. [2014], Vanderbeck [2001]). Depending on the industry, special features for the cuts may be required; a very common one which applies to glass and wood cutting is to have *guillotine* cuts.

[1]This chapter is based on Furini et al. [2014]

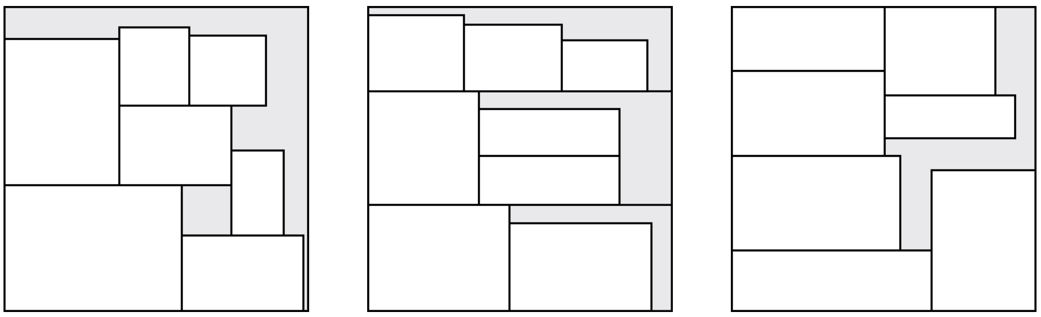


FIGURE 2.1: Examples of patterns which can be obtained through: non-guillotine cuts (left), three-stage restricted guillotine cuts (center), guillotine cuts (right).

(i) In guillotine cutting, items are obtained from panels through cuts that are parallel to the sides of the panel and cross the panel from one side to the other;

(ii) Cuts are performed in *stages*, where each stage consists of a set of parallel guillotine cuts on the shapes obtained in the previous stages;

(iii) Each cut removes a so-called *strip* from a panel. If during the cut sequence, the width of each cut strip equals the width of the widest item obtained from the strip, then the cut is denoted as *restricted*.

Our objective is to propose a way of modeling general guillotine cuts via Mixed Integer Linear Programs (MIPs), i.e., *we do not limit the number of stages* (restriction (ii)), *nor impose the cuts to be restricted* (restriction (iii)). We only ask the cuts to be guillotine ones (restriction (i)). In the following, we call these kind of cuts *general guillotine cuts* or simply *guillotine cuts*.

In Figure 2.1 we report, on the left, a pattern (cutting scheme) that cannot be obtained through guillotine cuts, in the center, a pattern that can be obtained through three-stage restricted guillotine cuts, and in the right a pattern that needs unrestricted guillotine cuts to be obtained, which are the ones we are interested in.

2.1.1 Families of cutting problems.

In the following, we will mainly concentrate our analysis on the Two Dimensional Knapsack Problem, and briefly discuss extensions of the modeling ideas to the Two Dimensional Cutting Stock Problem and the Strip Packing Problem. For convenience, we remind the definition of these problems.

- *Two-Dimensional Knapsack Problem* (2KP): we are given one rectangular panel of length L and width W, and a list of n rectangular items; each item i ($i = 1, \dots, n$) is characterized by a length l_i, a width w_i, a profit p_i, and is available in u_i copies. The 2KP requires to cut the subset of items of largest profit which can fit in the rectangular panel (without overlapping).

- *Two-Dimensional Cutting Stock Problem* (2CSP): we are given infinitely many identical rectangular panels, each one having length L and width W and a list of n rectangular items; each item i $(i = 1, \dots, n)$ is characterized by a length l_i, a width w_i, and must be cut in d_i copies. The 2CSP requires to cut all the items by minimizing the number of used panels. The special case where the demand of each items is equal to 1 is denoted as Two-Dimensional Bin Packing Problem (2BPP).
- *Strip Packing Problem* (SPP): we are given a strip having length L and infinite width and a list of n rectangular items; each item i $(i = 1, \dots, n)$ is characterized by a length l_i, a width w_i, and must be cut in d_i copies. The SPP requires to cut all the items from the strip by minimizing the used strip width.

In this chapter, we consider the *guillotine* versions of these problems (restriction (i)), i.e. the *Guillotine* 2KP (G2KP), *Guillotine* 2CSP (G2CSP) and the *Guillotine* SPP (GSPP). All these problems are NP-Hard.

2.1.2 Structure of general guillotine cuts.

Let us consider an example that shows the differences among the optimal solutions of the Guillotine 2KP obtained by imposing decreasing restrictions to the cuts performed.We compare the structure of the optimal solutions for an unweighted instance of two-dimensional knapsack of seven items, where item profits equal their areas. In the left of Figure 2.2, we report the optimal solution of the Guillotine *two-stage* 2KP, where the rectangular panel is first divided into horizontal strips, and then items are obtained from the strips by vertical cuts. Further horizontal cuts (trimming) may be necessary to obtain the final items. In the center of the figure, we represent the optimal solution of the Guillotine 2KP when we consider guillotine cuts with an *unlimited number of stages*, but cuts are *restricted*, i.e., they define strips whose width (resp., length) equals the width (resp., length) of some item which is obtained from the strip. The profit of this solution is 9.97% larger than the profit of the two-stage solution. Finally, in the right of the figure we report the optimal solution of the G2KP studied in this chapter: the only restriction imposed to the cuts is to be guillotine ones; they are not restricted nor limited in the number of stages. The profit of this solution, where all the seven items are obtained, is 17.04% larger than the profit of the two-stage solution. The example shows a case where the tree problems have different optimal solutions of strictly increasing profit.

2.1.3 Literature review.

Cutting problems were introduced by Gilmore and Gomory [1965], who considered the G2CSP and proposed the $k - stage$ version of the problem. The authors introduced

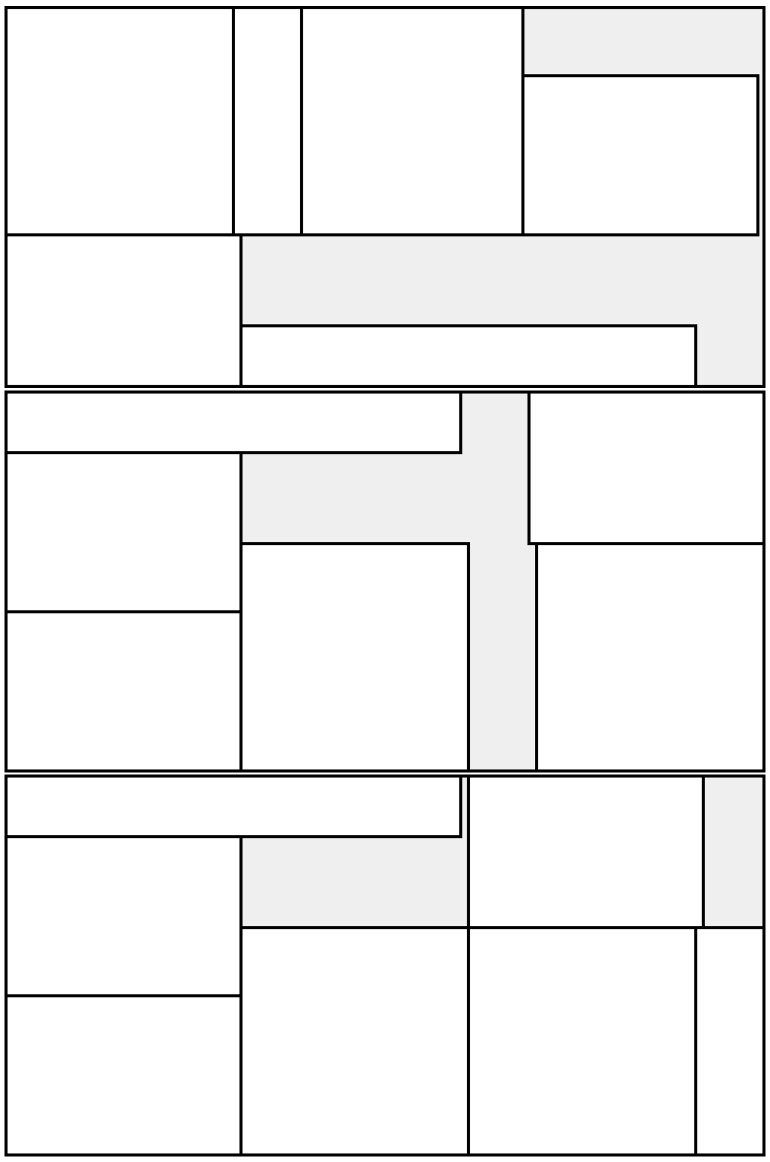


Figure 2.2: Optimal solutions for an unweighted seven items instance: two-stage 2KP (left), restricted guillotine 2KP (center), guillotine 2KP (right).

the well-known exponential-size model which is usually solved via column generation, where the pricing problem is a one dimensional Knapsack Problem. Since the seminal work of Gilmore and Gomory [1965], a relevant body of literature on two-dimensional cutting has been developed, thus, we mainly concentrate this review on 2KPs, and on guillotine cutting. For a more comprehensive survey on two-dimensional cutting and packing the reader is referred to Lodi et al. [2002] and Wäscher et al. [2007].

Concerning the *2KP* (with no specific restrictions on the cut features), Boschetti et al. [2002] proposed a branch-and-bound algorithm based on a MIP formulation. Caprara and Monaci [2004] and Fekete et al. [2007] proposed exact algorithms. The first one is based on a relaxation given by the KP instance with item weights coincident with the rectangle areas; for this relaxation a worst case performance ratio of 3 is proved. The latter is based on bounding procedures exploiting dual feasible functions.

Restricting the attention to *guillotine cutting*, the majority of contributions in the recent literature considered the case where the *number of stages is limited to two or three*. Unless explicitly stated, three stage approaches are for the restricted case. Pisinger and Sigurd [2007] consider the G2CSP and solve the pricing problem as a constraint satisfaction problem, by considering among others the case of guillotine cutting (with limited and unlimited number of stages). Puchinger and Raidl [2007] propose compact models and a branch-and-price algorithm for the three stage G2BPP. They consider the unrestricted case as well. A more application-oriented study is presented in Vanderbeck [2001], where a real-world G2CSP with multiple panel size and additional features is solved via column generation in an approximate fashion. A similar real-world problem, with the additional feature that identical cutting patterns can be processed in parallel, was recently considered by Malaguti et al. [2014].

In terms of *optimization models* not based on the Gilmore and Gomory (exponential size) formulation, Lodi and Monaci [2003] presented a compact model for the Guillotine Two Stage 2KP. Macedo et al. [2010] solved the Guillotine Two Stage 2CSP by extending a MIP formulation proposed by Valério de Carvalho [2002] for the one dimensional CSP. The extension of the model to two dimensions asks to define a set of flow problems to determine a set of horizontal strips, and a flow problem to determine how the strips fit into the rectangular panel. Silva et al. [2010] presented a pseudo-polynomial size model for the Guillotine Two and Three Stage 2CSP based on the concepts of item to-be-cut and residual plates, obtained after the cut. Recently, Furini and Malaguti [2015] extended this idea to model the Guillotine Two Stage 2KP. Finally, a computational comparison of compact, pseudo-polynomial and exponential size (based on the Gilmore and Gomory formulation) models for the Guillotine Two Stage 2CSP with multiple panel size is presented in Furini and Malaguti [2013].

Few contributions are available in the literature for *guillotine cutting problems* with an *unlimited number of stages*. This is probably due to the intrinsic difficulty of modeling guillotine restrictions. In addition to the mentioned paper by Pisinger and Sigurd [2007], where guillotine restrictions are tackled through constraint satisfaction techniques, exact approaches for G2KP have been proposed by Christofides and Whitlock [1977], Christofides and Hadjiconstantinou [1995], Cung et al. [2000]. Cintra and Wakabayashi [2004] proposed a recursive exact algorithm for the unconstrained case of G2KP, i.e., the case where no upper bound on the number of items of each type exists. A dynamic programming algorithm able to solve large-size instances of the latter problem was recently proposed by Russo et al. [2014]. The most recent exact approach to G2KP is due to Dolatabadi et al. [2012], where a recursive procedure is presented that, given a set of items and a rectangular panel, constructs the set of associated guillotine packings. This procedure is then embedded into two exact algorithms, and computationally tested on a set of instances from the literature. In addition to the reported exact methods, Hifi [1997] proposed two upper bounding procedures; concerning heuristic algorithms, we mention the hybrid algorithm by Hifi [2004] and the recursive algorithm by Chen [2008]. Finally, the related problem of determining if a given set of items can be obtained from a given large rectangle by means of guillotine cuts, was modeled through oriented graphs by Clautiaux et al. [2013].

Despite the relevance of general guillotine cutting problems, the only MIP model in the literature we are aware of was proposed by Ben Messaoud et al. [2008], and solves the guillotine GSPP. The model is polynomial in the input size, but in practice it has a very large number of variables and constraints and, as observed in Ben Messaoud et al. [2008], its linear programming relaxation produces "very loose lower bound". For these reasons the authors report computational experiments where instances with 5 items are solved in non-negligible computing time.

2.1.4 Contribution.

The main contribution of this chapter is to propose a way of modeling general guillotine cuts via MIPs. The modeling framework requires a pseudo-polynomial number of variables and constraints, which can be all explicitly enumerated for medium-size instances. To the best of our knowledge, this is the first attempt to model guillotine restrictions via MIPs that works in practice, i.e., once it is implemented within a state-of-the-art solver, can tackle instances of challenging size. In this chapter, we mainly concentrate on the G2KP. We model the problem as a MIP and propose an effective exact method for selecting a subset of the variables containing an optimal solution. Then, the resulting model can be solved by a general purpose MIP solver by only considering the subset of selected variables. This exact procedure is able to significantly increase the number of instances solved to proven optimality. In addition, we propose a number of procedures to further reduce the number of variables and constraints, and discuss conditions under which these reductions preserve the optimality of the solutions. We show how the modeling of guillotine cuts can be extended to other relevant problems such as the G2CSP and the GSPP. Finally, we conclude the chapter by an extensive set of computational experiments on benchmark G2KP and GSPP instances from the literature.

2.2 MIP modeling of guillotine cuts

Dyckhoff [1981] proposed a pseudo-polynomial size model for the (one dimensional) Cutting Stock problem, based on the concepts of *cut* and *residual element*. The idea is the following: each time a stock of length L is cut in order to produce an item i of length l_i, a residual element of size $L - l_i$ is obtained, which can be further used to produce additional items. The model associates a decision variable to each item and each (stock or residual) element, and feasible solutions are obtained by imposing balance constraints on the number of residual elements, while the cost of a solution is given by the number of used stock elements.

We extend the approach of Dyckhoff [1981] to two dimensions by using the concepts of *cut* and *plate*, where a plate can be either the original rectangular panel or a smaller rectangular residual plate obtained from the panel as result of a sequence of guillotine cuts. We concentrate on the G2KP; the main idea of the model we propose is the following: starting from the initial rectangular panel, we obtain two smaller plates through a horizontal or vertical guillotine cut; for each obtained plate, we need to decide where to perform further cuts, or eventually to keep the plate as it is when its dimensions equal the dimensions of one of the items to obtain. The process is iterated until the plates are large enough to fit some item.

In the model we propose, each cut decision is represented by a triple (q, j, o), where *position* q denotes the distance from the bottom left corner of a *plate* j, where a cut with *orientation* o is performed. In the left of Figure 2.3 we depict a vertical cut performed at position q on a generic plate j, producing two smaller plates j_1 and j_2. We depict a horizontal cut in the right of the figure.

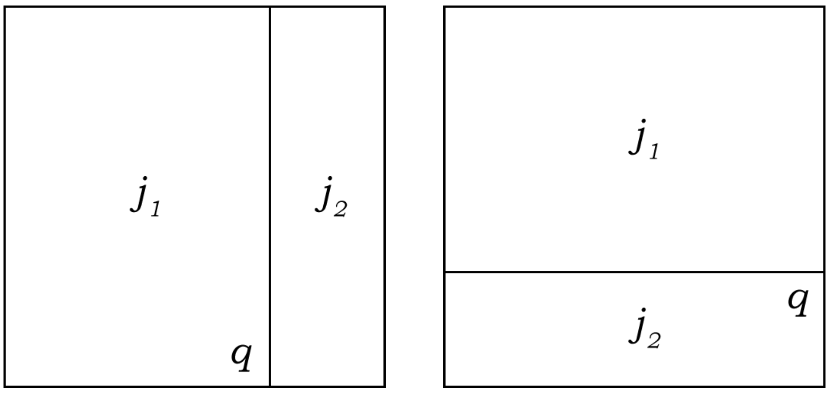


FIGURE 2.3: Vertical (left) and horizontal (right) cut at position position q producing two plates j_1 and j_2.

Without loss of generality we can assume that all problem data are positive integers. We denote by J the set of plates, where the rectangular panel, indexed by $j = 0$, has dimensions L, W, and each plate j has dimensions (l_j, w_j), with $1 \leq w_j \leq W$ and $1 \leq l_j \leq L$. The actual values of plate dimensions are discussed in the next section. We denote by $O = \{h, v\}$ the set of possible orientations for a cut (horizontal and vertical, respectively), and by $o \in O$ the generic orientation. We denote by $\bar{J} \subset J$ the subset of plates having dimensions equal to one of the items, thus, with a slight abuse of notation, $\bar{J}$ also denotes the set of items. Without loss of generality, we assume $0 \in \bar{J}$ (in case the rectangular panel does not correspond to an item to obtain, we set $u_0 = 0$). For a plate j we define by $Q(j, o)$ the set of positions where we can cut j with orientation $o \in O$. We have $Q(j, h) \subseteq \{1, \ldots, w_j - 1\}$ and $Q(j, v) \subseteq \{1, \ldots, l_j - 1)$.

The model has integer variables x^o_{qj} denoting the number of times a plate of type j is cut at position q through a guillotine cut with orientation o. Let a^o_{qkj} be a coefficient taking value 1 when a plate of type k is obtained by cutting at position q a plate of type j by a cut with orientation o, and 0 otherwise. In addition, we use the integer variables y_j, $j \in \bar{J}$, denoting the number of plates of type j that are kept as final items or, equivalently, the number of items of type j that are obtained.

The G2KP can be modeled as follows

$$
\begin{aligned}
&PP - G2KP : \max \sum_{j \in \bar{J}} p_j y_j && && (2.1)\\
&\sum_{k \in J} \sum_{o \in O} \sum_{q \in Q(k,o)} a^o_{qkj} x^o_{qk} - && && \\
&\sum_{o \in O} \sum_{q \in Q(j,o)} x^o_{qj} - y_j \geq 0 && j \in \bar{J},\ j \neq 0 && (2.2)\\
&\sum_{k \in J} \sum_{o \in O} \sum_{q \in Q(k,o)} a^o_{qkj} x^o_{qk} - && && \\
&\sum_{o \in O} \sum_{q \in Q(j,o)} x^o_{qj} \geq 0 && j \in J \setminus \bar{J} && (2.3)\\
&\sum_{o \in O} \sum_{q \in Q(0,o)} x^o_{q0} + y_0 \leq 1 && && (2.4)\\
&y_j \leq u_j && j \in \bar{J} && (2.5)\\
&x^o_{qj} \geq 0 \ integer && j \in J, o \in O, q \in Q(j,o) && (2.6)\\
&y_j \geq 0 \ integer && j \in \bar{J}, && (2.7)
\end{aligned}
$$

where the objective function (2.1) maximizes the profit of cut items; constraints (2.2) impose that the number of plates j that are cut or kept as items does not exceed the number of plates j obtained through the cut of some other plates; constraints (2.3) are equivalent to the previous constraints for plates $j \notin \bar{J}$ (hence, the corresponding y_j variables are not defined); constraint (2.4) impose that the original rectangular panel is not used more than once; constraints (2.5) impose not to exceed the maximum number of items which can be obtained. Finally, (2.6) and (2.7) force the variables to be non-negative integers.

The model has a pseudo-polynomial size, indeed, in the worst case the number of plates is WL, and each plate can be horizontally cut in $O(W)$ positions and vertically cut in $O(L)$ positions. The overall number of x variables is thus $O(WL(W+L))$, in addition to the y variables of which there are n. In the following, we denote this pseudo-polynomial size model as *PP-G2KP Model*. Indeed, not all the plates (accordingly, the variables and the constraints of the *PP-G2KP Model*) are necessary to preserve the optimality of the solutions. In the following, we discuss different ways of safely reducing the number of variables and constrains for the *PP-G2KP Model*.

The plates and, accordingly, the model variables can be enumerated by processing the item set $\bar{J}$ and the rectangular panel (plate 0) as described in Procedure 1. Starting from plate 0, new plates are obtained through vertical and horizontal cuts (line 7), and stored in set J when their size is such that they can fit some item (lines 9-12); otherwise the new plate is discarded. The definition of the set of positions $Q(j, o)$ where plate j is cut with orientation o (line 5) is discussed in the next section.

Algorithm 1: `Plate-and-variable enumeration`

```
Require: plate 0, items set $\bar{J}$
Ensure: plates set $J$, variables $x$
 1: initialize $J = \{0\}$, mark 0 as non-processed;
 2: while $J$ contains non-processed plates do
 3:     select a non-processed $j \in J$;
 4:     for all $o \in \{h, v\}$ do
 5:         compute the set of cut positions $Q(j, o)$;
 6:         for all positions $q \in Q(j, o)$ do
 7:             cut $j$ at $q$ with orientation $o$, generate plates $j_1$, $j_2$;
 8:             if $j_1 \notin$ J and $j_1$ can fit some item then
 9:                 set $J = J \cup \{j_1\}$;
10:             end if
11:             if $j_2 \notin$ J and $j_2$ can fit some item then
12:                 set $J = J \cup \{j_2\}$;
13:             end if
14:             create $x^o_{qj}$;
15:         end for
16:     end for
17:     mark $j$ as processed;
18: end while
19: return $J$, $x$.
```

A related extension of the model in Dyckhoff [1981] was proposed by Silva et al. [2010] to model two and three stage restricted guillotine 2CSPs. In Silva et al. [2010], a decision variable defines the cut of an *item* from a *plate* through two orthogonal guillotine cuts, which in addition to the item produce (up to) two residual plates. This idea cannot be extended to the unrestricted case, where the position of a cut may not correspond to the size of an item. In the following section we provide a lower bound on the largest profit loss which is incurred by considering the restricted case instead of the unrestricted one. An extension of the model of Silva et al. [2010] to two stage guillotine knapsack problems is discussed in Furini and Malaguti [2015].

Finally, we mention Arbib et al. [2002], where a further extension of the model in Dyckhoff [1981] is presented. The authors consider a one dimensional Cutting Stock problem where the residual elements can be re-used and, in the specific application case, combined, so as to obtain the requested items.

2.2.1 Definition of the cut position set

Model (2.1)–(2.7) can have very large size, depending on the cardinality of sets J and $Q(j,o)$, $j \in J$, $o \in O$. The number of plates that we consider and the number of cuts performed on each plate (eventually producing new plates) determine, in practice, the size of the model. Thus, a crucial question to be answered is the following:
Given a plate j of length l_j and width w_j, how should $Q(j,o)$, $o \in O$, be defined in order to minimize the number of variables and plates of the model, while preserving the optimality of the solution?
Let I_j be the set of items that can fit into plate j, i.e., $I_j = \{i \in \bar{J}:\ l_i \le l_j,\ w_i \le w_j\}$. The *complete position set (Q)* where a cut can be performed includes the dimensions of items $i \in I_j$, and all combinations of the items $i \in I_j$ dimensions, and is defined as follows:

$$Q(j,h) = \left\{ q:\ 0 < q < w_j;\ \forall i \in I_j,\ \exists n_i \in \mathbb{N},\ n_i \le u_i,\ q = \sum_{i \in I_j} n_i w_i \right\}, \tag{2.8}$$

and

$$Q(j,v) = \left\{ q:\ 0 < q < l_j;\ \forall i \in I_j,\ \exists n_i \in \mathbb{N},\ n_i \le u_i,\ q = \sum_{i \in I_j} n_i l_i \right\}. \tag{2.9}$$

These positions are known in the literature as *discretization points, and a pattern where cuts are performed at discretization points is known as a* normal *or* canonical *pattern, see Christofides and Whitlock [1977], Herz [1972].* All the combinations of items defining the complete position set can be effectively obtained by a Dynamic Programming (DP) algorithm. The DP algorithm we used is an extension to the case of items available in several copies of the one described in Trick [2003] (which only considers single items).

Let us also define the *restricted position set (Q_R)*, including only the dimensions of items $i \in I_j$, as:

$$Q_R(j,h) = \{q:\ \exists i \in I_j,\ q = w_i\}, \quad Q_R(j,v) = \{q:\ \exists i \in I_j,\ q = l_i\}. \tag{2.10}$$

Note that one can remove symmetric cut positions for a plate j from set $Q(j,o)$, $o \in O$ (resp. $Q_R(j,o)$), i.e.:

$$(w_j - q) \notin Q(j,h),\ \forall q \in Q(j,h),\ q < w_j/2$$
$$(l_j - q) \notin Q(j,v),\ \forall q \in Q(j,v),\ q < l_j/2.$$

These positions are automatically discarded by the DP algorithm.

Considering the *PP-G2KP Model* with the restricted position set only, does not guarantee in general the optimality of the corresponding solution. The following theorem states a condition under which the optimality is preserved.

Theorem 2.1. *If a plate j can fit at most five items by guillotine cuts, then an optimal solution to the* PP-G2KP Model *exists by considering the positions q for plate j restricted to $Q_R(j,h)$ and $Q_R(j,v)$.*

Proof. Proof of Theorem 2.1. We need to show that, when cutting five items from a plate, any packing can be obtained by considering restricted cut positions only. Without loss of generality, consider the first cut to be vertical. When performing the first guillotine cut, either we obtain two new plates which contain two and three items, respectively; or we obtain two new plates which contain one and four items, respectively. In the latter case the first cut can be performed at a position corresponding to the length of the item which is alone in the new plate. Assume instead the first case holds, and consider the new plate containing two items. If they are placed one on the side of the other (as in the left of Figure 2.4), it was possible to separate one of them with the first cut. When the two items are placed one on top of the other (as in the right of Figure 2.4), the first cut could be performed at a position equal to the length of the longest of the two. Similar considerations apply when cutting four items from a plate. □

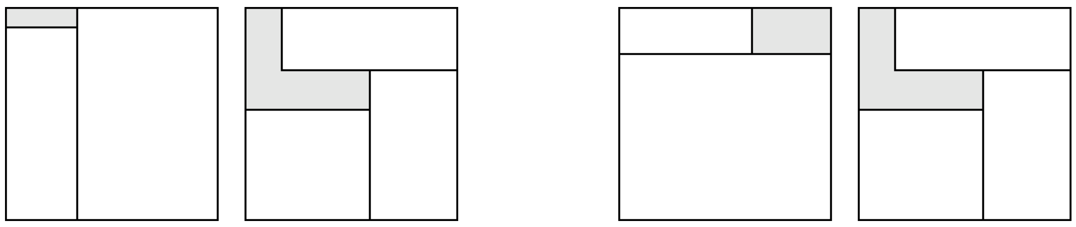


FIGURE 2.4: Possible configurations after separating five items with a vertical guillotine cut.

Consider the following

Example 1. Consider an instance of the G2KP of six items with dimensions $l = [47, 40, 40, 40, 11, 4]$ and $w = [34, 30, 30, 8, 31, 60]$ and a rectangular panel of dimensions $L = 102,\ W = 51$ (see Figure 2.5). All items can be obtained from the rectangular panel through guillotine cuts (in Figure 2.5, the first cut is vertical and separates the panel in two new plates containing three items each), but no feasible solution to the *PP-G2KP Model* with q restricted to $Q_R(j,o),\ o \in O$ allows to obtain the six items.

From Example 1 it follows that

Remark 2.2. The value of five items in Theorem 2.1 is tight.

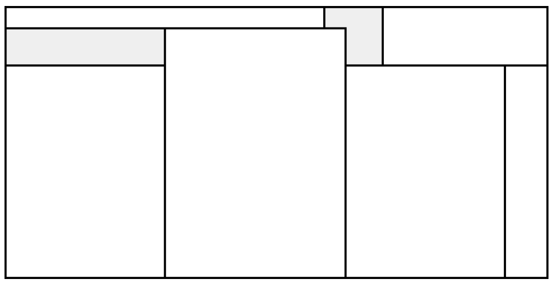


FIGURE 2.5: A six items packing that cannot be obtained by considering restricted positions Q_R only.

Given the result of Theorem 2.1, a reduction in the number of variables of the *PP-G2KP Model* can be obtained by considering positions in set Q_R for plates with the property that they can fit at most five items. Since exactly checking this condition can be computationally expensive, we considered the following relaxation. A sufficient condition for this property to hold is that the cumulate area of the six smallest items exceeds the plate area. In the following we denote the reduction obtained by checking the previous sufficient condition as `Cut-Position` reduction.

Restricting the positions to $Q_R(j,h)$ and $Q_R(j,v)$ for *all* plates has a large impact on the number of variables and plates of the *PP-G2KP Model* (see Section 2.4.1) but potentially leads to sub-optimal solutions. Thus it is natural to wonder what is the loss of profit in the worst case. In the following we denote the *PP-G2KP Model* with variables restricted to the the position sets Q_R as *Restricted PP-G2KP model.* Let z_R be the optimal solution of the *Restricted PP-G2KP model*, and z_U the optimal solution of the *PP-G2KP Model* (with complete position and Q). The following proposition provides an upper bound on the profit in the worst case.

Remark 2.3. In the worst case, $\frac{z_R}{z_U} \leq \frac{5}{6}$.

Proof. Proof of Remark 2.3. The result follows from Example 1 when the profit is the same for all the six items. □

Notice that the *Restricted PP-G2KP model* allows to perform a cut on a plate without obtaining a final item: this may happen each time the dimension of an item is the combination of the dimensions of two or more smaller items. As an example, if there are three items with widths $2, 3, 5$, the *Restricted PP-G2KP model* would allow to cut at position $q = 5$, and then to perform a further cut at position $q = 2$ on the obtained plate. Hence, the width of the strip obtained by cutting at position 5 would not correspond to the width of one of the obtained items. For this reason, the *Restricted PP-G2KP model* can produce solution which do not satisfy the definition of restricted guillotine cuts given in Section 2.1.

In order to solve the *restricted* G2KP, one possibility is to extend the modeling ideas presented by Silva et al. [2010] for the Guillotine Two and Three Stage 2CSP, and adapted by Furini and Malaguti [2015] to the 2KP. By removing the limitation on the number of stages, the model in Furini and Malaguti [2015], which cuts *items* from *plates*, can solve the restricted G2KP.

Another reduction of the model size can be obtained by removing redundant cuts (denoted as `Redundant-Cut` reduction in the following). Note that, while the `Cut-Position` reduction can reduce the number of plates in the model, the `Redundant-Cut` reduction do not affect the number of plates, but only the number of cut positions (and thus the variables of the model).

We say that q is a *trim cut* on plate j when cutting plate j at position q produces a single useful plate j_1 (the second produced plate j_2 is waste).

Remark 2.4. Given a plate j, one can remove a trim cut at position q with orientation o from $Q(j, o)$, while preserving the optimality of the solution of the *PP-G2KP Model*, in the following cases:

1. Plate j can *only* be obtained through a sequence of two orthogonal trim cuts on a larger plate.
2. Plate j can be obtained from one or more larger plates, but *always* through trim cuts, and at least one of these trim cuts has orientation o.

Proof. Proof of Remark 2.4. In both cases, plate j is obtained anyway through an alternative cut sequence, and thus the corresponding variable can be removed from the model preserving optimality. □

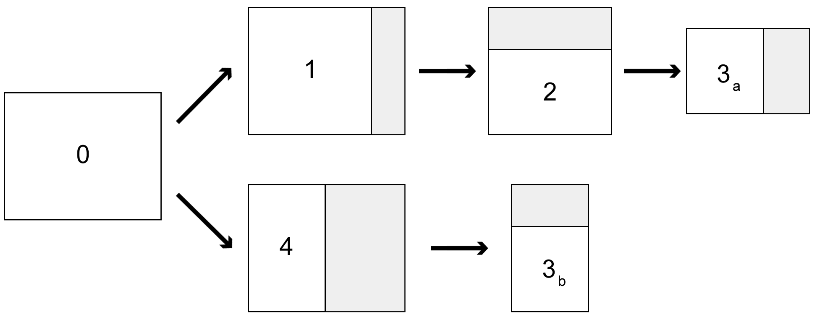


Figure 2.6: Trim cuts on plate 2 producing item 3a can be removed.

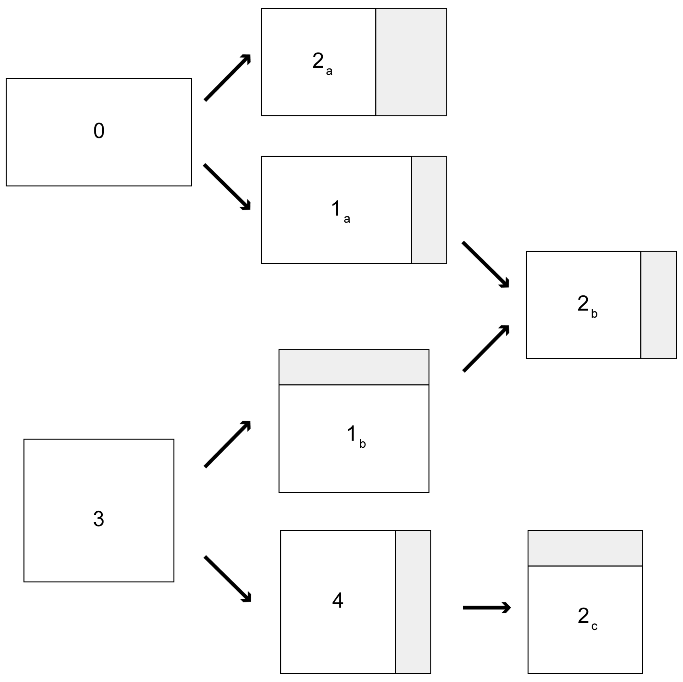


FIGURE 2.7: Trim cuts on plate 1 producing item 2b can be removed.

As an example of the first case, consider top of Figure 2.6: plate 0 is vertically trimmed obtaining plate 1, and plate 1 is horizontally trimmed obtaining plate 2. There are no other plates that can be cut to produce plate 2. The further trim cut of plate 2 to obtain plate 3 can be safely removed, because plate 3 is also obtained in the sequence $0 \rightarrow 4 \rightarrow 3$.

As an example of the second case, consider plate 1 in Figure 2.7. Since plate 1 is simultaneously generated from 0 and from 3 through a vertical and horizontal trim cut, respectively, no further trim cuts are considered on plate 1.

Conditions 1 and 2 of Remark 2.4 are checked during the enumeration of plates and variables through Procedure 1. In order to check these conditions, we associate four flags to each plate. The flags can assume values -1, 0, 1 (obtained only through a trim cut, not obtained through a cut with the same orientation, obtained without a trim cut):

- Flag sh indicates the status of the plate with respect to a cut with orientation h;
- Flag sv indicates the status of the plate with respect to a cut with orientation v;
- Flag fh indicates the status of flags sh for all the father plates of the plate;
- Flag fv indicates the status of flags sv for all the father plates of the plate.

At step 3 of Procedure 1 we select plate j, and check if it can be eventually obtained by further cuts from plates in J. If this is the case, we cannot safely remove redundant cuts, otherwise, at step 6:

- if one (or more) of the flags of plate j has value -1, do not perform trim cuts on j in the flag orientation.

At step 7, if a *new* plate $j_1 \notin J$ is obtained from j through a trim cut with orientation h:

- set the flag sh of j_1 to -1;
- set the flag sv of j_1 to 0;
- set the flags fh and fv equal to flags sh and sv of j;

if a new plate j_1 is obtained from j through a trim cut with orientation v:

- set the flag sv of j_1 to -1;
- set the flag sh of j_1 to 0;
- set the flags fh and fv equal to flags sh and sv of j;

if an *existing* plate $j_1 \in J$ is obtained from j through a trim cut:

- if sh of j is larger than -1, set the flag fh of j_1 to 1;
- if sv of j is larger than -1, set the flag fv of j_1 to 1;

if a plate (new or existing) j_1 is obtained from j without a trim cut:

- set all flags of j_1 to 1.

The computational effect on the number of variables and plates of the reductions discussed in this section are highlighted in Section 2.4.

2.2.2 Model extensions: Cutting Stock problem and Strip Packing problem

The modeling ideas of Section 2.2 can be extended to model other two-dimensional guillotine cutting problems. We present in this section two MIP models for the G2CSP and the GSPP.

By using the same variables defined for the *PP-G2KP Model*, a MIP formulation for the G2CSP reads as follows

$$
\begin{aligned}
PP-G2CSP: \min \quad & \sum_{o \in O} \sum_{q \in Q(0,o)} x^o_{q0} + y_0 && && (2.11)\\
& (2.2), (2.3) && &&\\
& y_j \geq d_j && j \in \bar{J}, && (2.12)\\
& x^o_{qj} \geq 0 \ integer && j \in J, o \in O, q \in Q(j,o) && (2.13)\\
& y_j \geq 0 \ integer && j \in \bar{J} && (2.14)
\end{aligned}
$$

where the objective function (2.11) minimizes the number of rectangular panels that are used; and constraints (2.12) impose to satisfy the demand associated with the items. The remaining constraints have the same meaning as in the *PP-G2KP Model*.

The PP-G2CSP model can be extended to the case of panels available in p different sizes, by defining p initial panels 0^t and a coefficient c_t, $t = 1, \dots, p$, specifying the corresponding cost. The objective function is promptly modified to:

$$
\min \sum_{t=1,\dots,p} c_t \left(\sum_{o \in O} \sum_{q \in Q(0^t,o)} x^o_{q0^t} + y_{0^t} \right). \qquad (2.15)
$$

To model the GSPP, we first need an upper bound W on the optimal solution value. We consider the first cut performed on the strip ($j = 0$) to be horizontal (h), with $Q(0,h) = \{1, \dots, W\}$ and do not define vertical cuts for the strip (i.e., $Q(0,v) = \{\emptyset\}$); in addition, out of the two parts obtained from the first cut, only the bottom one is a finite rectangle that can be used, while the top part is the residual of the infinite strip. The width of the obtained initial rectangle is in $Q(0,h)$ and equals the solution value. We use a variable z denoting the solution value, in addition to the variables defined for the *PP-G2KP Model*. A MIP formulation for the GSPP is then

$$
\begin{aligned}
PP-GSPP: \min \quad & z && && (2.16)\\
& z \geq q x^h_{q0} && q \in Q(0,h) && (2.17)\\
& \sum_{q \in Q(0,h)} x^h_{q0} = 1 && && (2.18)\\
& (2.2), (2.3) && &&\\
& y_j \geq d_j && j \in \bar{J} && (2.19)\\
& x^o_{qj} \geq 0 \ integer && j \in J, o \in O, q \in Q(j,o) && (2.20)\\
& y_j \geq 0 \ integer && j \in \bar{J} && (2.21)
\end{aligned}
$$

where the objective function (2.16) and constraints (2.17) minimize the (vertical) distance of the first cut from the bottom of the strip, constraint (2.18) impose to have one horizontal first cut (where q is the width of the cut, with $q \in Q(0, h)$). The remaining constraints have the same meaning as in the previous models.

2.3 An effective solution procedure for the *PP-G2KP Model*

Tackling directly the *PP-G2KP Model* through a general-purpose MIP solver can be out of reach for medium-size instances due to the large number of variables and constraints, thus in this section we describe an effective exact solution procedure based on *variable pricing*, aiming at reducing the number of variables and quicken the computational convergence.

The procedure starts by enumerating all the *PP-G2KP Model* variables by means of Algorithm 1, considering the complete position set Q (see Section 2.2). Symmetric cut positions are not generated (Section 2.2.1). Variables are then stored in a *variable pool*. We denote the *PP-G2KP Model* with all these variables as *Complete PP-G2KP Model*. The variable pool of the *Complete PP-G2KP Model* can be preprocessed by means of the `Cut-Position` and `Redundant-Cut` reductions, so as to reduce its size.

We perform two subsequent variable pricing procedures executed in cascade. The first one concerns the solution of the linear relaxation of the *PP-G2KP Model*, where variables having positive reduced profit are iteratively selected from the variable pool. The value of the linear programming relaxation of the *PP-G2KP Model*, denoted as LP in the following, gives an upper bound on the optimal integer solution value. By exploiting the dual information from the linear programming relaxation, and by computing a feasible solution of value LB, a second pricing of the variables can be performed. This second variable pricing allows us to select from the variable pool all the variables that, by entering in the optimal base with an integer value (e.g., after a branching decision), could potentially improve on the incumbent solution of value LB. We denote the *PP-G2KP Model* after the second variables pricing as *Priced PP-G2KP Model*.

The details on the two variable pricing procedures are given in the next section.

2.3.1 Variable pricing procedures

The linear programming relaxation of the *PP-G2KP Model* can be solved by variable pricing, where we iteratively solve the model with a subset of variables and exploit dual information to add variables with positive reduced profit. Initially, we relax the integrality requirements for the variables (constraints (2.6) and (2.7)) to:

$$x^o_{qj} \geq 0 \qquad j \in J,\ o \in O,\ q \in Q(j,o), \tag{2.22}$$

$$y_j \geq 0 \qquad j \in \bar{J}, \tag{2.23}$$

and we initialize the resulting linear program (2.1)–(2.5) and (2.22), (2.23) with all the y_j, $j \in \bar{J}$ variables and the x^o_{qj}, $j \in J$, $o \in O$, $q \in Q_R(j,o)$ variables, i.e., variables corresponding to cuts in the restricted positions set Q_R (see Section 2.2). The solution of the resulting linear programming relaxation provides optimal dual variables π_j associated with constraints (2.2) and (2.3).

The reduced profit of a variable x^o_{qj}, associated with a cut of plate j producing (up to) two new plates j_1 and j_2, is readily computed as:

$$\tilde{p}(x^o_{qj}) = \pi_{j_1} + \pi_{j_2} - \pi_j, \tag{2.24}$$

and can be evaluated for all the variables of the *Complete PP-G2KP Model*, stored in the pool, in linear time in the size of the pool. The reduced profit $\tilde{p}(x^o_{qj})$ represents the change in the objective function for an unitary increase in the value of the corresponding variable x^o_{qj}.

We optimally solve the linear relaxation of the *PP-G2KP Model* by iteratively adding variables with positive reduced profit, i.e., a subset of

$$\left\{x^o_{qj},\ j \in J,\ o \in O,\ q \in Q(j,o) : \tilde{p}(x^o_{qj}) > 0\right\},$$

and then re-optimizing the corresponding linear program. When all variables in the variable pool have a non-positive reduced profit, then the linear programming relaxation of the *Complete PP-G2KP Model* is optimally solved, providing us an upper bound of value LP.

Given a feasible solution of value LB, the optimal value LP of the linear programming relaxation, and the optimal value of the dual variables π^*_j, $j \in J$, we perform a last round of pricing. The *Priced PP-G2KP Model* is defined by including the y variables and the subset of the x variables

$$\left\{x^o_{qj},\ j \in J,\ o \in O,\ q \in Q(j,o) : \lfloor \tilde{p}(x^o_{qj}) + LP \rfloor > LB\right\}.$$

This includes all the variables in base plus the variables that, by entering in the current basic solution with value 1 (i.e., at the minimal non-zero integer value), would produce a solution of value $z > LB$.

The effectiveness of the last round of variable pricing in reducing the number of variables of the *Priced PP-G2KP Model* largely depends on the gap between the upper bound value LP and the value of a feasible solution LB. Heuristic feasible solutions of excellent quality can be computed by solving the *Restricted PP-G2KP model*, defined by the variables from the restricted position set Q_r (see Section 2.4.1 in the following).

The exact solution procedure is summarized in Algorithm 2.

Algorithm 2: `Solution procedure for the` *`PP-G2KP Model`*

```
1: Set $LB$ to the value of a feasible solution to the *PP-G2KP Model*;
2: Generate the variable pool through Algorithm 1;
3: Apply the Cut-Position and Redundant-Cut reductions to the pool variables;
4: Initialize the model with the variables of the restricted position set $Q_R$;
5: repeat
6:     Solve the linear programming relaxation, compute reduced profits, add the variables with
       positive reduced profit from the pool,
7: until variables with positive reduced profit exist;
8: Let $LP$ be the optimal solution value of the linear relaxation of the *PP-G2KP Model*;
9: *Final Pricing*: define the *Priced PP-G2KP Model* by including the $x$ variables with reduced
   profit $\tilde{p}(x)$ such that $\lfloor \tilde{p}(x) + LP \rfloor > LB$, and all the $y$ variables;
10: Solve the *Priced PP-G2KP Model* with a MIP solver.
```

2.4 Computational Experiments

To the best of our knowledge, the modeling framework of guillotine restrictions that we propose in this chapter is the first approach based on Mixed-Integer Linear Programming that is able to solve benchmark instances to optimality. Thus, the scope of the reported computational experiments is broader than simply comparing the obtained results against previous approaches. Namely, with these experiments we wish to evaluate:

- the size and practical solvability of the *Complete PP-G2KP Model* for a set of G2KP benchmark instances by means of a general purpose MIP solver;
- the effectiveness of the proposed `Cut-Position` and `Redundant-Cut` reductions in removing variables and constraints from the *Complete PP-G2KP Model* .
- the capability to solve the *PP-G2KP Model* by means of the pricing procedure described in Section 2.3 (*Priced PP-G2KP Model*);
- the quality of the solutions that are obtained by optimally solving the *PP-G2KP Model* by considering the restricted position set Q_R only;
- finally, we wish to discuss the computational performance of our framework with respect to a state-of-the-art combinatorial algorithm for the G2KP (Dolatabadi et al. [2012]), and with the only alternative MIP formulation of guillotine restrictions we are aware of, which is described for the GSPP (Ben Messaoud et al. [2008]).

We performed all the computational experiments on one core of a Core2 Quad Q9300 2.50GHz computer with 8 GB RAM, under Linux operating system. As linear programming and MIP solver we used IBM ILOG CPLEX 12.5.

In the computational experiments, we considered two sets of classical two-dimensional instances, listed in Table 2.1. The first set of 21 instances, for which Dolatabadi et al. [2012] reports computational results as well, is from the OR library (OR-Library); the second set of 38 instances is from Hifi and Roucairol [2001]. Both sets include weighted and unweighted instances, where the profit of each item is given, or equals the item area, respectively.

	unweighted	*weighted*
OR-Library	gcut1 - gcut12, wang20,	cgcut1 - cgcut3, okp1 - okp5,
Hifi and Roucairol [2001]	2s, 3s, A1s , A2s, STS2s, STS4s, OF1, OF2, W, CHL1s, CHL2s, A3, A4, A5, CHL5, CHL6, CHL7, CU1, CU2, Hchl3s, Hchl4s, Hchl6s, Hchl7s, Hchl8s.	HH, 2, 3, A1, A2, STS2, STS4, CHL1, CHL2, CW1, CW2, CW3, Hchl2, Hchl9.

Table 2.1: List of the considered G2KP instances.

In order to classify the instances according to the size, we generated the *Complete PP-G2KP Model*, and we grouped the instances into three sets, according to the corresponding number of variables. Table 2.2, 2.3 and 2.4 reports the features of *small* (less than 100,000 variables), *medium* (between 100,000 and 500,000 variables), and *large*-size instances, having more than 200,000 variables).

For each instance, the table reports the name (*name*), the optimal solution value (*opt*), the number of different items n, the total number of items $\bar{n}$, the largest ratio between an item length or width and the length or width of the rectangular panel (ρ). Then the table reports the number of variables of the *Complete PP-G2KP Model* (*vars*) and the corresponding number of plates (*plates*). We solved the model with the CPLEX MIP solver by allowing 1 hour of computing time. The table reports the effective computing time (t) and the corresponding percentage optimality gap (*gap*), computed as $100\frac{(UB_{MIP}-LB_{MIP})}{UB_{MIP}}$ (where UB_{MIP} and LB_{MIP} are the lower and upper bound achieved by the MIP solver at the end of the computation).

The size of the *Complete PP-G2KP Model* can be very large, in particular for the *gcut* instances, the largest model (*gcut*12) has more than 66 million variables and almost 0.5 million constraints. In addition to a very large size, since no reductions are applied to the *Complete PP-G2KP Model*, it may contain equivalent solutions. The CPLEX MIP solver can solve to optimality 24 out of 26 small-size instances, 12 out of 18 medium-size instances, and none of the 21 large-size instances. For instance $A5$ and all the

	instance features				*Complete PP-G2KP Model*			
name	*best*	*n*	$\bar{n}$	ρ	*vars*	*plates*	*t*	*gap*
cgcut1	244	7	16	10.0	801	140	0.1	0.0
CHL5	390	10	18	20.0	2,858	345	0.6	0.0
Hchl8s	911	10	18	49.0	13,997	896	*tl*	1.8
OF2	2,690	10	24	10.0	37,261	2,110	66.0	0.0
cgcut3	1,860	19	62	6.4	38,485	1,860	58.6	0.0
wang20	2,721	19	42	6.4	38,485	1,860	60.7	0.0
3	1,860	20	62	6.4	38,485	1,860	81.6	0.0
3s	2,721	20	62	6.4	38,485	1,860	60.7	0.0
W	2,721	20	62	6.4	38,485	1,860	56.5	0.0
OF1	2,737	10	23	10.0	38,608	2,098	53.9	0.0
gcut1	48,368	10	10	3.8	39,896	4,429	8.8	0.0
A1	2,020	20	62	5.6	45,333	2,040	85.4	0.0
A1s	2,950	20	62	5.6	45,333	2,040	82.3	0.0
cgcut2	2,892	10	23	10.0	52,590	2,017	58.4	0.0
2	2,892	10	23	10.0	52,590	2,017	55.2	0.0
2s	2,778	10	23	10.0	52,590	2,017	57.2	0.0
CHL2	2,326	10	19	6.1	57,567	2,348	104.1	0.0
CHL2s	3,279	10	19	6.1	57,567	2,348	110.4	0.0
A2	2,505	20	53	5.0	61,047	2,276	236.6	0.0
A2s	3,535	20	53	5.0	61,047	2,276	131.0	0.0

TABLE 2.2: Small-size instances.

	instance features				*Complete PP-G2KP Model*			
name	*opt*	*n*	$\bar{n}$	ρ	*vars*	*plates*	*t*	*gap*
STS2	4,620	30	78	8.5	118,036	3,383	289.8	0.0
STS2s	4,653	30	78	8.5	118,036	3,383	385.7	0.0
Hchl9	5,240	35	76	7.6	130,738	3,666	612.5	0.0
A3	5,451	20	46	5.7	134,164	3,752	435.5	0.0
HH	11,586	5	18	7.5	141,167	5,486	*tl*	4.5
A4	6,179	20	35	10.0	179,759	4,860	1,259.0	0.0
gcut5	195,582	10	10	3.8	250,327	25,336	*tl*	1.4
okp1	27,589	15	50	100.0	255,497	7,947	490.5	0.0
okp3	24,019	30	30	33.3	261,074	8,356	*tl*	8.8
okp4	32,893	33	61	100.0	287,773	9,049	684.4	0.0
okp2	22,503	30	30	100.0	289,825	9,506	*tl*	30.0
CU1	12,330	25	82	5.0	335,415	7,700	1,460.8	0.0
STS4	9,700	20	50	7.1	352,590	7,224	2,476.6	0.0
STS4s	9,770	20	50	7.1	352,590	7,224	2,452.4	0.0
okp5	27,923	29	97	100.0	368,529	9,506	2,118.6	0.0
CW1	6,402	25	67	5.0	410,004	8,407	*tl*	70.7
gcut9	919,476	10	10	3.4	474,360	38,936	2,847.9	0.0
A5	12,985	20	45	10.2	494,455	10,402	*tl*	100.0

TABLE 2.3: Medium-size instances.

larger ones, gaps at time limit are 100%, meaning that no feasible solution is found by the solver.

	instance features				*Complete PP-G2KP Model*			
name	*best*	*n*	$\bar{n}$	ρ	*vars*	*plates*	*t*	*gap*
Hchl4s	12,006	10	32	8.5	525,902	9,670	*tl*	100.0
Hchl3s	12,215	10	51	8.5	526,403	9,670	*tl*	100.0
CHL1	8,699	30	63	10.2	549,019	10,402	*tl*	100.0
CHL1s	13,099	30	63	10.2	549,019	10,402	*tl*	100.0
CHL6	16,869	30	65	10.8	817,604	13,170	*tl*	100.0
Hchl2	9,954	35	75	7.2	828,561	12,594	*tl*	100.0
CHL7	16,881	35	75	7.2	831,884	12,594	*tl*	100.0
CW2	5,354	35	63	4.9	884,289	14,664	*tl*	100.0
CU2	26,100	35	90	5.0	951,889	16,068	*tl*	100.0
gcut2	59,307	20	20	3.8	1,105,140	28,793	*tl*	100.0
gcut6	236,305	20	20	3.8	1,717,110	74,797	*tl*	100.0
gcut3	60,241	30	30	4.0	1,969,390	32,681	*tl*	100.0
gcut10	903,435	20	20	3.8	2,625,443	138,062	*tl*	100.0
CW3	5,689	40	96	4.6	2,837,862	33,224	*tl*	100.0
gcut4	60,942	50	50	4.0	2,963,221	35,176	*tl*	100.0
Hchl6s	61,040	22	60	7.2	4,730,229	44,593	*tl*	100.0
gcut7	238,974	30	30	3.0	4,908,322	100,800	*tl*	100.0
Hchl7s	63,112	40	90	8.0	5,722,617	46,491	*tl*	100.0
gcut8	245,758	50	50	3.9	16,329,925	136,524	*tl*	100.0
gcut11	955,389	30	30	3.8	32,997,962	400,070	*tl*	100.0
gcut12	970,744	50	50	3.9	66,415,467	489,428	*tl*	100.0

TABLE 2.4: Large-size instances.

2.4.1 Lower bound (feasible solution) computation

Computing a feasible solution is the first step of the solution procedure for the *PP-G2KP Model*, summarized in Algorithm 2. A possible fast way of computing a feasible solution is by the iterated greedy algorithm proposed by Dolatabadi et al. [2012]: given a random order of the items, the algorithm selects the first k items in the ordering whose cumulate profit would improve on the incumbent solution. The algorithm then tries to pack the selected items into the rectangular panel, according to a First Fit Decreasing strategy (see Coffman et al. [1980]); in case of success, the incumbent solution is updated (the attempt is not performed if the sum of the areas of the selected items is larger than the area of the panel). In our implementation, we allow 1 million of iterations after the last update of the incumbent solution.

Improved feasible solutions can be obtained by considering the optimal solution of the *PP-G2KP Model* with cut positions in the restricted position set Q_R, i.e., the *Restricted PP-G2KP model*. Example 1 tells us that the *Restricted PP-G2KP model* might not contain the optimal solution. However, this occurrence is rare in practice and, in any case, the optimal solution value of the *Restricted PP-G2KP model* is a valid lower bound LB on the optimal solution value.

In order to show the relative size of the *Restricted PP-G2KP model*, in Figure 2.8 we use performance profiles to depict the percentage of variables and plates of the *Restricted PP-G2KP model* with respect to variables and plates of the *Complete PP-G2KP Model*. In the horizontal axis the figure reports the percentage of variables (resp., plates), and in the vertical axis the percentage of instances for which the *Restricted PP-G2KP model* has no more than the corresponding percentage of variables (resp., plates). The continuous line is for the variables, the dashed line for the plates. We see that for 80% of the instances, the *Restricted PP-G2KP model* has at most 25% of the variables and 65% of the plates of the *Complete PP-G2KP Model*.

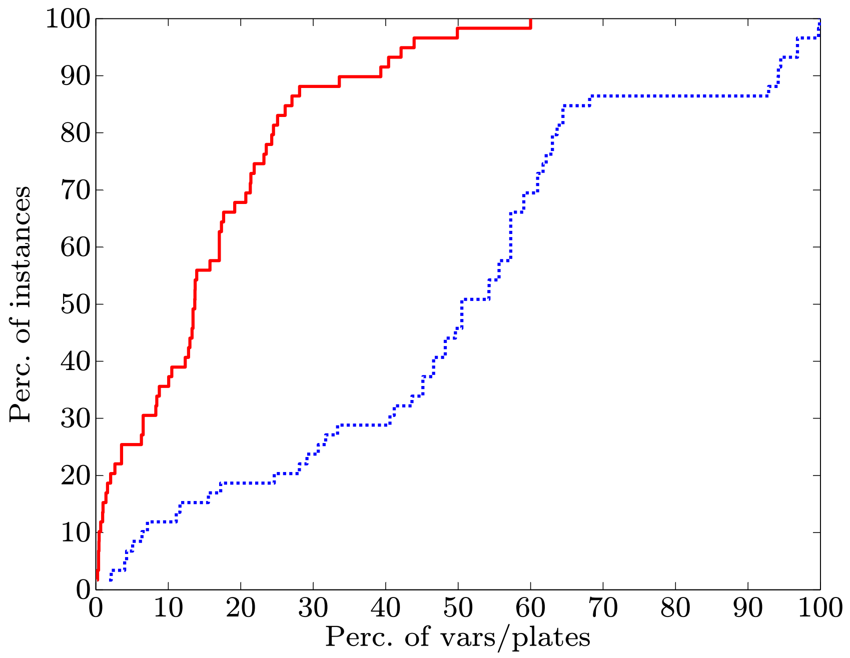


FIGURE 2.8: Variables and plates of the *Restricted PP-G2KP model* with respect to the *Complete PP-G2KP Model*.

Despite the large reduction in the number of variables and plates, solving the *Restricted PP-G2KP model* by using a MIP solver can be very time consuming, hence, we adapted the pricing procedure of Algorithm 2 to this case. We compute an initial feasible solution by means of the iterated greedy algorithm, we solve the linear programming relaxation of the *Restricted PP-G2KP model*, and we price the model variables, thus defining a *Restricted Priced PP-G2KP Model* containing only the variables that, by entering in base with value 1 or larger, could improve on the incumbent solution. Then, we solve the resulting *Restricted Priced PP-G2KP Model* by means of the CPLEX MIP solver.

In Figure 2.9 we use performance profiles to represent the gaps between the values of the greedy heuristic solution (LB_g) and of the *Restricted PP-G2KP model* (LB_R) optimal solution, and the value of the best solution we can compute. In the horizontal axis of the figure we report the percentage gap, computed as $100(best - LB)/best$, and in the vertical axis the percentage of instances for which the corresponding or a smaller gap is obtained. The dashed line denotes the greedy heuristic solution and the continuous line the *Restricted PP-G2KP model*, solved through the pricing procedure.

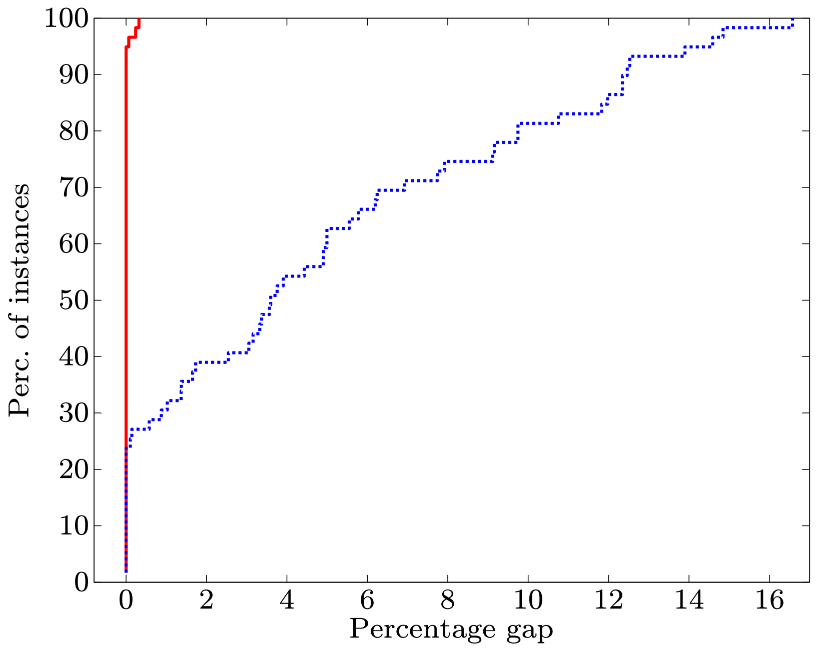


FIGURE 2.9: Value of the greedy heuristic solution (dashed line) and value of the *Restricted PP-G2KP model* optimal solution (continuous line), percentage gap with respect to the best computed solution.

The quality of the solutions obtained by solving the *Restricted PP-G2KP model* is very good: for all but three cases it coincides with the best solution we could compute. This suggests that, even though in principle the *Complete PP-G2KP Model* solution can have a profit that is at least 20% larger than the profit of the *Restricted PP-G2KP model* (see Remark 2.3), in many cases one could consider solving (exactly or heuristically) the latter, still obtaining very good solutions. Concerning the greedy heuristic, the largest gap with respect to the best computed solution does not exceed 17%.

Concerning the computational effort, the greedy heuristic takes few seconds, while solving the *Restricted PP-G2KP model* can be time consuming, even if the pricing procedure is used. More details on the computing times are reported next in Table 2.10.

2.4.2 Iterative Variable Pricing

Solving the linear programming relaxation of the *PP-G2KP Model* through the variable pricing procedure asks to iteratively add variables with positive reduced profit, and then re-optimize the linear program, as explained in Algorithm 2. The actual way variables are added has practical relevance, because it heavily impacts on the computing time and number of iterations of the procedure. On the one hand, one could add at each iteration all the variables having positive reduced profit, solving the linear programming relaxation in less iterations (but eventually solving large LPs). On the other hand, one could add a single variable to the linear program at each time, eventually performing more iterations.

We designed an optimized procedure for iteratively adding variables with positive reduced profit to the linear programming relaxation of the *PP-G2KP Model*. The procedure uses two parameters, namely, the maximum number n_{max} of variables added at each iteration, and a threshold $\bar{p}$ for the reduced profit of the variables to be added. At each iteration, the first n_{max} variables in the pool, having reduced profit larger than $\bar{p}$, are added to the linear programming relaxation of the *PP-G2KP Model*. If no variable with reduced profit larger than $\bar{p}$ exists, the first n_{max} variables with positive reduced profit are added. Variable number n_{max} is defined as the number of variables that have positive reduced profit at the first iteration, times a parameter α. Threshold $\bar{p}$ is defined as the sum of the profits of all the available items, times a parameter β. The values of parameters α and β were experimentally tuned, so as to minimize the cumulate computing time of the variable pricing procedure for the whole instance set. In Figure 2.10 we report three lines, one per value of α; each line depicts the average variable pricing time for different values of β. As result of these experiments, we choose the following values of the parameters: $\alpha = 0.20$ and $\beta = 0.25$.

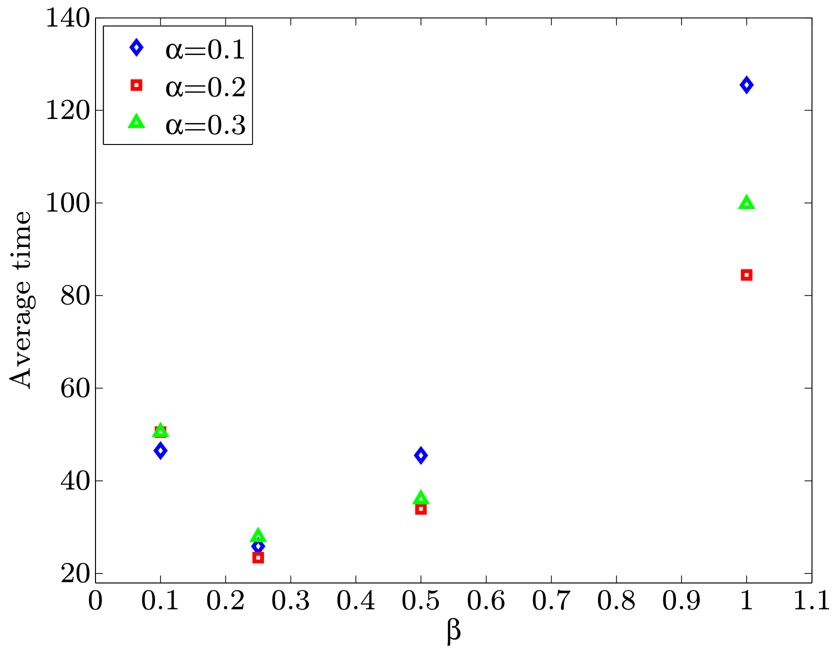


FIGURE 2.10: Linear programming relaxation solution time for different values of the α and β parameters.

2.4.3 Models size and reductions

In this section we discuss the size of the various models we work with, namely, the *Complete PP-G2KP Model*, the *Complete PP-G2KP Model* after applying the `Cut-Position` and `Redundant-Cut` reductions, and the *Priced PP-G2KP Model*.

We use performance profiles to represent the percentage of variables of each considered model with respect to the number of variables of the *Complete PP-G2KP Model* (reported in Tables 2.2, 2.3 and 2.4). Figure 2.11 reports on the horizontal axis the percentage of residual variables, and on the vertical axis the percentage of instances having no more than the specified percentage of variables. From right to left, the lines in the figure correspond to the application of the `Redundant-Cut` reduction, the `Cut-Position` reduction, and the two reductions together. The reduction strategies appear to be very effective for the *gcut* instances, where in several cases the number of variables is halved, while for the rest of the instance set the percentage of residual variables ranges between 72% and 96%. The leftmost line of Figure 2.11 depicts the percentage of residual variables of the *Priced PP-G2KP Model* (i.e., after the variable pricing procedures are applied). For approximately 80% of the instances, the percentage of residual variables in the *Priced PP-G2KP Model* is smaller than 40%. The benefit of such a reduction in the model size is discussed in the next section.

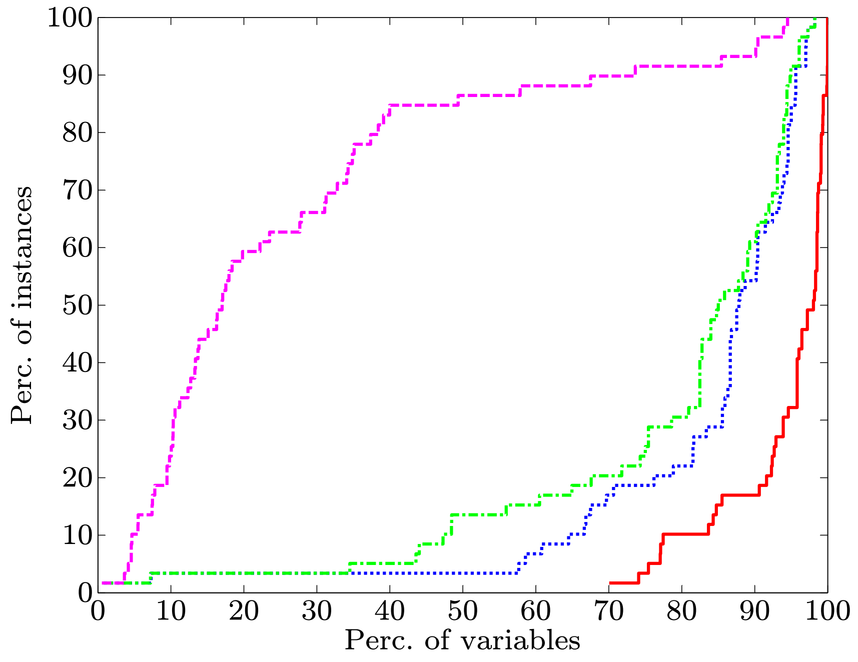


FIGURE 2.11: Percentage of residual variables of the *PP-G2KP Model* after the reductions. From right to left: variables after the `Redundant-Cut` reduction, the `Cut-Position` reduction, the two reductions together. Leftmost line: percentage of variables of the *Priced PP-G2KP Model*.

Concerning the number of plates, as anticipated only the `Cut-Position` reductions can affect it. In practice this happens for a small fraction of the instance set, where there is a large reduction: for 3% of the instances, the percentage of residual plates is no larger than 10%; for 90% of the instances, the number of plates is unchanged.

Table 2.5, 2.6 and 2.7 reports the percentage of residual variables and plates of the *PP-G2KP Model* after the reductions.

For each instance, the table reports the name (*name*), and the data of the *Complete PP-G2KP Model* after the `Redundant-Cut` reduction, the `Cut-Position` reduction, the two reductions together and finally of the *Restricted PP-G2KP model*. The percentage of computing time (with respect to the total t_{tot} in Tables 2.8, 2.9 and 2.10) is reported in column two ($\%t_{gen}$) for the *Complete PP-G2KP Model* with no reductions and in column ten for the *Restricted PP-G2KP model* . Then the table reports the number of variables after the different reductions (*vars(%)*) and the corresponding number of plates (*plates(%)*).

	Complete PP-G2KP Model	`Redundant-Cut`	`Cut-Position`		*PP-G2KP Model*		*Restricted PP-G2KP model*		
name	$t_{gen(\%)}$	*vars(%)*	*vars(%)*	*plates(%)*	*vars(%)*	*plates(%)*	*vars(%)*	*plates(%)*	$t_{gen(\%)}$
cgcut1	0.0	99.9	93.5	100.0	93.4	100.0	68.7	92.9	0.0
CHL5	0.7	100.0	87.8	100.0	87.8	100.0	61.9	94.2	0.0
Hchl8s	0.0	100.0	94.0	100.0	94.0	100.0	50.0	94.5	0.0
OF2	0.7	98.2	86.8	100.0	85.0	100.0	15.9	43.6	0.2
cgcut3	0.3	95.8	86.7	100.0	82.5	100.0	26.2	57.3	0.1
wang20	2.0	95.8	86.7	100.0	82.5	100.0	26.2	57.3	1.0
3	0.3	95.8	86.7	100.0	82.5	100.0	26.2	57.3	0.1
3s	2.5	95.8	86.7	100.0	82.5	100.0	26.2	57.3	1.0
W	2.0	95.8	86.7	100.0	82.5	100.0	26.2	57.3	1.0
OF1	2.3	99.1	90.2	100.0	89.3	100.0	23.9	63.7	0.9
gcut1	6.3	84.8	7.3	11.6	7.2	11.6	7.2	11.6	3.1
A1	0.5	96.5	87.5	100.0	84.0	100.0	21.2	48.2	0.2
A1s	3.1	96.5	87.5	100.0	84.0	100.0	21.2	48.2	1.2
cgcut2	0.4	98.5	94.6	100.0	93.1	100.0	17.3	50.5	0.3
2	0.4	98.5	94.6	100.0	93.1	100.0	17.3	50.5	0.2
2s	0.4	98.5	94.6	100.0	93.1	100.0	17.3	50.5	0.2
CHL2	0.1	97.2	85.6	100.0	82.8	100.0	16.1	45.1	0.0
CHL2s	0.2	97.2	85.6	100.0	82.8	100.0	16.1	45.1	0.1
A2	0.1	93.9	81.5	100.0	75.4	100.0	22.1	46.6	0.1
A2s	2.3	93.9	81.5	100.0	75.4	100.0	22.1	46.6	1.1

TABLE 2.5: Reductions of the small-size instances.

	Complete PP-G2KP Model	`Redundant-Cut`	`Cut-Position`		*PP-G2KP Model*		*Restricted PP-G2KP model*		
name	$t_{gen(\%)}$	*vars(%)*	*vars(%)*	*plates(%)*	*vars(%)*	*plates(%)*	*vars(%)*	*plates(%)*	$t_{gen(\%)}$
STS2	0.8	98.6	90.4	100.0	89.1	100.0	36.5	64.5	0.3
STS2s	1.1	98.6	90.4	100.0	89.1	100.0	36.5	64.5	0.5
Hchl9	0.4	98.8	91.5	100.0	90.2	100.0	40.8	68.2	0.2
A3	0.7	96.1	88.7	100.0	84.8	100.0	20.7	49.6	0.3
HH	0.0	99.0	97.1	100.0	96.1	100.0	6.3	29.2	0.0
A4	0.1	99.1	92.4	100.0	91.5	100.0	26.5	61.7	0.1
gcut5	0.0	92.3	69.7	96.6	67.6	96.6	2.6	5.1	0.0
okp1	0.4	100.0	98.2	100.0	98.2	100.0	29.5	99.7	0.2
okp3	0.0	99.9	85.9	100.0	85.9	100.0	42.1	94.2	0.0
okp4	0.4	100.0	94.4	100.0	94.4	100.0	40.2	99.9	0.1
okp2	0.0	100.0	90.4	100.0	90.4	100.0	43.9	96.8	0.0
CU1	3.4	92.7	81.6	100.0	74.3	100.0	9.4	30.7	1.1
STS4	0.4	98.3	95.6	100.0	94.0	100.0	19.0	55.6	0.1
STS4s	0.4	98.3	95.6	100.0	94.0	100.0	19.0	55.6	0.1
okp5	0.2	100.0	97.3	100.0	97.3	100.0	33.8	96.8	0.1
CW1	0.2	94.6	86.3	100.0	80.9	100.0	12.9	40.6	0.1
gcut9	1.1	92.4	66.9	92.8	65.0	92.8	1.3	2.1	0.0
A5	0.2	99.3	95.6	100.0	95.0	100.0	15.6	54.3	0.1

Table 2.6: Reductions of the medium-size instances.

	Complete PP-G2KP Model	`Redundant-Cut`	`Cut-Position`		*PP-G2KP Model*		*Restricted PP-G2KP model*		
name	$t_{gen(\%)}$	*vars(%)*	*vars(%)*	*plates(%)*	*vars(%)*	*plates(%)*	*vars(%)*	*plates(%)*	$t_{gen(\%)}$
Hchl4s	0.0	99.1	97.0	100.0	99.1	100.0	13.0	59.1	0.0
Hchl3s	0.0	99.1	97.0	100.0	99.1	100.0	13.0	59.1	0.0
CHL1	0.1	99.4	95.0	100.0	94.4	100.0	26.2	63.0	0.0
CHL1s	0.2	99.4	95.0	100.0	94.4	100.0	26.2	63.0	0.1
CHL6	0.2	99.3	95.5	100.0	94.9	100.0	24.3	62.2	0.1
Hchl2	0.1	98.6	93.3	100.0	92.0	100.0	23.4	61.0	0.0
CHL7	0.1	98.6	93.8	100.0	92.4	100.0	23.2	61.0	0.0
CW2	0.0	91.6	83.4	100.0	75.0	100.0	9.0	31.7	0.0
CU2	2.4	92.9	78.9	100.0	71.8	100.0	9.9	33.4	0.8
gcut2	1.8	77.0	64.5	100.0	48.5	100.0	2.6	11.1	0.4
gcut6	1.2	83.7	60.8	98.7	55.9	98.7	1.3	4.3	0.1
gcut3	2.6	77.1	66.7	100.0	47.3	100.0	3.7	17.2	0.7
gcut10	28.2	85.5	0.8	1.8	0.8	1.8	0.8	1.8	1.4
CW3	0.3	90.6	88.0	100.0	78.6	100.0	6.8	28.1	0.1
gcut4	1.8	84.3	76.2	100.0	60.5	100.0	6.3	24.6	0.6
Hchl6s	0.1	98.1	90.3	100.0	88.4	100.0	8.1	41.2	0.0
gcut7	1.4	74.1	58.6	99.7	44.0	99.7	1.3	7.1	0.2
Hchl7s	0.5	98.5	94.6	100.0	93.1	100.0	15.8	54.3	0.2
gcut8	2.1	77.5	70.7	100.0	48.5	100.0	2.4	15.5	0.5
gcut11	0.3	75.4	67.5	100.0	43.6	100.0	0.5	4.0	0.0
gcut12	1.4	70.1	57.7	100.0	34.5	100.0	0.7	6.4	0.2

TABLE 2.7: Reductions of the large-size instances.

2.4.4 Overall solution procedure

Finally, Tables 2.8, 2.9 and 2.10 report the results obtained by applying the proposed solution procedure to the small, medium and large-size instances, respectively. The tables follow the steps of the solution procedure:

- After the instance name, the tables report the overall computing time in seconds spent for the corresponding instance (t_{tot}). The procedure has three time limits of 1 hour, for solving the *Restricted PP-G2KP model*, for the variable pricing and for solving the *Priced PP-G2KP Model*, respectively. If a time limit is incurred, the corresponding value is marked with a $*$ in the tables.

- The first step of the solution procedure is to enumerate the problem variables (and plates) to be stored in the variable pool, and to apply the reductions (`Cut-Position` and `Redundant-Cut`). The percentage of computing time (with respect to the total t_{tot}) is reported in column three ($\%t_{gen}$).

- The solution procedure first asks for a feasible solution, that we compute by solving the *Restricted PP-G2KP model.* As anticipated, the *Restricted PP-G2KP model* is tackled by solving the corresponding MIP through CPLEX, after a pricing of the variables has been performed. The pricing asks for a feasible solution, which is obtained through the greedy heuristic. The percentage computing time $\%t_{LB}$ in column five accounts for all these computations. Instead, the percentage computing time for solving only the greedy heuristic is reported in column four $\%t_{HEU}$. Note that the aim of this step is producing a good quality feasible solution, thus, even if the step reaches the time limit, the correctness of the procedure is maintained. In column six we report the the percentage of residual variables used after the first pricing procedure is applied.

- The next step of the procedure is to solve the linear programming relaxation of the *PP-G2KP Model* through iterative pricing of the pool variables. The tables report, in column seven, the percentage of the overall computing time devoted to this task. No useful upper bound is available if this step reaches the time limit. Column eight (gap_{LP}) reports the percentage optimality gap at this step of the procedure, computed as $100\frac{UB_{LP}-LB}{UB_{LP}}$, where LB is the value of the feasible solution computed in the previous step, and UB_{LP} is the upper bound obtained by rounding down the value of the linear programming relaxation.

- After applying a final round of pricing, the procedure defines the *Priced PP-G2KP Model.* The *Priced PP-G2KP Model* is then tackled by the CPLEX MIP solver with a time limit of 1 hour. In column nine we report the percentage computing time, and in column ten the percentage optimality gap, i.e., $100\frac{UB_{best}-LB_{best}}{UB_{best}}$, where LB_{best} and UB_{best} are the values of the best feasible solution and upper bound computed during the overall procedure, respectively. In column eleven we report the final percentage of residual variables of the *Priced PP-G2KP Model* (i.e., after the variable pricing procedures are applied).

			Restricted PP-G2KP model			*linear relaxation*		*Priced PP-G2KP Model*		
name	t_{tot}	$t_{gen(\%)}$	$t_{HEU(\%)}$	$t_{LB(\%)}$	$col_{LB(\%)}$	$t_{LP(\%)}$	gap_{LP}	$t_{(\%)}$	gap	$col_{(\%)}$
cgcut1	0.6	0	95.2	96.8	27.1	1.6	0.2	1.6	0	36.1
CHL5	1.4	0.7	29.0	86.2	60.0	5.1	8.3	8	0	39.0
Hchl8s	3,638.20	0	0.0	1.1	49.9	0	22	98.9*	0.8	93.9
OF2	4.3	0.7	9.7	69.7	13.3	27.7	7.8	1.8	0	23.9
cgcut3	13.9	0.3	4.2	53.8	24.3	7.6	13.5	38.3	0	28.6
wang20	1.9	2.7	23.2	58.9	13.4	37.3	5	1.1	0	9.7
3	16.3	0.2	3.3	63.2	25.1	7	14.5	29.6	0	30.0
3s	1.9	2.6	23.6	58.1	13.4	38.7	5	0.5	0	9.6
W	1.8	2.2	32.6	43.1	10.5	53.6	3.6	1.1	0	9.8
OF1	2.1	2.3	27.1	34.6	10.1	58.9	1.5	4.2	0	15.6
gcut1	0.3	3.1	71.9	78.1	3.6	9.4	4.7	9.4	0	3.6
A1	9.8	0.4	5.4	84.8	20.7	14.5	15.1	0.3	0	12.9
A1s	1.5	3.4	32.4	33.1	3.6	62.8	0	0.7	0	7.4
cgcut2	10	0.5	10.1	78.6	17.1	20.7	12.5	0.2	0	13.4
2	9.2	0.4	8.5	84.4	17.1	14.9	12.5	0.3	0	11.3
2s	9.6	0.5	8.0	79.2	17.1	19.2	13	1	0	16.0
CHL2	65.7	0.1	0.8	10	13.7	3.1	6.8	86.8	0	32.0
CHL2s	23.9	0.2	2.1	23.7	13.7	6.3	6.6	69.8	0	33.3
A2	47.9	0.1	1.4	19.8	21.9	3.2	15.7	76.9	0	36.5
A2s	2.4	2.9	26.4	45	8.8	50.4	3.4	1.7	0	10.1

TABLE 2.8: Solution of the small-size instances.

			Restricted PP-G2KP model			*linear relaxation*		*Priced PP-G2KP Model*		
name	t_{tot}	$t_{gen(\%)}$	$t_{HEU(\%)}$	$t_{LB(\%)}$	$col_{LB(\%)}$	$t_{LP(\%)}$	gap_{LP}	$t_{(\%)}$	*gap*	$col_{(\%)}$
STS2	41.7	0.9	3.9	47.7	19.2	31.2	3	20.3	0	18.1
STS2s	29.1	1.2	2.5	52.1	13.7	45.7	1.8	1	0	7.5
Hchl9	121.6	0.4	0.6	75.4	40.4	21.4	9.9	2.7	0	13.4
A3	25.1	0.8	3.2	28.1	8.4	22.4	2.4	48.7	0	16.8
HH	3,654.00	0	0.0	1.5	6.3	0.1	12.1	98.5^*	4.4	67.3
A4	297.9	0.2	0.2	43.2	24.5	9	5.8	47.6	0	26.9
gcut5	558.3	0.1	0.1	0.5	2.0	1.4	11.7	98	0	53.1
okp1	319.4	0.5	0.5	64.2	28.1	31.8	11.2	3.5	0	37.7
okp3	7,429.20	0	0.0	48.4^*	42.1	3.1	11.4	48.4^*	8.4	85.3
okp4	685.5	0.4	0.1	32.5	39.3	66.6	5.6	0.4	0	10.0
okp2	7,667.20	0	0.0	46.9^*	43.9	6.1	15.9	46.9^*	15.9	90.4
CU1	7.4	3.7	11.3	11.7	1.6	83.9	0.1	0.8	0	5.0
STS4	205.5	0.4	0.5	66.8	17.4	31.3	5.4	1.6	0	9.7
STS4s	197.2	0.4	0.6	69.1	17.6	30.5	5.9	0.1	0	4.4
okp5	1,276.20	0.3	0.1	29.9	33.6	69.8	12.6	0	0	11.5
CW1	318.9	0.2	0.3	25.2	12.8	6.2	17.2	68.4	0	26.7
gcut9	20.5	4.2	1.5	2.3	0.7	76.1	1.9	17.4	0	30.8
A5	700.2	0.2	0.1	37.1	13.9	17.2	4.1	45.5	0	19.0

Table 2.9: Solution of the medium-size instances

			Restricted PP-G2KP model			*linear relaxation*		*Priced PP-G2KP Model*		
name	t_{tot}	$t_{gen(\%)}$	$t_{HEU(\%)}$	$t_{LB(\%)}$	$col_{LB(\%)}$	$t_{LP(\%)}$	gap_{LP}	$t_{(\%)}$	*gap*	$col_{(\%)}$
Hchl4s	7,602.50	0	0.0	47.3*	13.0	5.3	9	47.3*	9	75.3
Hchl3s	2,518.00	0	0.1	15.7	12.4	12.7	4.4	71.6	0	39.1
CHL1	4,577.60	0.1	0.0	33.3	26.1	7.5	11.9	59.2	0	49.7
CHL1s	1,520.00	0.2	0.1	59.9	23.5	22.8	4.1	17.1	0	17.0
CHL6	2,576.80	0.2	0.0	67.5	21.4	32.2	3.9	0.1	0	4.4
Hchl2	4,346.30	0.1	0.0	38.6	23.2	34.2	10	27	0	22.0
CHL7	3,801.30	0.1	0.0	51.5	21.3	43.3	5	5.1	0	18.6
CW2	3,932.40	0	0.0	7.6	8.3	0.9	11.9	91.5*	5.5	28.9
CU2	51.8	2.5	1.9	3.8	2.6	93.4	1.7	0.3	0	4.5
gcut2	17.6	4.7	1.7	2.6	0.9	78.9	1.5	13.9	0	10.7
gcut6	46.5	8.9	0.6	0.9	0.5	87.2	0.6	3.1	0	13.6
gcut3	34.9	4	0.9	2	1.0	89.9	1.2	4.1	0	5.6
gcut10	4.9	2.4	5.5	46.4	0.5	9.9	6.1	41.3	0	0.5
CW3	1,264.80	0.3	0.1	70.7	6.5	22.3	15.4	6.6	0	13.5
gcut4	181.7	2.1	0.3	5.9	1.4	71.8	1.1	20.2	0	7.6
Hchl6s	7,234.10	0.3	0.0	49.8*	6.5	50*	-	-	-	88.4
gcut7	117.1	7.3	0.2	0.5	0.4	72.5	1.3	19.7	0	8.2
Hchl7s	7,270.40	0.6	0.0	49.8*	15.8	49.5*	-	-	-	93.1
gcut8	666.4	3.6	0.1	0.1	0.4	95.1	0.2	1.2	0	3.3
gcut11	4,522.00	2.8	0.0	0.2	0.2	17.2	2.1	79.7*	2.1	10.2
gcut12	2,454.60	8	0.0	0.1	0.2	89.8	0.8	2.1	0	3.6

TABLE 2.10: Solution of the large-size instances

We compare the results obtained by applying the proposed solution procedure with the performance of the CPLEX MIP solver in solving the *Complete PP-G2KP Model*. Concerning small-size instances, two instance remains unsolved, ($Hchl8s$), but the final optimality gaps are reduced from 1.8% to 0.8%. Concerning medium-size instances, three additional instance are solved to optimality. For the tree instances which remain unsolved, the final optimality gaps are reduced. The computing times are reduced for almost all solved instances, in several cases of one or two orders of magnitude. Concerning the 21 large-size instances, they are all unsolved when the *Complete PP-G2KP Model* is tackled directly by the CPLEX MIP solver, and no feasible solution is produced by the solver, hence the optimality gap is 100%. By applying the solution procedure based on pricing, we could solve to optimality 16 of them, and for three out of five unsolved instances, the final optimality gap is at most 9%. For only two instances, namely, $Hchl6s$ and $Hchl7s$, the time limit is reached during the solution of the linear programming relaxation, and a valid upper bound is not provided.

In order to give a graphical representation of the performance of the two methods, namely, solving the *Complete PP-G2KP Model* directly by the CPLEX MIP solver and applying the proposed solution procedure, we report a performance profile in Figure 2.12. For each instance we compute a normalized time τ as the ratio of the computing time of the considered solution method over the minimum computing time for solving the instance to optimality. For each value of τ in the horizontal axis, the vertical axis reports the percentage of the instances for which the corresponding method spent at most τ times the computing time of the fastest method. The proposed solution procedure based on pricing (dashed line) has a much better performance than solving the *Complete PP-G2KP Model* directly with the CPLEX MIP solver (continuous line). The performance profile of both methods reaches an horizontal line denoting the percentage of instances that could be solved within time limit.

We conclude this section by commenting on the quality of the solutions obtained by solving the *Restricted PP-G2KP model* to optimality: quite often this is the optimal solution; among 50 instances solved to optimality, the solution of the *Restricted PP-G2KP model* is optimal in 47 cases. For three instances only, the incumbent is improved, namely: A5, CHL1 and CHL7. For the remaining instances, the value of the lower bound LB is not improved when solving the *Priced PP-G2KP Model*. In other words, solving the *Priced PP-G2KP Model* is quite often only needed to certify the optimality of the incumbent solution.

2.4.5 Comparison with state-of-the-art approaches

The most recent and effective combinatorial algorithm for the 2GKP is the A1 algorithm by Dolatabadi et al. [2012], which embeds a recursive procedure to enumerate all possible packings of the items within a given rectangular panel. Algorithm A1 needs

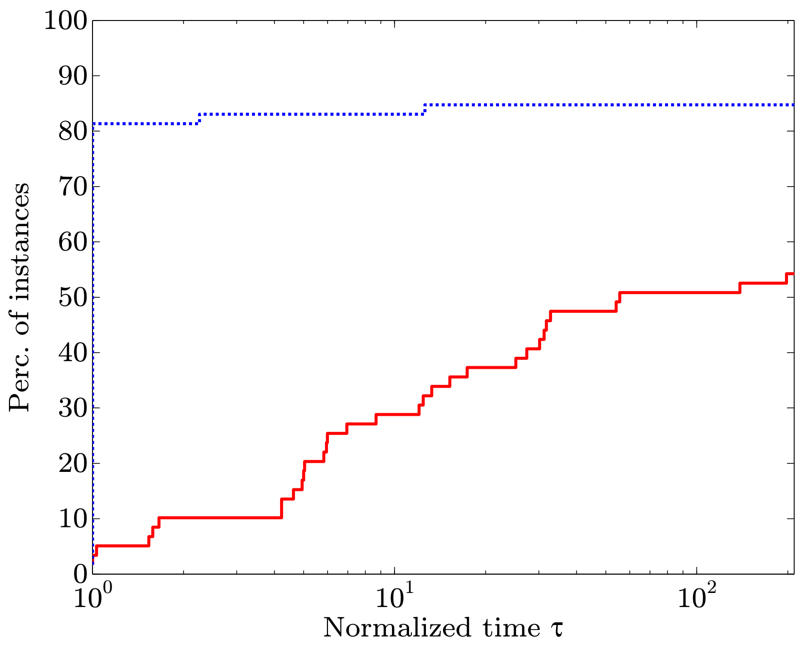


FIGURE 2.12: Percentage of solved instances with respect to the normalized computing time. *Complete PP-G2KP Model* (continuous line) and *Priced PP-G2KP Model* (dashed line).

as input an upper bound UB on the maximum profit which can be obtained from the original rectangular panel, which is set to $UB = \min\{U_{kp}, U_{unc}\}$, where U_{kp} is the optimal solution of the associated non-guillotine problem, and U_{unc} is the optimal solution value of the associated unconstrained two-dimensional problem (i.e., the problem in which infinite copies of each item are available). Algorithm A1 also needs a lower bound which is computed by running the greedy heuristic described in Section 2.4.1. We ran an implementation of the algorithm received from the authors of Dolatabadi et al. [2012], which computes internally the UB, while the lower bound is obtained by running for 60 seconds (as in Dolatabadi et al. [2012]) our implementation of the greedy heuristic.

In our experiments, A1 could solve all the 59 instances we considered, with an average computing time of 62.6 seconds, with a minimum of 60 and a maximum of 99.21 seconds. It has a strictly better performance (in terms of computing time or optimality of the solution) in 31 cases and a worse performance (in terms of computing time) in 28 cases. Most of the computing time (60 seconds) is spend by the initial greedy heuristic, however, the quality of this information is crucial for the performance of the algorithm. If the initial greedy heuristic is removed and a trivial lower bound of value 0 is used, the algorithm runs into time limit for 14 out of the 59 considered instances.

In addition, as reported in Dolatabadi et al. [2012], A1 can also solve many of the APT problems from Alvarez-Valdes et al. [2002]. Because of their structure, where very small items need to be packed into large rectangular panels, these instances are intractable for the *PP-G2KP Model*, whose number of variables would be too large in that case.

Concerning the modeling of guillotine restrictions through MIPs, the only alternative framework we are aware of was proposed by Ben Messaoud et al. [2008], and applied to the GSPP. The model was tested on instances having 5 items, to be packed into strips of length $L = 300$. Items lengths l were randomly generated with uniform distribution in $[\alpha L, \beta L]$, with $(\alpha, \beta) \in \{(0.1, 0.5), (0.3, 0.7)\}$. Three shape classes were considered for the items: wide items, long items and almost square items. Wide items, with $w \in [1.2l,$ $5l]$, were generated with probability A; almost square items, with $w \in [0.8l,\ 1.2l]$, where generated with probability B; and long items, with $w \in [0.1l,\ 0.8l]$, were generated with residual probability. Ben Messaoud et al. [2008] considered the following set of values for A and B: $(A, B) \in \{(1/2, 1/4), (1/4, 1/2), (1/4, 1/4), (1/3, 1/3)\}$.

In Table 2.11, we report on some experiments we performed by generating instances with these features. We implemented the *PPS-GSPP Model* (2.16)–(2.21) and computed the upper bound W on the optimal solution by using a greedy first-fit algorithm for the two-stage version of the problem. For each combination of the A, B, α and β parameters, in the table we report the average computing time for solving a set of 10 homogeneous instances: t_{FMT} is the time need by our approach for solving the instances we generated, while t_{BCE} is the time reported in Ben Messaoud et al. [2008] for solving instances generated with the same features. Our approach is 2 orders of magnitude faster in solving 6 of the 8 problem classes, and still faster on the remaining 2 classes. This testifies the better performance of our approach, although Ben Messaoud et al. [2008] used an older version of the CPLEX MIP solver and a slightly slower computer. We are confident that our results could still be improved by refining the value of the initial upper bound W.

A, B	$[\alpha, \beta]$			
	[0.1,0.5]		[0.3,0.7]	
	t_FMT	t_BCE	t_FMT	t_BCE
1/2,1/4	37.0	5,174	13.9	6,218
1/4,1/2	74.0	5,511	10.4	3,299
1/4,1/4	50.1	688	22.5	5,993
1/3,1/3	58.8	80	18.6	5,169

Table 2.11: Average computing times for solving homogeneous classes of five items Strip Packing instances.

2.4.6 Relevance of guillotine cuts

Guillotine cuts potentially allow a better use of the panels and can reduce the waste of raw material when compared with more constrained cut paradigms. General guillotine cuts, being less constrained than restricted guillotine cuts, potentially are the most efficient in this respect.

In our computational experiments we considered 59 *general* (unrestricted) guillotine G2KP from the literature. We tackled the same instances with the model described in Furini and Malaguti [2015], after removing all limitations on the number of stages, i.e., we solved the *restricted* G2KP. For 4 out of 56 instances we could solve to optimality for the latter problem, the optimal solution of the *general* G2KP has larger profit than the solution of the *restricted* G2KP. The profit increase is between 0.07% and 0.45%. Considering the overall set of 59 instances, instead, there is a profit increase with respect to the corresponding two-stage optimal solution in 48 cases, with an overall increase of 2.05%; and a profit increase with respect to the corresponding three-stage optimal solution in 46 cases, with an overall increase of 1.17%.

In Figure 2.5 of Section 2.2.1, we depicted an instance for which the profit of the optimal solution of the *general* G2KP is 20% larger than the profit of the optimal solution obtained by using restricted guillotine cuts or two-stage cuts.

These examples shows that, even though in general the profit increase obtained by allowing general instead of restricted guillotine cuts is limited (at least for the considered set of instances), there exist cases in which it can be quite large. On the other hand, the profit increase obtained by allowing general guillotine cuts instead of two or three-stage cuts is often substantial for the considered set of instances.

As a final consideration, the practical suitability of general (instead of restricted) guillotine cuts depends on the cutting technology. When in a general pattern the height (or width) of a strip is determined by more than just one element, there might be precision problems with the size of the elements. Problems occur when it is possible to cut off from the raw material a waste part in a precise way, but the width of the cut and therefore the size of the remaining part are not known with full certainty in advance. In this case, it is preferable avoiding cuts from which a part is obtained that needs to be further cut into multiple elements without any remaining waste.

2.5 Conclusions

In this chapter we proposed a way of modeling (general) guillotine cuts in Mixed Integer Linear Programs, without limiting the number of stages, nor imposing the cuts to be restricted. We concentrated on the Guillotine Two-Dimensional Knapsack Problem (G2KP), and we discussed extensions of the approach to Guillotine Two-Dimensional Cutting Stock and Guillotine Strip Packing problems. As our framework, based on the concepts of *cuts* and residual *plates*, can lead to a very large (pseudo-polynomial) number of variables, we proposed effective procedures for generating, managing and solving the obtained models.

Specifically, we devised a solution procedure based on the computation of a feasible solution of very good quality, which is obtained by restricting the modeling to consider only cuts that coincide with the size of some item. By exploiting dual information, we then perform a pricing of the variables which allows us to define smaller-size models while preserving the optimality of the solutions.

We reported extensive computational experiments, where the approach we proposed solved to optimality several benchmark instances from the literature. Compared with the state-of the art combinatorial approach for the G2KP, the proposed approach had a satisfactory performance, and it outperformed the only alternative framework based on Mixed-Integer Programming we are aware of.

Chapter 3

Mid-Term Hydro Scheduling Problem

[1]

We present a Branch-and-Cut algorithm for a class of nonlinear chance-constrained mathematical optimization problems with a finite number of scenarios. This class corresponds to the problems that can be reformulated as deterministic convex mixed-integer nonlinear programming problems, but the size of the reformulation is large and quickly becomes impractical as the number of scenarios grows. The Branch-and-Cut algorithm is based on an implicit Benders decomposition scheme, where we generate cutting planes as outer approximation cuts from the projection of the feasible region on suitable subspaces. The size of the master problem in our scheme is much smaller than the deterministic reformulation of the chance-constrained problem. We apply the Branch-and-Cut algorithm to the mid-term hydro scheduling problem, for which we propose a chance-constrained formulation. A computational study using data from ten hydroplants in Greece shows that the proposed methodology solves instances orders of magnitude faster than applying a general-purpose solver for convex mixed-integer nonlinear programming problems to the deterministic reformulation, and scales much better with the number of scenarios. Our numerical experiments show that introducing a small amount of flexibility in the formulation,by allowing constraints to be violated with a joint probability $\leq 5\%$, increases the expected profit by 6.1%.

3.1 Introduction

Mathematical programming is an invaluable tool for optimal decision-making that was initially developed in a deterministic setting. However, early studies on problems with

[1]This chapter is based on Lodi et al. [2016]

probabilistic (i.e., nondeterministic) constraints have appeared since the late 50s, see, e.g., Charnes and Cooper [1959], Charnes et al. [1958], Prekopa [1970]. In a problem with probabilistic constraints, the formulation involves a (vector-valued) random variable that parametrizes the feasible region of the problem; the decision-maker specifies a probability α, and the solution to the problem must maximize a given objective function subject to being inside the feasible region for a set of realizations of the random variable that occurs with probability at least $1-\alpha$. The interpretation is that a solution that does not belong to the feasible region is undesirable, and we want this event to happen with small probability α. This type of problem is called a *chance-constrained mathematical programming* problem in the literature (see, e.g., Charnes et al. [1958]).

Without loss of generality, a chance-constrained mathematical program can be expressed as

$$\max\{cx : \Pr(x \in C_x(w)) \geq 1 - \alpha, x \in X\}, \tag{CCP}$$

where w is a random variable, $C_x(w)$ is a set that depends on the realization of w (the set of probabilistic constraints), and X is a set that is described by deterministic constraints Prekopa [1970]. We use the subscript C_x to emphasize the fact that, given w, $C_x(w)$ is described in terms of the x variables only; this notation will be useful in subsequent parts of the chapter. A considerable simplification of the problem is that in which $C_x(w)$ is described by a set of constraints and $\Pr(x \in C_x(w))$ takes into account the violation of constraints one at a time, instead of considering the joint probability of $x \in C_x(w)$, which is more difficult. Chance-constrained mathematical programming problems find applications in many different contexts, see, e.g., Tanner et al. [2008], Watanabe and Ellis [1993]. The formulation (CCP) allows for two-stage problems with recourse actions, because the sets $C_x(w)$ can be the projection of higher-dimensional sets. This chapter discusses the case where recourse actions are allowed and we are interested in the joint probability of $x \in C_x(w)$.

If uncertainty affects only the right-hand side values of the system of inequalities that defines the feasible region, under certain assumptions it is possible to derive a tractable reformulation of the problem (see, e.g., Charnes and Cooper [1963], Lejeune [2012]). A more general case is considered when the uncertainty can affect all parts of the system of inequalities describing $C_x(w)$. In this case,

- if the sample space, denoted as Ω, is discrete and finite, and in particular $\Omega = \{w^i : i = 1, \dots, k\}$, and
- if all the $C_x(w^i)$'s are polyhedra sharing the same recession cone,

then, (CCP) can be reformulated as a deterministic mathematical program with integer variables, following a result in Jeroslow [1987]. This is accomplished by defining a problem with all the constraints of each of the $C_x(w^i)$, and introducing an indicator variable z_i for each w^i to activate/deactivate the corresponding constraints, see, e.g., Shen et al. [2010]. We should note that the assumption of discrete and finite sample space, while restrictive, includes a large number of practically relevant situations: typically, forecasts of future events cannot be too detailed and a general distribution can be truncated and discretized if necessary. Furthermore, even in the case that discretization and truncation cannot be applied, one can typically obtain good solutions and approximation bounds for a problem that requires general distributions via sample-average approximation Luedtke and Ahmed [2008]. From now on, we indeed assume $\Omega = \{w^i : i = 1, \dots, k\}$.

Unsurprisingly, the size of the problems obtained with the indicator-variable reformulation is unmanageable in most practically relevant situations, and moreover, the relaxations of mathematical programs with this type of indicator variables tend to be very weak, leading to poor performance of solution methods (see, e.g., Bonami et al. [2015]). However, under relatively mild assumptions it is possible to perform implicit solution of the reformulated problem Luedtke [2014]. The idea is to keep the indicator variables, but avoid the classical on/off reformulation of the constraints that involves them. Then, if cut separation routines for the set $C_x(w)$ are available, a Branch-and-Cut algorithm Padberg and Rinaldi [1991] can be applied to the problem $\max_{x \in X} cx$, augmented with the indicator variables for the sets $C_x(w)$ and a constraint to ensure that the scenarios for which the indicator variables are on occur with probability at least $1 - \alpha$. This problem is called a master problem. Whenever the solution of the master problem $\hat{x}$ does not satisfy the chance constraint $\Pr(\hat{x} \in C_x(w)) \geq 1 - \alpha$, cuts are generated for the sets $C_x(w^i)$ for which the corresponding indicator variable z_i is active, but $\hat{x} \notin C_x(w^i)$. The cuts are then added to the master problem. This basic idea yields an exact algorithm for the original chance-constrained mathematical program, and it has been successfully applied to different types of problem Liu et al. [2014], Luedtke [2014]. However, the literature mainly focuses on the case where all of the constraints are linear and all the original variables are continuous. While there are a few studies on linear problems with integer variables and certain classes of integer two-stage problems, see, e.g., Gade et al. [2014], Song et al. [2014], they are limited to specific problem structures, thus, the methods proposed cannot be applied in general. The classical decomposition approach for two-stage nonlinear problems is generalized Benders decomposition Geoffrion [1972], but it has the drawback of requiring separability and/or knowledge of the problem structure to be practically viable; for these reasons, to the best of our knowledge it has not been embedded in an automated, general-purpose (i.e., problem-independent) decomposition scheme for this class of problems so far.

In this chapter we consider the case where each set $C_x(w^i)$ is nonlinear convex, and propose a finitely convergent Branch-and-Cut algorithm. The cutting planes that we generate can be obtained as outer approximation cuts Duran and Grossmann [1986] and are therefore linear, as opposed to the generalized Benders cuts of Geoffrion [1972], which can be nonlinear in general. We show that our cuts are a linearization of generalized Benders cuts from a particular choice of dual variables, but our cut generation algorithm is much simpler than the generalized Benders procedure: it has fewer assumptions, in particular it does not require separability of the first and second stage variables or knowledge of the gradients, and it can be automated. While our main focus and computational testing is for the continuous convex case, our algorithm is finitely convergent also in the case where each $C_x(w^i)$ is a mixed-integer set with a convex continuous relaxation. The main application studied in this chapter is the scheduling of a hydro valley in a mid-term horizon Baslis and Bakirtzis [2011], Carpentier et al. [2012], Kelman [1998]. We propose a chance-constrained quantile optimization model for this problem that is equivalent to the minimization of the Value-at-Risk (see, e.g., McNeil et al. [2015]), and perform a case study on the scheduling of a 10-plant hydro valley in Greece, using a mix of historical and realistically generated data. Computational experiments show that our approach is able to solve large instances obtained from data of Baslis and Bakirtzis [2011] very effectively. Furthermore, in our case study introducing a moderate amount of flexibility, i.e., allowing some of the constraints to be violated with probability $\leq \alpha = 0.05$, increases the expected profit by approximately 6%. If the maximum probability of violating the constraints is relaxed to $\alpha = 0.1$ we can obtain an additional 1% increase in the expected profit, but there are no further gains increasing α beyond 0.1.

This chapter is organized as follows. In the rest of this section, we briefly introduce the main driving application for our paper, namely, the mid-term hydro scheduling problem, and discuss our choice for the robustness model of the objective function. Section 3.2 describes the decomposition approach with the associated Branch-and-Cut algorithm, discussing separating inequalities and their properties. Section 3.3 formalizes a mathematical model for the hydro scheduling problem. Section 3.4 contains a computational evaluation of several algorithms on instances of increasing difficulty derived from our case study, and discusses the numerical results. Finally, some conclusions are drawn in Section 3.5.

3.1.1 Mid-term hydro scheduling

A central problem in power generation systems is that of optimally planning resource utilization in the mid and long term and in the presence of uncertainty. Hydro power production networks usually consist of several reservoir systems, often interconnected, which are operated on a yearly basis: it is common to have seasonal cycles for demand and inflows, which can be out of phase by a few months, i.e., inflow peaks typically precede demand peaks.

The mid-term hydro scheduling problem refers to the problem of planning production over a period of several months. To be effective, such planning must take into account uncertainty affecting rainfall and energy demand, as well as the complex and nonlinear power production functions. A commonly used approach in practice is to rely on deterministic optimization tools and on the experience of domain experts to deal with the uncertainty, because of the sheer difficulty of incorporating uncertainty into the model. Many deterministic approaches can be found in the literature, see, e.g., Carneiro et al. [1990]. More recently, methodologies that can take into account the uncertainty in the model have appeared, see, e.g., Baslis and Bakirtzis [2011], Carpentier et al. [2012], Kelman [1998]. We are not aware of previous work that employs a chance-constrained formulation for the mid-term hydro scheduling problem, although there has been work on the related unit commitment problem, see, e.g., van Ackooij [2014], Wang et al. [2012]. Even in the case of unit commitment, chance-constrained optimization approaches are the least commonly used in the literature, due to their difficulty [Tahanan et al., 2015, Sect. 4.4].

The problem studied in this chapter can be described as follows: there are n hydroplants, each one associated with a reservoir, with an initial amount of water $q_h, h = 1, \dots, n$. The water in each reservoir can be used to obtain energy through the power plant. Our goal is to define a mid-term production plan, that is, how much water to release in each period from each reservoir, over a time horizon of several months, in order to maximize a profit function. The profit depends on the amount of energy obtained and on the market price, assuming that the amount of energy sold influences the final price, i.e., the generating company is a price maker. In each time period, the total quantity of water in the reservoirs must satisfy some lower and upper bounds. All the water that is not released in period t is available at $t+1$, in addition to the natural water inflow from rivers, precipitations and seasonal snow melting. The definition of a production plan faces two sources of uncertainty, namely: the natural water inflow, and the energy price on the market. We model the uncertainty by defining a finite number of inflow and energy market scenarios, each one with an associated probability of realization. The assumption of a finite number of scenarios is typical and widely accepted in the energy scheduling literature.

3.1.2 Choice of the objective function

When the problem takes into account a long time span, the decision-maker is typically interested in the optimal present-time (i.e., first stage) decisions: future decisions can be adjusted depending on the evolution of the market and the context. Consequently, we consider a problem formulation with recourse, where in our case, the recourse actions are simply all the decision taken at time periods $t > 1$.

It is important to remark that the profit for the generating company is a function of the first-stage decisions and the scenario, i.e., the realization of w. Thus, in order to formulate the objective function of the problem, we must decide what measure of profit we are interested in. Widely used choices when optimizing an uncertain profit are the expected profit and the worst-case profit. Our approach draws from the financial risk management literature: we use a measure of profit related to the well-known Value-at-Risk McNeil et al. [2015], which allows the decision-maker to determine the trade-off between risk and returns. In particular, given $0 \leq \alpha < 1$, our objective function is the maximization of the α-quantile of the profit. We now show how this relates to Value-at-Risk.

Let $\varphi(x, w^i)$ be the profit that can be obtained in scenario w^i with first-stage decision variables x; notice that given x and w^i, the value of $\varphi(x, w^i)$ can be computed by solving a deterministic optimization problem. Define the random variable $\varphi_x : \Omega \to \mathbb{R}$, $\varphi_x(w) = \varphi(x, w)$. Since φ_x is a random variable that measures the profit, we define the loss as $L_x = -\varphi_x$. The α-Value-at-Risk is defined as

$$\mathrm{VaR}_\alpha(L_x) = \inf\{\ell \in \mathbb{R} : \Pr(L_x > \ell) \leq 1 - \alpha\}.$$

Thus, we can write

$$\begin{aligned}
\min_x \mathrm{VaR}_{1-\alpha}(L_x) &= \min_x \inf\{\ell \in \mathbb{R} : \Pr(L_x > \ell) \leq \alpha\} \\
&= \min_x \inf\{\ell \in \mathbb{R} : \Pr(-\varphi_x > \ell) \leq \alpha\} \\
&= \max_x \sup\{-\ell \in \mathbb{R} : \Pr(-\varphi_x > \ell) \leq \alpha\} \\
&= \max_x \sup\{-\ell \in \mathbb{R} : \Pr(\varphi_x < -\ell) \leq \alpha\} \\
&= \max_x \sup\{q \in \mathbb{R} : \Pr(\varphi_x < q) \leq \alpha\} \\
&= \max_x \sup\{q \in \mathbb{R} : \Pr(\varphi_x \geq q) \geq 1 - \alpha\} \\
&= \max_x Q_\alpha(\varphi_x),
\end{aligned}$$

where Q_α is the α-quantile. The equation above shows that maximizing the α-quantile of the profit is equivalent to minimizing the $(1-\alpha)$-Value-at-Risk of the loss.

3.2 Decomposition algorithm for nonlinear (CCP)

When the sets $C_x(w^i)$ in (CCP) are polyhedra with the same recession cone, it is easy to write a *mixed-integer linear programming* problem (MILP) reformulation of (CCP). The MILP model naturally leads to Benders decomposition algorithm, and this is the approach followed, e.g., in Luedtke [2014]. We now introduce this MILP model for the case where each $C_x(w^i)$ is a polyhedron, to explain the basic ideas and notation before transitioning to the case where each $C_x(w^i)$ is instead a general convex set, which is the focus of this chapter. In terms of notation, we use x to denote the decision variables of (CCP), y^i to denote the recourse variables for scenario w^i, and z to denote binary variables with the property that $z_i = 0 \Rightarrow x \in C_x(w^i)$. Let $p_i = \Pr(w = w^i)$, $X = \{x : Ax \leq b\}$, $C_x(w^i) = \{x : \exists y^i \; A^i x + H^i y^i \leq b^i\}$. Then, (CCP) can be formulated as follows:

$$\begin{array}{llllllllll}
\max & cx & & & & & & & & \\
\text{s.t.:} & Ax & & & & & & & \leq & b \\
& A^1x & + & H^1y^1 & & & & & \leq & b^1 + M\mathbb{1}z_1 \\
& A^2x & & & + & H^2y^2 & & & \leq & b^2 + M\mathbb{1}z_2 \\
& \vdots & & & & & & & \vdots & \\
& A^kx & & & & & + & H^ky^k & \leq & b^k + M\mathbb{1}z_k \\
& & & p_1z_1 & + & p_2z_2 & + \dots + & p_kz_k & \leq & \alpha \\
& & & z_1, & & z_2, & \dots & z_k & \in & \{0,1\}.
\end{array} \tag{3.1}$$

In this formulation, $\mathbb{1}$ is a vector of ones and M is a large enough constant that is able to deactivate each constraint i in which $z_i = 1$. The joint chance constraint $\sum_{i=1}^k p_i z_i \leq \alpha$ ensures that the probability associated with unsatisfied scenarios is smaller than α. The formulation (3.1) is a two-stage problem with recourse where there is no objective function contribution associated with the recourse variables, therefore the second-stage problems are feasibility problems. It is well known that the case where the second-stage decisions affect the objective function can be reduced to the feasibility case by means of an additional first-stage variable for each scenario to represent the corresponding objective function contribution.

This chapter studies the case where $C_x(w^i)$ is a general convex set, described as $C_x(w^i) = \{x : \exists y^i \; g^i_j(x, y^i) \leq 0, j = 1, \dots, m_i\}$. For all i, we write the vector function $g^i(x, y^i) = (g^i_1(x, y^i), \dots, g^i_{m_i}(x, y^i))^T$. For ease of notation we keep the assumption that $X = \{x : Ax \leq b\}$, but this does not affect our development and the generalization to the case where X is a general convex set is straightforward. If all the $C_x(w^i)$ have the same recession cone, we can write a MINLP model for (CCP) as follows:

$$\begin{array}{llllllllll}
\max & cx & & & & & & & & \\
\text{s.t.:} & Ax & & & & & & & \leq & b \\
& g^1(x, & & y^1) & & & & & \leq & M\mathbb{1}z_1 \\
& g^2(x, & & & & y^2) & & & \leq & M\mathbb{1}z_2 \\
& \vdots & & & & & & & \vdots & \\
& g^k(x, & & & & & & y^k) & \leq & M\mathbb{1}z_k \\
& & & p_1z_1 & + & p_2z_2 & + \dots + & p_kz_k & \leq & \alpha \\
& & & z_1, & & z_2, & \dots & z_k & \in & \{0,1\}.
\end{array} \tag{3.2}$$

Assuming the functions g^i_j are convex, (3.2) is a convex MINLP in the sense that it has a convex continuous relaxation.

3.2.1 Overview of the approach

Solving directly the MINLP model (3.2) can be impractical, therefore we follow a decomposition approach whereby we define a *master problem* with the constraints defining $x \in X$, and k scenario subproblems, one for each scenario, involving scenario-dependent constraints. Let $C_{x,y}(w^i)$ be the feasible region of a scenario, and define $C_x(w^i) = \text{Proj}_x C_{x,y}(w^i)$. So, $\hat{x}$ is feasible for scenario i if $\hat{x} \in C_x(w^i)$. The basic idea we exploit is to generate solutions for the master, and if they are not feasible for enough scenarios to satisfy the joint chance-constraint, we cut them off. This is essentially a Benders decomposition approach applied to (3.2). In the linear case (3.1), the solution to the master problem can be cut off by means of textbook Benders cuts. In the nonlinear case (3.2), we can use generalized Benders cuts. This chapter advocates a particular choice of outer approximation cuts, that are linearizations of Benders cuts and present several advantages: this will be the subject of Section 3.2.2; the relationship with generalized Benders decomposition Geoffrion [1972] is discussed in Section 3.2.4.

Instead of applying a pure Benders decomposition approach to (3.2), we use a Branch-and-Cut approach adapted from Luedtke [2014], where the linear case is considered and therefore applies to (3.1) rather than (3.2). However, the steps of the algorithm remain the same, as this is essentially implicit Benders decomposition: we do not solve the master problem to (integral) optimality, but apply Branch-and-Cut and separate Benders cuts at every node with an integral solution. The algorithm uses a separation routine for the scenario subproblems, combined with the variables z. A basic version of the algorithm is given by Algorithm 3.

Algorithm 3: `Decomposition Algorithm`

```
1  Define a master problem of the form
```

$$\left.\begin{array}{lrcl} \max & cx & & \\ \text{s.t.:} & Ax & \leq & b \\ & \sum_{i=1}^{k} p_i z_i & \leq & \alpha \\ & z & \in & \{0,1\}^k \end{array}\right\} \qquad (3.3)$$

```
2  repeat
3      Perform Branch and Bound on (3.3);
4      At every node of the tree with solution $(\hat{x}, \hat{z}), \hat{z} \in \{0,1\}^k$, do the following: for
       $i = 1, \dots, k$ do
5          if $\hat{z}_i = 0$ and $\hat{x} \notin C_x(w^i)$ then
6              separate $\hat{x}$ from $C_x(w^i)$ via an inequality $\gamma x \leq \beta_i$;
7              add inequality $\gamma x \leq \beta_i + M z_i$ to the master problem (3.3);
8          end
9      end
10 until no more nodes to be explored;
```

It is not difficult to see that this algorithm can be applied even if the sets $C_x(w^i)$ are nonlinear provided that we have access to a separation routine, although termination is in general not guaranteed. We remark that we could employ a *nonlinear* separating inequality rather than a hyperplane in step 6 of Algorithm 3, as is done in generalized Benders decomposition Geoffrion [1972]. However, linear inequalities have several computational advantages, and allow for an easy lifting procedure of the coefficients on the z variables following Luedtke [2014]. We will revisit this topic in Section 3.2.4 from a theoretical point of view, whereas a discussion of lifting on the z variables is given in Section 3.4.1; notice that lifting does not affect the general scheme of the algorithm.

Algorithm 3 has some similarities with the LP/NLP-BB approach of Abhishek et al. [2010] and the Hybrid approach of Bonami et al. [2008], in the sense that all these methodologies involve a Branch-and-Cut algorithm where additional outer approximation inequalities are computed at nodes of the tree with integer solution. However, a fundamental difference exists: the algorithms of Abhishek et al. [2010], Bonami et al. [2008] as applied to (3.2) would work with a relaxation of the feasible region that includes all the decision variables, using NLP subproblems to construct outer approximation cuts fixing the integer variables. In the case of Algorithm 3, the master contains a subset of decision variables and is not aware of the recourse variables y^i. Therefore, we work on a projection of the feasible region of (3.2), and some integer and continuous variables (z and x) are fixed to obtain outer approximation cuts. It can be easily seen that the sequence of points generated by the algorithm is not necessarily the same.

3.2.2 Separation algorithm

In this section we provide a separation algorithm for step 6 of Algorithm 3 that applies to chance-constrained mathematical programming problems with recourse variables within a convex feasible region. For ease of notation, we drop the dependence on w and refer to $C_{x,y}, C_x$ as the subproblems associated with a particular realization of w, i.e., a scenario. Therefore, for a given scenario i, we can write

$$C_{x,y} = \{(x, y) : g_j(x, y) \leq 0, j = 1, \dots, d\} \tag{3.4}$$

where $g_j(x, y)$ is convex for all j. (Note that for scenario i, system (3.4) would have been $C_{x,y}(w^i) = \{(x, y^i) : g^i_j(x, y^i) \leq 0, j = 1, \dots, m_i\}$, i.e., $d = m_i$.) Given a solution for the master problem $\hat{x}$, we need to answer the question: does there exist $\hat{y}$ such that $(\hat{x},\, \hat{y}) \in C_{x,y}$? If such $\hat{y}$ does not exist, we must find a separating hyperplane: this is the purpose of the separation routine.

Notice that the master problem involves the x variables only. For this reason, the separation routine must find a cut in the x space. One approach to do so is given by generalized Benders decomposition Geoffrion [1972]. Here we advocate a simpler approach that allows computation of a separating hyperplane under mild conditions; we discuss its relationship with generalized Benders decomposition in Section 3.2.4.

Define the problem

$$\min_{(x,y)\in C_{x,y}} \frac{1}{2}\|x-\hat{x}\|_x^2, \tag{PROJ}$$

where by $\|\cdot\|_x$ we denote the Euclidean distance in the x space only. If $\hat{x} \notin C_x$, the optimal value of PROJ must be > 0.

Theorem 3.1. *Assume that $C_{x,y}$ is a continuous convex set and constraint qualification conditions are met. Let $(\bar{x}, \bar{y})$ be the optimal solution to (PROJ), $\ell^* > 0$ the optimal objective function value, and $\mu_j, j = 1, \dots, d$ a set of the corresponding optimal KKT multipliers. Let $I = \{j \in \{1, \dots, d\} : g_j(\bar{x}, \bar{y}) = 0\}$. Then, the hyperplane*

$$\left(\sum_{j\in I} \mu_j \nabla g_j(\bar{x},\bar{y})\right)((x,y)-(\bar{x},\bar{y})) \le 0$$

separates $\hat{x}$ from C_x and involves only the x variables. This hyperplane is the deepest valid cut that separates $\hat{x}$ from C_x, if depth is computed in ℓ_2-norm.

Proof. By KKT conditions, we must have

$$-\nabla\frac{1}{2}\|\hat{x}-\bar{x}\|_x^2 = (\hat{x}-\bar{x}) = \sum_{j=1}^{d} \mu_j \nabla g_j(\bar{x},\bar{y})$$

for some $\mu_j \ge 0$. Notice that $\|\cdot\|_x$ does not depend on y, hence the y components of the gradient must be zero. By complementary slackness, $\mu_j g_j(\bar{x},\bar{y}) = 0$ for all j. Then, $\sum_{j\in I} \mu_j \nabla_y g_j(\bar{x},\bar{y}) = 0$. Consider the hyperplane

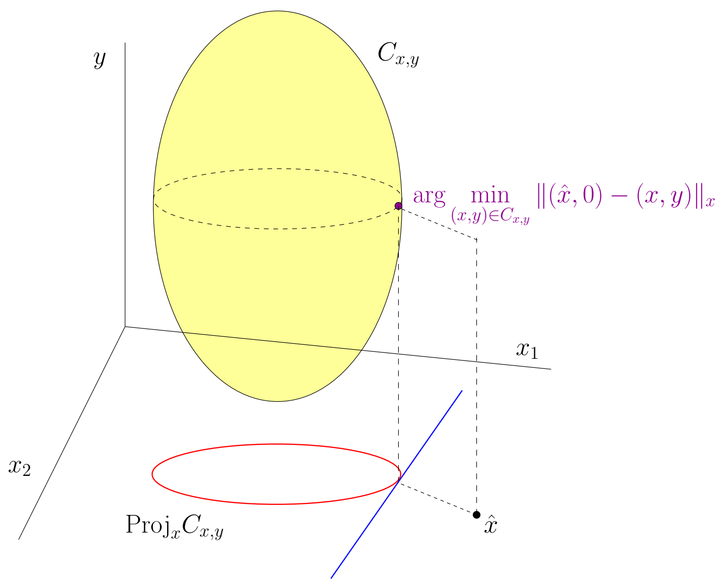
$$\left(\sum_{j\in I} \mu_j \nabla g_j(\bar{x},\bar{y})\right)((x,y)-(\bar{x},\bar{y})) \le 0.$$

Then clearly this hyperplane only involves the x variables, and it does not cut off any point in $C_{x,y}$ by convexity of the g_j's. Moreover, if we plug in the point $\hat{x}$, we obtain

$$\left(\sum_{j\in I} \mu_j \nabla_x g_j(\bar{x},\bar{y})\right)(\hat{x}-\bar{x}) = (\hat{x}-\bar{x})(\hat{x}-\bar{x}) = \|\hat{x}-\bar{x}\|^2 = 2\ell^* > 0.$$

Hence, the hyperplane cuts off $\hat{x}$.

To show that it is the deepest valid cut, notice that $\mathrm{dist}_x(\hat{x},\bar{x}) = 2\ell^*$. Any cut that cuts $\hat{x}$ by more than $2\ell^*$ in Euclidean distance computed in the x space would cut $\bar{x}$ off, forsaking validity. □



y
$C_{x,y}$
$\arg \min_{(x,y)\in C_{x,y}} \|(\hat{x},0)-(x,y)\|_x$
x_1
x_2
$\mathrm{Proj}_x C_{x,y}$
$\hat{x}$

FIGURE 3.1: Separating hyperplane.

The discussion above suggests an easy approach to derive maximally violated separating hyperplanes, which requires fewer assumptions with respect to generalized Benders. We observe that the cutting plane derived is simply $(\hat{x}-\bar{x})^T(x-\bar{x}) \le 0$. It can be computed from the solution of (PROJ) even without the KKT multipliers.

Theorem 3.2. *Let $C_{x,y}$ be a closed set such that $C_x = Proj_x C_{x,y}$ is convex, and $\hat{x} \notin C_x$. Let $(\bar{x}, \bar{y})$ be the optimal solution to (PROJ), $\ell^* > 0$ the optimal objective function value. Then, the hyperplane*

$$(\hat{x}-\bar{x})^T(x-\bar{x}) \le 0$$

separates $\hat{x}$ from C_x. This hyperplane is the deepest valid cut that separates $\hat{x}$ from C_x, if depth is computed in ℓ_2-norm.

Proof. Because $C_{x,y}$ is closed, C_x is closed, and convex by assumption. Therefore, there exists a unique vector v that minimizes $\|v-\hat{x}\|$ over all $v \in C_x$. By definition of (PROJ), $v = \bar{x}$. Then, we can apply the projection theorem (see, e.g., [Bertsekas, 1999, Prop. B.11 (b)]) to obtain

$$(\hat{x}-\bar{x})^T(x-\bar{x}) \le 0 \quad \forall x \in C_x.$$

Hence, this hyperplane is valid for C_x, and it separates $\hat{x}$ because $\|\hat{x}-\bar{x}\|^2 = \ell^* > 0$ by hypothesis. The argument about this being the deepest valid cut is the same as in the previous theorem. □

A sketch of the main elements of Theorem 3.2 can be found in Fig. 3.1. It is evident that the inequalities described in Theorem 3.2 are outer approximation cuts. Outer approximation was introduced in Duran and Grossmann [1986] and has proven to be an extremely useful tool in mixed-integer convex programming, see, e.g., Bonami et al. [2008, 2009], Fletcher and Leyffer [1994]. Outer approximation is used to separate a point not belonging to a convex set from the convex set itself, and typically the point and the set live in the same space. In this chapter, we apply outer approximation to separate a point from the *projection* of a set on a lower-dimensional space, and we do not have an explicit description of such projection: for this reason, to obtain the separating inequality we perform an optimization in the higher-dimensional space, and the result is the outer approximation cut that would have been obtained if we had the explicit description of the projection.

The important difference between Theorem 3.2 and Theorem 3.1 is that we dropped all assumptions, except that $C_{x,y}$ projects to a closed convex set, i.e., we no longer require constraint qualification. However, to find the hyperplane we must be able to solve (PROJ), which is an optimization problem over $C_{x,y}$: the difficulty of separation depends on the difficulty of optimizing over $C_{x,y}$. If $C_{x,y}$ is described as a set of (continuous) nonlinear convex constraints and constraint qualification holds, then Theorem 3.2 is equivalent to Theorem 3.1 and the separation can be carried out in polynomial time. However, an interesting case is when $C_{x,y}$ is the convex hull of the mixed-integer points satisfying a set of convex constraints. In this case, typically $C_{x,y}$ is described as the set of mixed-integer points satisfying some convex constraints, but an explicit description of the convex hull is not available and therefore gradients of the boundary-defining constraints cannot be computed. With the approach we advocate, the description of the convex hull is not necessary: we can solve problem (PROJ), i.e., optimize over $C_{x,y}$, using a Branch-and-Bound solver for convex MINLPs, then apply Theorem 3.2 to obtain a separating hyperplane. We remark that in this case separation cannot be performed in polynomial time in general.

3.2.3 Termination of the Branch-and-Cut algorithm

We now show that Algorithm 3, combined with the separation routine that generates the cut $(\hat{x} - \bar{x})^T(x - \bar{x}) \leq 0$ as in Theorem 3.2, terminates in the case of convex and mixed-integer convex scenario problems under mild assumptions.

Theorem [Kelley, 1960, Sec. 2] considers a continuous convex function $G(x)$ defined on a compact convex set X such that, at every point $\hat{x} \in X$, there exists an extreme support $y = p(x, \hat{x})$ to the graph of $G(x)$ whose gradient is bounded by a constant. Given a cost vector c, if $\hat{x}_h$ defines a sequence of points such that $c\hat{x}_h = \min\{cx | x \in X_h\}$, $h = 0, 1, \ldots$, where $X_0 = X$ and $X_h = X_{h-1} \cup \{x | p(x, \hat{x}_{h-1}) \leq 0\}$, then the sequence $\{\hat{x}_h\}$ contains a subsequence that converges to a point ξ in X with $G(\xi) \leq 0$.

We are ready to prove the following theorem.

Theorem 3.3. *Consider a problem of the form*

$$\max\{cx : \Pr(x \in C_x(w)) \geq 1 - \alpha, x \in X\}, \tag{CCP}$$

where X is compact, $C_x(w)$ is a closed and convex set for all $w = w^1, \ldots, w^k$, and assumptions (A1)-(A3) of Luedtke [2014] are satisfied. Then, given any $\varepsilon_c > 0$, Algorithm 3 finds a solution $\tilde{x}$ with $\|x^* - \tilde{x}\| \leq \varepsilon_c$ after a finite number of iterations, where x^* is an optimal solution to (CCP).

Proof. We can rely on the convergence proof of the counterpart of Algorithm 3 in [Luedtke, 2014, Theorem 3]. Theorem 3.2 shows that the separation routine separates exactly over $C_x(w^i)$ for $i = 1, \ldots, k$. The notable difference with respect to the proof of [Luedtke, 2014, Theorem 3] is that our separation routine does not return inequalities from a finite set, hence we must show that the termination condition for processing a node is satisfied after finitely many iterations of the separation routine. In other words, we must show that after a finite number of separation rounds at a node, the solution to the master problem $\hat{x}$ belongs to $C_x(w^i)$ (or is ε_c-close) for all i such that $z_i = 0$. This is true in the setting of Luedtke [2014] because $C_{x,y}(w^i)$ is a polyhedron, and the chapter considers only inequalities corresponding to extreme points of the Benders cut generating problem, which are in finite number. In the context of the present chapter, it must be proven.

For this, it is sufficient to show that for every $C_x(w^i)$ satisfying the assumptions, the separation routine of Theorem 3.2 requires a finite number of inequalities for ε_c-convergence. This ensures finite ε_c-convergence to the intersection of $C_x(w^i)$ for all i such that $z_i = 0$, if the separation routine is applied to all $C_x(w^i)$. For ease of notation, we drop w^i and discuss a generic set $C_{x,y}$ with projection C_x.

We can now apply the convergence result of Theorem [Kelley, 1960, Sec. 2] as follows. Let X be the set defined by feasible region of the master problem, and define $G(x) = \min_{\tilde{x} \in C_x} \|x - \tilde{x}\|$, i.e., as the distance function from the convex set C_x. Therefore, $G(x)$ is convex. By convexity, an extreme support of $G(x)$ exists at each point of X, and, by the definition of $G(x)$, its gradient is bounded. We have $G(x) = 0 \Leftrightarrow x \in C_x$, $G(x) > 0 \Leftrightarrow x \notin C_x$. Given $\hat{x} \in X$, $\hat{x} \notin C_x$, define $\bar{x} = \arg\min_{\tilde{x} \in C_x} \|\hat{x} - \tilde{x}\|$, so that $G(\hat{x}) = \|\hat{x} - \bar{x}\|$. An extreme support $y = p(x, \hat{x})$ to $G(x)$ at $\hat{x}$ is

$$y = G(\hat{x}) + \nabla^T G(\hat{x})(x - \hat{x}) = \|\hat{x} - \bar{x}\| + \frac{(\hat{x} - \bar{x})}{\|\hat{x} - \bar{x}\|}(x - \hat{x})$$

Then, since $\hat{x} = \bar{x} + \hat{x} - \bar{x}$, the expression $p(x, \hat{x}) \leq 0$ reads as

$$(\hat{x} - \bar{x})(x - \bar{x}) \leq 0,$$

which is exactly the condition we use to (iteratively) separate $\hat{x}$. By Theorem [Kelley, 1960, Sec. 2], we can define a sequence of $\hat{x}_h$ converging to a point ξ in X, $G(\xi) \leq 0$, i.e., $\xi \in C_x$. By definition of convergence, for every ε_c, there exists an integer v such that after v inequalities, $\|\hat{x} - \xi\| \le \varepsilon_c$. This concludes the proof. □

3.2.4 Comparison with generalized Benders cuts

This section investigates the relationship between the separation approach we advocate and generalized Benders decomposition Geoffrion [1972], which applies to the same class of problems studied in this chapter, namely those that can be formulated as (3.2). Here we only discuss the case where the second-stage problems are feasibility problems, as we can always reduce to that case. The result in Geoffrion [1972] assumes that a "dual adequate" algorithm to solve the scenario subproblems is available, that is, if the problem is infeasible a dual certificate of infeasibility can be computed. In its computational considerations it remarks that "it appears necessary" to assume additional properties on the structure of the problem, namely, that the function

$$L(x, \lambda) = \min_{y \in C_{x,y}} \lambda^T g(x, y)$$

can be easily computed for all $x \in X, \lambda \in \mathbb{R}^m, \lambda \ge 0$. In particular this means that we should be able to find an analytical expression for such function. This can be done in some specific situations, for example if the nonlinear functions are separable in x and y (see, e.g., Bloom [1983], França and Luna [1982]), but may be difficult in general if the solution to the minimization problem over y depends on x. Even when that is the case, one issue remains: in the approach of Geoffrion [1972] these functions are the Benders cut added to the master problem, and they have the form of the constraints $g(x, y)$. If the $g(x, y)$ are nonlinear, we are left in the unfortunate situation of possibly adding nonlinear constraints to the master problem. The nonlinear cuts could be stronger than linear inequalities, but are computationally less attractive and would not allow us to use the existing well-developed machinery for linear inequalities, such as mixing techniques Günlük and Pochet [2001]. Of course, one could simply linearize a generalized Benders cut: we show that this is in fact exactly what is happening.

Proposition 3.4. *Assume that constraint qualification conditions are met, and let $(\bar{x}, \bar{y})$ be the optimal solution to (PROJ), μ be the corresponding KKT multipliers. Then, the cut*

$$(\hat{x} - \bar{x})^T (x - \bar{x}) \le 0$$

is the linearization of a generalized Benders cut obtained from $\hat{x}$ with multipliers μ.

Proof. A generalized Benders cut has the form $L(x, \lambda) \le 0$, where $L(x, \lambda) = \min_{y \in C_{x,y}} \lambda^T g(x, y)$ and λ is a nonegative vector such that $\min_{y \in C_{x,y}} \lambda^T g(\hat{x}, y) > 0$; see Geoffrion [1972]. By construction, the hyperplane $\left(\sum_{j \in I} \mu_j \nabla g_j(\bar{x}, \bar{y})\right)((x, y) - (\bar{x}, \bar{y})) = (\hat{x} - \bar{x})^T (x - \bar{x}) \le 0$ is supporting for $\sum_{j \in I} \mu_j g_j(x, y)$ at $(\bar{x}, \bar{y})$, so

$\sum_{j \in I} \mu_j g_j(x, y) \geq (\hat{x} - \bar{x})^T (x - \bar{x})$ for all $(x, y) \in C_{x,y}$ because the left-hand side expression is convex. It follows that

$$\min_{y} \sum_{j \in I} \mu_j g_j(\hat{x}, y) \geq (\hat{x} - \bar{x})^T (\hat{x} - \bar{x}) = \ell^* > 0.$$

This shows that the multipliers μ yield a violated generalized Benders cut. Furthermore, $\sum_{j \in I} \mu_j g_j(\bar{x}, y) \geq (\hat{x} - \bar{x})^T (\bar{x} - \bar{x}) = 0$ for all y, and $\sum_{j \in I} \mu_j g_j(\bar{x}, \bar{y}) = 0$ by complementary slackness, hence $\bar{y} = \arg\min_y \sum_{j \in I} \mu_j g_j(\bar{x}, y)$. It follows that $(\hat{x} - \bar{x})^T (x - \bar{x})$ is the tangent plane to $L(x, \mu)$ at the point $\bar{x}$. $\square$

The fact that outer approximation cuts are linearizations of generalized Benders cuts is well known: since every nonnegative combination of the constraints g_j can be considered a generalized Benders cuts, every valid linear inequality for C_x is a linearization of a generalized Benders cuts. [Abhishek et al., 2010, Sect. 3.1] remarks that aggregating linearizations to the constraints using optimal dual multipliers simplifies the cut, and the unfixed variables disappear from the cut expression.

It is important to remark that our way of generating cuts is conceptually simpler than applying generalized Benders decomposition, and it has some clear advantages. In fact, let $\lambda \geq 0$ be any vector of dual variables that gives rise to a violated generalized Benders cut, i.e., $\min_{y \in C_{x,y}} \lambda^T g(\hat{x}, y) > 0$. Since the expression $\min_{y \in C_{x,y}} \lambda^T g(\hat{x}, y) \leq 0$ is convex, any tangent hyperplane is a valid inequality. The approach of Geoffrion [1972] requires the dual variables λ only, but in order to compute a tangent hyperplane, we additionally need a point about which the linearization is obtained. To this end, Abhishek et al. [2010] proposes a hierarchy of points, where the weakest one is analogous to the ECP method Westerlund et al. [1998] and does not require solving a subproblem, while the strongest one obtains the point by solving the NLP relaxation of the current node. Notice that in our context, because no value for y is initially known, it seems that solving an NLP subproblem to generate the point is a better approach. Furthermore, if the point about which the linearization is generated does not belong to $C_{x,y}$ the tangent hyperplane may not be supporting for C_x, hence it would be dominated by some other valid inequality. Finally, if the set $C_{x,y}$ contains integrality requirements, linearizing generalized Benders cuts is not a viable approach as optimal dual variables are not available unless the convex hull is explicitly known.

In principle, our projection approach to generate a separating inequality can also be applied in the case where $C_{x,y}$ is a polyhedron, and it yields violated Benders cuts from a particular choice of dual variables. The most commonly approach used in the literature is instead to obtain the dual variables by minimizing the largest constraint violation, which corresponds to a specific truncation of the unbounded dual rays (see Fischetti et al. [2010]). The standard approach guarantees that all the inequalities are generated from extreme points of the dual polyhedron, whereas our projection approach may construct a Benders cut from dual variables that are not extreme, in which case the cut would not be extreme either, i.e., it could be obtained as a convex combination of extreme Benders cuts.

3.3 (CCP) for mid-term hydro scheduling

We apply the decomposition algorithm for nonlinear chance-constrained problem of Section 3.2 to the hydro scheduling problem that we describe next.

We consider a multi-period planning problem with T periods (indexed by $t = 1, \dots, T$), where all information regarding period 1 is deterministically known, while the remaining periods are subject to uncertainty. As mentioned in Section 3.1.1, we consider uncertainty with respect to inflows and energy market prices, hence, at each period one of the possible inflow-price scenarios is realized. Our objective in a deterministic setting would be to maximize the profit obtained by selling energy on the energy market. Electrical energy is obtained by transforming the potential energy of the water when, during each period, the water is released from the reservoirs. There are n reservoirs in total, indexed by $h = 1, \dots, n$. We denote by x_{th} the amount of water released in period t from reservoir h, and by w_{th} the water level of reservoir h at the end of the period (w_{0h} is a parameter denoting the initial water level). Parameter f_{th} denotes the natural water inflow in period t at reservoir h. The water released from reservoir h is transformed into an amount of energy that depends on a nonlinear function $g_h(w, x)$. Energy obtained this way, denoted as e_{th} for period t and reservoir h, is sold on the market; since hydro power production has in general a large capacity, we assume to be price-makers on the electricity market, according to a price function $\pi_t(\cdot)$ that depends on the total amount of electrical energy to be sold at period t, namely, $e_t = \sum_{h=1}^{n} e_{th}$. In the deterministic setting, the hydro scheduling problem described above is modeled by the following nonlinear program:

$$\max \sum_{t=1}^{T} \pi_t(e_t) e_t \tag{3.5}$$

$$w_{(t-1)h} - x_{th} + f_{th} \geq w_{th} \qquad t = 1, \dots, T, h = 1, \dots, n \tag{3.6}$$

$$0 \leq x_{th} \leq u_{th} \qquad t = 1, \dots, T, h = 1, \dots, n \tag{3.7}$$

$$q_{th} \leq w_{th} \leq Q_{th} \qquad t = 1, \dots, T, h = 1, \dots, n \tag{3.8}$$

$$e_{th} \leq g_h(w_{th}, x_{th}) \qquad t = 1, \dots, T, h = 1, \dots, n \tag{3.9}$$

$$d_t \leq e_t \leq m_t \qquad t = 1, \dots, T \tag{3.10}$$

$$e_t = \sum_{h=1}^{n} e_{th} \qquad t = 1, \dots, T. \tag{3.11}$$

The objective function (3.5) maximizes the profit obtained by selling the transformed energy. Constraint (3.6) is an inventory constraint that defines the water balance between consecutive periods: since water can be released without obtaining energy (spillage), we have an inequality. Constraints (3.7) and (3.8) impose lower and upper bounds on the quantity of water used for transforming energy and on the water levels in the reservoirs, respectively. Constraints (3.9) define the relation between the released water and the obtained electrical energy at a specific plant h. Finally, (3.10) defines lower and upper bounds on the amount of obtained electrical energy. Notice that the above problem is convex assuming that g_h is concave.

To model uncertainty, Baslis and Bakirtzis [2011] assumes that forecasts for aggregated demand and precipitations are available as discrete random variables. The optimization occurs over a relatively long period of time (i.e., twelve months), therefore it would be unrealistic to assume temporal independence of demand and precipitations, and the assumption in Baslis and Bakirtzis [2011] is that the realization of the random variables at any time period depends on the realization in the previous time period. We follow the approach of Baslis and Bakirtzis [2011]. This yields a scenario tree, where a scenario is a realization of the random parameters over the entire time period, i.e., a sample path. A scenario tree starts from the root node at the first period and, for each possible realization of the random parameters, branches into a node at the next period. The branching continues up to the leaves of the tree, whose number corresponds to the number of scenarios k.

3.3.1 Decomposition

We decompose the problem into a master problem and k scenario subproblems. Each scenario subproblem i includes decision variables x^i_{ht}, and has a feasible region defined by (3.6) – (3.10). In addition, we link the profit in each scenario to an overall measure of profit in the master problem by introducing a master variable ψ that is maximized, and defining the following additional constraints:

$$\psi \leq \sum_{t=1}^{T} \pi^i_t(e_t) e_t \quad i = 1, \dots, k. \tag{3.12}$$

Hence, a specific scenario is satisfied given the decision variables in the master (energy obtained in the first time period, and measure of profit ψ) if not only constraints (3.6) – (3.10) can be satisfied for subsequent time periods, but also the total profit for the scenario is not smaller than ψ. Since the master maximizes the profit that can be obtained by satisfying a subset of scenarios having associated probability not smaller than $1 - \alpha$, this is equivalent to optimizing the α-quantile of the profit.

Following Baslis and Bakirtzis [2011], we assume that all scenarios have an associated probability of $1/k$ (modifying the formulation to allow for nonuniform scenario probabilities is straightforward), and the joint chance constraints are equivalent to imposing that at least $k-p$ scenarios are satisfied, where $p = \lfloor \alpha k \rfloor$. Nonanticipativity constraints are enforced by the master, guaranteeing that for all t, decisions up to period t are the same for all sample paths that are identical up to t. Given two scenario indices i and r, define $\tau(i,r)$ as the largest time period index such that the sample path realizations of scenarios i and r are identical up to it. We can then write the initial master problem (before addition of outer approximation cuts) as the following MILP:

$$\max \psi \tag{3.13}$$

$$\sum_{i=1,\ldots,k} z_i \leq p, \tag{3.14}$$

$$x^i_{th} = x^r_{th}, \quad i = 1,\ldots,k-1, \; r = i+1,\ldots,k, \; t \leq \tau(i,r), \; h = 1,\ldots,n \tag{3.15}$$

$$0 \leq x^i_{th} \leq u_{th} \qquad t = 1,\ldots,T-1, \; i = 1,\ldots,k, \; h = 1,\ldots,n \tag{3.16}$$

$$z_i \in \{0,1\} \qquad i = 1,\ldots,k. \tag{3.17}$$

where (3.14) is the joint probability constraint, constraints (3.15) express nonanticipativity, constraints (3.16) impose bounds on the quantity of water released. We remark that in practice we do not explicitly write constraints (3.15), because we keep only one copy of the x variables for all sample paths identical up to a given period, implicitly performing the substitution. This is conceptually equivalent and reduces the size of the problem.

3.3.1.1 Electricity generation function

The transformation of the water potential energy into electrical energy is described in terms of a nonlinear power function $v_h(w,\dot{x})$ that depends on the water flow and water level w at reservoir h. We assume that the water flow and level are constant within each time period, and that the amount of electrical energy obtained during a given period is directly proportional to the length of the period θ_t. Hence, we can write

$$g_h(w_{th}, x_{th}) = v_h(w_{th}, x_{th}/\theta_t)\theta_t. \tag{3.18}$$

Several alternatives are proposed in the literature regarding the shape of $v_h(w,\dot{x})$ (see e.g., Bacaud et al. [2001], Chang and Chen [1998], Salam et al. [1998]). These alternatives depend on the characteristics of each power plant and typically must be experimentally evaluated.

The most common power functions consider power as a quadratic expression of the flow, as $v_h = \rho(x/\theta_t)^2 + \nu x/\theta_t + \sigma$, where the values of the coefficients ρ, ν, and σ, when specified, accurately describe the characteristics of several real-world plants. The value of these parameters is not a constant, but it is instead read or interpolated from a table, and depends on the water level w (see, e.g., Ružić et al. [1996]). Instead of interpolating the values from a table, since the value of the parameters ρ, ν, σ is approximately linear in the water level w Salam et al. [1998], we define the power function as

$$v_h(w,x) = (w+\eta)(\rho(x/\theta_t)^2 + \nu x/\theta_t + \sigma), \tag{3.19}$$

where η is then a fourth parameter to be experimentally tuned.

3.3.1.2 Demand and price function

Obtained electrical energy can be sold on the electricity market at the market price; since we are considering a hydro power producer with a large capacity, the producer is a price-maker, i.e., the market price depends on the amount of energy that it sells. We define a linear function to describe the price-quantity relation, obtained by linearizing the staircase-shaped price-quantity functions of Baslis and Bakirtzis [2011]. A finer description of the market effect of a price-maker activity can be obtained by using staircase-shaped decreasing functions. Modeling a staircase function would require binary variables in the subproblems, making them computationally more difficult. Notice that as remarked in Section 3.2, our approach can theoretically deal with binary variables in the scenario subproblems, but this complicates the solution process and we did not test it numerically. Using a linear price function, the profit-quantity relation in equation (3.5) is expressed by a quadratic function of the energy, that is

$$\pi_t(e_t)e_t = (\pi_1 e_t + \pi_0)e_t,$$

where we recall that $e_t = \sum_{h=1}^{n} e_{th}$.

3.3.2 Data

The computational evaluation presented in this chapter considers a case study based on the data from Baslis and Bakirtzis [2011], which describes a hydro system configuration comprising 10 major hydroplants of the Greek power system, for a production capacity of 2720 MW. As in Baslis and Bakirtzis [2011], we consider a three period configuration covering 12 months. The choice of the time periods is based on the Greek hydrological and load demand patterns, where high inflows are observed in winter and spring, and a load peak is observed in summer: the first period is the month of October, the second period goes from November to February, and the third period from March to September. Inflows and demand curves are computed based on historical data; we refer the reader to Baslis and Bakirtzis [2011] for details. The first time period is deterministic, as previously mentioned; a scenario tree (Figure 3.2) comprising 90 scenarios is obtained by considering 5 inflow realizations coupled with 3 demand realizations at the second time period, and 3 inflow realizations coupled with 2 demand realizations at the third time period.

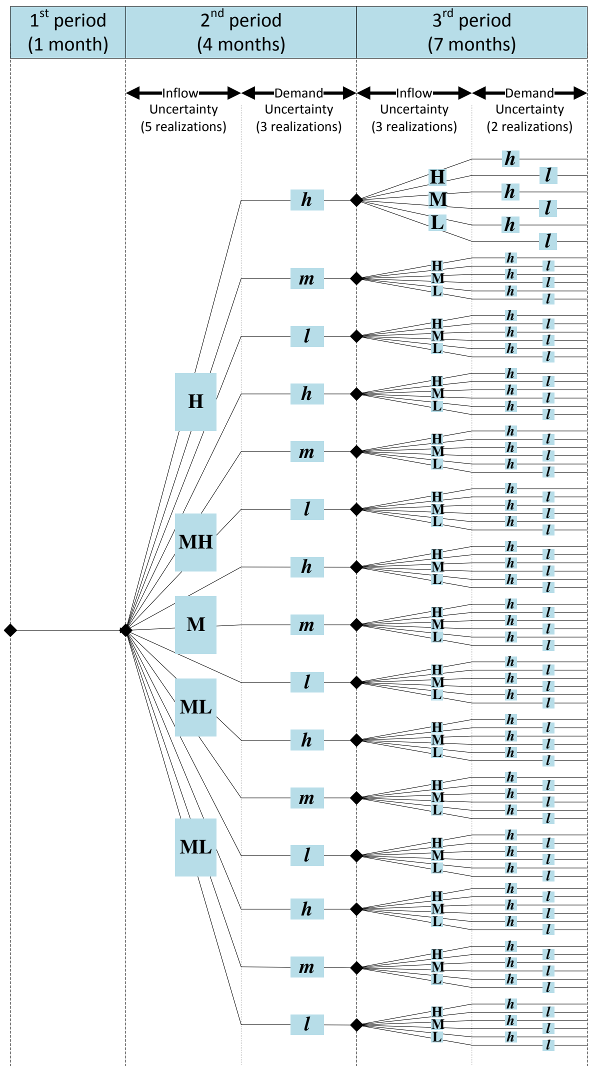

1st period (1 month)
2nd period (4 months)
3rd period (7 months)
Inflow Uncertainty (5 realizations)
Demand Uncertainty (3 realizations)
Inflow Uncertainty (3 realizations)
Demand Uncertainty (2 realizations)
H
MH
M
ML
ML
h
m
l
H
M
L

FIGURE 3.2: Yearly three-period 90-scenario tree.

Period	**2**					**3**		
Realization	**H**	**MH**	**M**	**M L**	**L**	**H**	**M**	**L**
Factor	1.5	1.25	1	0.75	0.5	1.25	1	0.75

TABLE 3.1: Inflow realizations.

For each of these possible realizations we used a corresponding scale factor to modify the average inflow and demand, as in Baslis and Bakirtzis [2011]. These factors are presented in the Tables 3.1 and 3.2. In order to express the high dependence of the third period on the second period, we multiplied the scale factors of the third period with the scale factor of its parent second period realization.

For each scenario we modified the average demand and price function using the scale factors in Table 3.2 and the modification factors of the price in the Tables 3.3 and 3.4, respectively for the second and the third period.

Period	**2**			**3**	
Realization	**h**	**m**	**l**	**h**	**l**
Factor	1.1	1	0.9	1.1	0.9

TABLE 3.2: Demand realizations.

Inflow	**H**			**MH**			**M**			**ML**			**L**		
Demand	**h**	**m**	**l**	**h**	**m**	**l**	**h**	**m**	**l**	**h**	**m**	**l**	**h**	**m**	**l**
Factor	1	0.95	0.925	1	0.975	0.95	1.05	1	0.975	1.15	1.05	1	1.25	1.15	1

TABLE 3.3: Price modification for the second period.

Inflow	**H**		**M**		**L**	
Demand	**h**	**l**	**h**	**l**	**h**	**l**
Factor	1	0.95	1.05	0.975	1.15	1

TABLE 3.4: Price modification for third period.

In addition, the final reservoir volume w_{Th} was set equal to half of the initial value w_{0h}.

3.4 Computational experiments

In this section we report on the experimental results obtained by the described Branch-and-Cut algorithm when solving decomposable chance constrained problems, where the subproblems are continuous and convex. We tested the algorithm on instances described in Section 3.3, and compare the algorithm performance with the direct solution of the large MINLP (3.2) by a general purpose solver for convex MINLPs.

The objective of these experiments is twofold: on the one hand, they are intended to assess the algorithmic performance of the method we propose; on the other hand, they allow us to evaluate our modeling approach for mid-term hydro scheduling problems, determining the size of the instances that can successfully be dealt with, and highlighting the trade-off between profit and robustness of the solution.

3.4.1 Implementation details

We implemented the Branch-and-Cut algorithm within the IBM ILOG CPLEX 12.6 MILP solver, and solved the convex subproblems with IPOPT 3.12 using the interface provided by BONMIN. In our implementation, CPLEX manages the branching tree of the master problem, and returns the control to a user-written callback function when the solution associated with a tree node is integer feasible.

Within the callback function, we define a separation problem PROJ for those scenarios i having associated variable $z_i = 0$, i.e., the scenarios whose constraints must be satisfied.

Problems PROJ are then solved by IPOPT. If the optimal solution of problem PROJ has strictly positive value for some scenario j, that is, the current master solution $\hat{x}$ violates the constraints of scenario j, then we derive a (single) valid cut $\gamma x \leq \beta_j$ separating $\hat{x}$ from the feasible region of scenario j, as explained in Section 3.2.2.

Then, we consider adding the obtained cut to the master problem in two alternative ways:

big M The cut is directly added to the master problem in the form $\gamma x \leq \beta_j + M z_j$. We compute the value for the M coefficient as: $M = \sum_{l:\gamma_l>0} \gamma_l u_l - \beta_j$, where l denotes the index of the x variables in the cut and u_l is the associated upper bound in the master problem;

lifted The cut is lifted computing valid coefficients for the z_i variables corresponding to other scenarios, i.e., $i \neq j$, as suggested in Luedtke [2014].

In the second case, for every i we first compute the coefficient β_i making the inequality valid for the corresponding scenario w^i, solving the optimization problem:

$$\beta_i = \max\{\gamma x | x \in X \cap C_x(w^i)\}. \tag{3.20}$$

Assuming the β_i values, $i = 1, \ldots, k$, are sorted by non-decreasing order, we consider the first $p+1$ scenarios (recall $p = \lfloor \alpha k \rfloor$), and we obtain the following valid inequalities (see [Luedtke, 2014, Lemma 1]):

$$\gamma x + (\beta_i - \beta_{p+1}) z_i \leq \beta_i, \; i = 1, \ldots, p. \tag{3.21}$$

From this basic set of inequalities, one could obtain stronger star inequalities (see Atamtürk et al. [2000]). The basic idea is that, given an ordered subset $T = \{t_1, t_2, \ldots, t_l\}$ of $\{1, \ldots, p\}$, one can derive the following star inequality (see Luedtke [2014] for further details):

$$\gamma x + \sum_{i=1}^{l} (\beta_{t_i} - \beta_{t_{i+1}})) z_i \leq \beta_{t_1}. \tag{3.22}$$

Since the star inequalities (3.22) are in exponential number, they need to be separated. Separation can be performed by solving a longest path problem in an acyclic digraph. However, since we are separating integer solutions in the z variables, the most violated inequality by a solution $(\hat{x}, z)$ is exactly the inequality (3.21) associated with the first t_i in the ordering such that $z_{t_i} = 0$. Thus, in our implementation we add precisely the inequalities (3.21).

Notice that to ensure correctness of the Branch-and-Cut algorithm, it is sufficient to find one violated scenario i having associated variable $z_i = 0$, and to add the cut obtained by solving the PROJ problem to the master problem (alternatively, all scenarios having associated variable $z_i = 0$ are satisfied, and the node does not have to be processed further). In our implementation we considered the following alternatives to determine how and when to perform separation:

sepAll Separation is performed at integer solutions for each scenario i having associated variable $z_i = 0$;

sepGroup Scenarios are partitioned in subsets, where each subset includes those scenarios of the scenario tree having a common ancestor at the second time period (i.e., the corresponding sample paths are equal up to that point in time). Separation is performed at integer solutions for each group, until a violated scenario i in the group having associated variable $z_i = 0$ is found.

The rationale for sepGroup is that scenarios in the same group have common decision variables at the second time period, hence a cut for one of these scenarios might change the primal solution for all scenarios in the same group. We tested two additional strategies that turned out to have poor computational performance, hence we describe them briefly below, but we will not report the corresponding results:

sep1 Separation is performed at integer solutions until the first violated scenario i having associated variable $z_i = 0$ is found.

sepFrac We attempt to separate cuts at fractional solutions using one of the other strategies mentioned above.

Both **sep1** and **sepFrac** were ineffective for the same reason: these two strategies increase the number of separation rounds, and, as it will be seen in the next section, the vast majority of the CPU time is already spent in solving the nonlinear separation subproblems, therefore increasing in the number of separation rounds is an issue.

Concerning the large MINLP (3.2), it is tackled through BONMIN, with IPOPT 3.12 as embedded nonlinear solver. For each constraint of the MINLP formulation to be activated/deactivated by the associated z variable, we compute the smallest value of the M coefficient using the bounds on the x variables and the maximum profit that can be obtained in the scenarios by releasing the associated water quantities.

3.4.2 Computational performance

The data from Baslis and Bakirtzis [2011] includes 10 hydroplants and a scenario tree with 90 equiprobable scenarios. From these data, we construct 5 smaller configurations with a number of plants chosen from the set $\{1, 2, 5, 7, 10\}$. For each configuration, we can specify the robustness of the solution: we consider values of the probability α starting from $\alpha = 0.5$ and decreasing by 0.1 down to $\alpha = 0.1$ (the computed solution must satisfy scenarios with associated probability of at least $1 - \alpha$). In the discussion

	Nodes		Time			
B&C algorithm	B&B	Sep.	CPU [s]	% NLP	NLP solved	Added cuts
sepAll-bigM	30.4	4.2	121.1	99.5	8,228.0	1,384.2
sepGroup-bigM	28.9	4.0	154.5	99.5	10,504.2	1,346.9
sepAll-lifted	10.3	2.7	1,251.1	100.0	(8,309.6) 102,529.3	1,409.1
sepGroup-lifted	13.3	2.7	1,098.2	100.0	(9,829.5) 87,653.6	1,188.3

TABLE 3.5: Performance summary for the four main variants of the B&C algorithm.

about the performance of the hydroplants in Section 3.4.4 we additionally report results for $\alpha = 0.05$, but they are not included here as they do not provide further insight. Moreover, for 5 and 10 hydroplants and all values of α, we considered four simplified scenario trees that contain 30, 48, 60 or 72 scenarios. We therefore obtain 65 instances of varying difficulty. All experiments are performed on a single node of a cluster containing machines equipped with an Intel Xeon E3-1220 processor clocked at 3.10 GHz and 8 GB RAM.

In Table 3.5 we report the main indicators to evaluate the performance of the four variants of the Branch-and-Cut algorithm that are obtained combining the separation procedures **sepAll** and **sepGroup** with the **bigM** and **lifted** procedures to add cuts to the master problem. More specifically, the table reports average values of the total number of Branch-and-Bound nodes (second column), number of Branch-and-Bound nodes at which separation is performed (third column), total computing time and fraction of time spent in the nonlinear separation subproblems (fourth and fifth column respecitvely), number of nonlinear programs solved (sixth column), number of cuts added to the master problem (seventh column). For the **bigM** case, the number of NLPs solved is the same as the number of iterations of the separation procedure. For the **lifted** case, the number in brackets in the seventh column is the number of nonlinear programs solved to prove a given scenario is satisfied or derive the cut (iterations of the separation procedure), and the number of NLPs solved includes the NLPs to lift the cut.

All versions of the B&C algorithm solve all tested instances in less than 2 hours of computing time per instance. The **sepAll-bigM** variant is the fastest version on average, and we take it as our reference. Table 3.5 shows that the number of Branch-and-Bound nodes is very small on average (about 30), and almost all the computing time is spent in solving NLPs (on average, 8,228 NLPs per instance). Most of the separation iterations occur at the root node of the Branch-and-Cut algorithm (approximately 3/4 on average). We observe that many separation rounds are performed at each node where separation occurs. In the majority of the cases, when several mixed-integer solutions are produced at the same node each new mixed-integer solution differs from the previous one only in its continuous components. Only occasionally a new mixed-integer solution has different values for the z variables, unless of course the separation is performed at different nodes of the Branch-and-Bound tree. This behavior can be explained by recalling that the master problem (3.13)-(3.17) is not aware of the non-linear dynamics of the scenario subproblems, therefore a good approximation must be constructed by means of several linear cuts, even when the integer variables are fixed. Results with **sepGroup-bigM** are similar, with a small increase in the number of NLPs solved, and a corresponding increase of computing time.

Concerning the **lifted** cuts, we note that given a cut $\gamma x \leq \beta_i$ obtained for some scenario i, computing the lifting is computationally expensive due to the solution of several additional NLPs. This additional effort would be justified only if lifted cuts were able to significantly reduce the number of cut separation iterations with respect to **bigM** cuts. Table 3.5 shows that this is not the case: although the number of Branch-and-Bound nodes is reduced, the average number of separation iterations is of similar magnitude. As a consequence, **sepAll-lifted** and **sepGroup-lifted** solve many more NLPs and the CPU time increases accordingly. The ineffectiveness of lifted cuts can be explained in connection to the specific structure of the scenario tree we consider: when solving the optimization problem (3.20) for a given scenario i and a given hyperplane γx, only a subset of the variables with nonzero coefficient in γ appears in nontrivial constraints (i.e., not bound constraints) for scenario i. Hence, the lifting procedure is rarely able to produce stronger cuts. The same observation on the weak computational performance of the mixing inequalities generated by an analogous lifting procedure is reported in Qiu et al. [2014], where a chance-constrained formulation is studied as well.

The computational performance of BONMIN's NLP-based Branch-and-Bound algorithm, applied directly to the MINLP (3.2), are also evaluated on all 65 problem instances. The time limit for BONMIN is set to 10 hours. In Figure 3.3 we report a performance profile for the **sepAll-bigM** Branch-and-Cut algorithm and BONMIN, for the whole set of instances. The Branch-and-Cut algorithm can solve all instances, while BONMIN's Branch-and-Bound algorithm hits the time limit in 7 cases. In addition, the profiles clearly show the significantly better performance of the proposed approach compared to the direct solution of the large MINLP (3.2). Before reporting detailed computational results comparing the two approaches, we remark that we tried to solve the MINLP (3.2) with additional solvers based on other solution methods, namely, the BONMIN Outer Approximation algorithm, the BONMIN hybrid algorithm and the FilMINT Branch-and-Cut algorithm. None of the mentioned solvers could consistently handle the MINLP (3.2), and all solvers were plagued by severe numerical issues; as a consequence, they could correctly solve only small instances or instances with simplified nonlinear functions.The computational results obtained with these solution methods are described in Section **??**.

In Table 3.6 we report detailed results for a subset of instances of increasing complexity, comparing **sepAll-bigM** with BONMIN Branch and Bound. All instances in the table have 90 scenarios. The table reports the number of hydroplants and the level of risk α in the first two columns. Subsequent columns report the results obtained

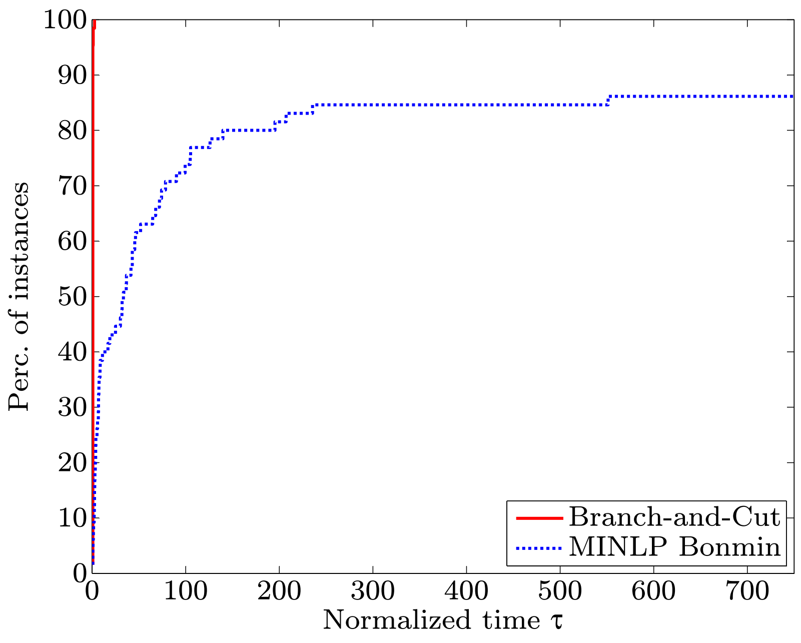


FIGURE 3.3: Performance profiles for 65 instances.

by the Branch-and-Cut algorithm, as in Table 3.5. The last two columns report the performance of BONMIN Branch-and-Bound algorithm, indicating the total CPU time and the number of Branch-and-Bound nodes. The Branch-and-Cut algorithm solves all instances in less than 10 minutes each, and in a very limited number of Branch-and-Bound nodes. Instances with a smaller number of hydroplants appear easier for the Branch-and-Cut algorithm, while the level of risk α has little effect on the solution time. Solution via BONMIN Branch-and-Bound algorithm takes a much larger number of nodes and computing time (two orders of magnitude larger on average). Very few instances are solved to optimality within 1 hour of computing time.

Similar considerations can be drawn from Tables 3.7 and 3.8, where all the instances are for the 5 and the 10 hydroplants configuration, and different number of scenarios, as reported in the first column. In addition, the tables clearly show that reducing the number of scenarios makes the problem easier, for both the Branch-and-Cut algorithm and the BONMIN Branch-and-Bound algorithm.

		Branch and Cut						BONMIN	
Plants	α	B&B nodes	Sep. nodes	Time (s)	% time NLP	NLP solved	Added cuts	Time (s)	Nodes
1	0.1	31	5	22.3	100.0	2,520	228	155.6	365
1	0.2	9	3	26.5	100.0	3,114	289	2,775.4	14,173
1	0.3	21	4	23.0	99.9	2,799	304	2,291.6	12,955
1	0.4	14	4	20.9	99.9	2,845	327	4,331.8	26,144
1	0.5	16	3	15.7	99.8	2,160	273	3,691.6	22,856
2	0.1	16	5	27.0	99.9	2,844	395	1,836.7	2,606
2	0.2	1	1	16.0	99.7	1,746	339	14,279.8	24,749
2	0.3	31	3	27.1	99.7	2,925	418	14,961.3	29,606
2	0.4	35	4	32.4	99.8	3,514	482	6,326.3	20,554
2	0.5	7	2	8.1	99.8	990	329	7,996.1	27,761
5	0.1	12	2	98.6	99.9	7,887	1,315	2,471.5	3,060
5	0.2	34	5	82.3	99.7	6,720	1,258	5,331.1	6,255
5	0.3	17	3	93.6	99.8	7,676	1,339	13,086.6	17,800
5	0.4	63	7	89.1	99.6	7,380	1,338	9,376.6	11,745
5	0.5	27	5	88.7	99.7	7,427	1,325	8,011.3	10,742
7	0.1	9	3	205.3	99.8	14,883	2,021	9,554.3	7,001
7	0.2	36	3	136.7	99.7	9,977	1,803	7,107.8	6,338
7	0.3	47	6	180.3	99.6	13,201	2,424	5,776.8	4,445
7	0.4	115	6	131.8	99.1	9,052	1,605	16,619.1	13,397
7	0.5	74	9	175.1	99.5	11,783	1,785	7,520.0	6,159
10	0.1	45	7	237.6	99.6	14,375	2,295	4,520.6	2,189
10	0.2	33	4	200.9	99.6	11,822	2,062	9,186.7	4,811
10	0.3	29	5	383.6	99.6	23,904	3,602	14,136.2	6,380
10	0.4	131	9	429.7	99.1	26,668	3,845	13,818.4	7,035
10	0.5	94	8	414.2	99.2	26,280	3,821	T.L.	17,077

Table 3.6: Comparison between **sepAll-bigM** and BONMIN Branch and Bound on configurations of 1, 2, 5, 7, and 10 hydroplants and 90 scenarios.

		Branch and Cut						BONMIN	
		B&B	Sep.		% time	NLP	Added		
# scen.	α	nodes	nodes	Time (s)	NLP	solved	cuts	Time (s)	Nodes
30	0.1	1	1	25.7	99.8	1,947	435	19.7	5
30	0.2	5	2	25.6	99.8	2,046	424	71.6	276
30	0.3	6	3	30.3	99.7	2,382	566	203.5	878
30	0.4	1	1	19.2	99.9	1,576	381	51.9	191
30	0.5	2	2	15.3	99.5	1,178	326	60.5	218
48	0.1	10	3	32.5	99.8	2,688	685	238.1	221
48	0.2	13	5	68.6	99.8	5,742	948	2,504.6	3,809
48	0.3	15	4	23.5	99.8	1,986	562	1,748.8	2,609
48	0.4	7	2	44.5	99.8	3,728	699	159.6	249
48	0.5	8	2	43.9	99.8	3,624	744	492.4	26
60	0.1	4	2	57.1	99.9	4,597	765	132.0	57
60	0.2	8	4	85.0	99.8	6,971	1,030	454.2	846
60	0.3	25	6	60.3	99.7	4,929	1,099	4,351.6	2,605
60	0.4	27	5	55.5	99.7	4,632	1,056	4,351.6	8,621
60	0.5	11	2	42.6	99.8	3,571	704	377.3	720
72	0.1	32	8	100.1	99.8	8,262	1,260	408.0	279
72	0.2	59	7	84.5	99.7	7,061	1,249	T.L.	29,114
72	0.3	46	7	80.1	99.8	6,779	1,118	T.L.	27,436
72	0.4	30	6	89.8	99.7	7,493	1,153	T.L.	25,633
72	0.5	28	4	43.4	99.7	3,636	801	1,812.1	2,836
90	0.1	12	2	98.6	99.9	7,887	1,315	2,471.5	3,060
90	0.2	34	5	82.3	99.7	6,720	1,258	5,331.1	6,255
90	0.3	17	3	93.6	99.8	7,676	1,339	13,086.6	17,800
90	0.4	63	7	89.1	99.6	7,380	1,338	9,376.6	11,745
90	0.5	27	5	88.7	99.7	7,427	1,325	8,011.3	10,742

Table 3.7: Comparison between **sepAll-bigM** and BONMIN Branch and Bound on configurations with 5 hydroplants and 30, 48, 60, 72, and 90 scenarios.

		Branch and Cut						BONMIN	
# scen.	α	B&B nodes	Sep. nodes	Time (s)	% time NLP	NLP solved	Added cuts	Time (s)	Nodes
30	0.1	1	1	74.9	99.7	4,506	780	27.2	7
30	0.2	2	2	23.5	99.7	1,470	512	39.7	34
30	0.3	1	1	34.7	99.6	2,171	666	85.0	117
30	0.4	15	3	55.0	99.7	3,396	788	60.1	85
30	0.5	1	1	24.9	99.8	1,425	409	36.0	30
48	0.1	30	3	115.1	99.7	6,643	1,378	868.9	347
48	0.2	29	5	187.3	99.5	11,278	2,121	6,286.4	2,029
48	0.3	38	7	266.5	99.5	16,702	2,642	2,301.4	1,169
48	0.4	14	3	190.4	99.6	11,304	1,615	5,785.1	2,907
48	0.5	19	3	71.0	99.5	4,283	974	52.6	1
60	0.1	1	1	215.4	99.7	12,992	1,956	1,299.7	615
60	0.2	11	3	209.1	99.6	13,091	2,182	1,472.1	891
60	0.3	31	5	377.1	99.4	22,024	3,408	1,111.1	746
60	0.4	106	8	294.9	99.4	18,419	2,659	1,155.8	769
60	0.5	39	6	127.7	99.4	7,923	1,869	406.7	329
72	0.1	3	2	183.8	99.6	11,181	2,399	7,896.0	2,692
72	0.2	29	4	221.9	99.4	13,700	2,698	T.L.	8,575
72	0.3	48	7	284.4	99.5	17,814	2,942	T.L.	11,505
72	0.4	112	8	458.5	99.1	28,658	3,905	T.L.	10,471
72	0.5	161	11	260.1	99.0	16,522	2,844	4,421.3	2,860
90	0.1	45	7	237.6	99.6	14,375	2,295	4,520.6	2,189
90	0.2	33	4	200.9	99.6	11,822	2,062	9,186.7	4,811
90	0.3	29	5	383.6	99.6	23,904	3,602	14,136.2	6,380
90	0.4	131	9	429.7	99.1	26,668	3,845	13,818.4	7,035
90	0.5	94	8	414.2	99.2	26,280	3,821	T.L.	17,077

Table 3.8: Comparison between **sepAll-bigM** and BONMIN Branch and Bound on configurations with 10 hydroplants and 30, 48, 60, 72, and 90 scenarios.

3.4.3 Quadratic electricity generation function

We tested the 65 instances also using a different formulation for (3.9). We approximated the nonlinear electricity generation function to a quadratic function that depends only on the water flows. As we mentioned in Section 3.3, the most common power functions consider power as a quadratic expression of the flow, as $v_h = \rho(x/\theta_t)^2 + \nu x/\theta_t + \sigma$. Thus we obtained convex quadratic subproblems and we tested the Branch-and Cut algorithm solving the subproblems with IPOPT 3.12 using the interface provided by BONMIN and CPLEX 12.6.

The computational performance of BONMIN's NLP-based Branch-and-Bound algorithm, applied directly to the MINLP (3.2) using a quadratic formulation for (3.9), are also evaluated on all 65 problem instances. The time limit is still set to 10 hours.

In Table 3.9 we report detailed results for a subset of instances of increasing complexity, comparing **sepAll-bigM** using BONMIN for solving the subproblems, **sepAll-bigM** using CPLEX for solving the subproblems and BONMIN Branch and Bound. All instances in the table have 90 scenarios. The table reports the number of hydroplants and the level of risk α in the first two columns. Subsequent columns report the results obtained by the Branch-and-Cut algorithm using BONMIN and using CPLEX. The last two columns report the performance of BONMIN Branch-and-Bound algorithm, indicating the total CPU time and the number of Branch-and-Bound nodes. The Branch-and-Cut algorithm solves all instances in less than 2 hours each, and in a limited number of Branch-and-Bound nodes. Instances with a smaller number of hydroplants appear easier for the Branch-and-Cut algorithm using CPLEX, while instances with a larger number of hydroplants appear easier for the Branch-and-Cut algorithm using BONMIN. Instead solution via BONMIN Branch-and-Bound algorithm takes a much larger number of nodes and computing time in most of the cases.

Similar considerations can be drawn from Tables 3.10 and 3.11, where all the instances are for the 5 and the 10 hydroplants configuration, and different number of scenarios, as reported in the first column. But the tables clearly show that reducing the number of scenarios makes the problem easier for the BONMIN Branch-and-Bound algorithm.

		Branch and Cut BONMIN						Branch and Cut CPLEX						BONMIN	
Plants	α	B&B nodes	Sep. nodes	Time (s)	% time NLP	NLP solved	Added cuts	B&B nodes	Sep. nodes	Time (s)	% time NLP	NLP solved	Added cuts	Time (s)	Nodes
1	0.1	11	2	20.4	99.7	3,249	506	11	2	8.9	99.5	3,249	513	29.7	41
1	0.2	31	6	23.2	99.8	3,618	503	31	6	10.6	99.7	3,618	505	1,285.8	7,846
1	0.3	21	4	21.3	99.9	3,064	530	21	4	9.2	99.7	3,065	533	907.0	5,877
1	0.4	5	3	17.1	99.8	2,359	375	24	4	9.4	99.7	2,952	421	1,858.0	12,371
1	0.5	15	4	23.4	99.9	3,027	411	15	4	10.1	99.9	3,027	416	1,201.9	7,684
2	0.1	10	2	26.1	99.7	2,520	615	25	2	16.1	99.3	3,413	713	83.4	87
2	0.2	68	7	39.3	99.7	3,762	657	69	6	17.0	99.3	3,402	651	5,088.6	16,402
2	0.3	51	6	44.0	99.8	4,185	713	19	4	22.6	99.6	4,248	728	2,148.7	7,200
2	0.4	76	6	52.5	99.7	5,014	800	51	6	21.5	99.6	3,762	672	2,327.0	7,423
2	0.5	13	4	30.8	99.8	2,970	529	10	3	21.2	99.7	3,896	781	2,259.1	7,733
5	0.1	11	3	149.7	99.6	11,597	2,251	31	3	163.0	99.0	13,666	3,271	108.0	17
5	0.2	101	12	314.7	99.4	24,126	3,176	121	6	157.3	98.6	13,847	3,503	2,531.1	3,323
5	0.3	128	8	179.5	99.2	13,966	2,422	109	6	230.9	98.4	19,223	3,840	5,859.6	8,702
5	0.4	70	7	228.1	99.5	16,989	2,650	253	7	244.5	97.5	20,300	4,793	1,792.2	2,702
5	0.5	28	4	107.7	99.6	8,309	1,386	28	5	185.6	99.2	14,684	3,538	2,468.9	3,647
7	0.1	34	4	562.9	99.1	38,537	5,842	88	8	914.8	97.9	64,080	11,629	280.3	62
7	0.2	54	5	628.7	99.2	43,129	5,778	67	8	864.6	97.2	57,010	11,101	3,195.4	3,014
7	0.3	110	8	421.7	97.8	29,277	5,577	161	10	496.6	96.8	32,768	7,554	3,810.3	3,846
7	0.4	230	8	315.3	97.1	21,721	4,213	113	8	501.2	96.8	32,532	8,760	10,321.6	10,559
7	0.5	240	21	923.8	98.2	64,110	9,333	69	8	569.2	97.9	36,329	8,615	2,597.4	2,573
10	0.1	45	4	1,744.6	98.3	104,781	13,898	42	5	1,216.1	96.2	50,353	17,448	205.2	18
10	0.2	217	15	3,675.0	96.8	218,367	26,881	208	11	4,053.7	91.8	145,394	42,084	5,716.7	3,756
10	0.3	203	16	2,000.0	96.2	118,623	17,354	295	13	3,394.1	90.8	118,098	37,627	1,615.8	1,051
10	0.4	272	21	2,794.0	96.1	163,791	21,959	270	16	3,758.6	92.0	128,264	39,488	4,846.4	2,953
10	0.5	341	22	1,956.5	94.7	114,643	19,276	536	45	3,232.8	93.7	113,101	35,454	2,893.1	1,846

TABLE 3.9: Comparison among **sepAll-bigM** BONMIN, **sepAll-bigM** CPLEX, and BONMIN Branch and Bound on configurations of 1, 2, 5, 7, and 10 hydroplants and 90 scenarios.

		Branch and Cut BONMIN						Branch and Cut CPLEX						BONMIN	
# scen.	α	B&B nodes	Sep. nodes	Time (s)	% time NLP	NLP solved	Added cuts	B&B nodes	Sep. nodes	Time (s)	% time NLP	NLP solved	Added cuts	Time (s)	Nodes
30	0.1	1	1	46.7	99.7	3,673	963	1	1	62.1	99.7	4,782	1,474	16.9	22
30	0.2	5	2	63.4	99.8	4,900	870	3	2	124.5	99.7	9,167	2,067	72.7	286
30	0.3	1	1	56.1	99.8	4,400	746	4	2	110.2	99.6	7,821	1,954	15.7	38
30	0.4	2	2	35.3	99.5	2,813	1,009	7	4	26.6	99.5	2,309	1,142	26.2	70
30	0.5	9	3	29.7	99.3	2,359	1,092	13	3	29.1	99.7	2,431	1,021	17.0	26
48	0.1	3	2	69.7	99.7	5,586	1,389	19	4	134.8	99.4	8,936	3,409	161.7	189
48	0.2	21	4	100.6	99.4	8,046	1,874	12	2	132.2	99.6	9,216	2,468	170.6	175
48	0.3	30	4	103.8	99.6	8,393	1,665	8	2	111.5	99.4	8,537	3,148	1,132.9	1,489
48	0.4	25	5	93.6	99.6	7,520	1,178	31	5	98.1	99.3	7,066	2,638	175.4	202
48	0.5	1	1	28.1	99.8	2,303	686	1	1	54.1	99.6	4,200	1,869	19.9	1
60	0.1	7	2	134.3	99.6	10,624	2,306	5	2	124.5	99.4	10,202	2,627	226.6	347
60	0.2	58	4	149.7	99.5	11,287	2,033	27	5	101.1	99.4	8,228	2,195	96.2	148
60	0.3	111	8	150.5	99.4	11,740	1,856	30	6	170.6	99.4	11,890	2,824	1,719.6	3,494
60	0.4	32	4	63.6	99.6	5,068	1,044	39	5	158.2	99.5	11,080	2,690	234.7	295
60	0.5	1	1	57.3	99.9	4,708	798	1	1	69.8	99.8	5,250	1,597	30.3	1
72	0.1	15	3	94.2	99.7	7,284	1,386	21	4	103.2	99.4	7,092	2,353	214.0	143
72	0.2	72	3	110.4	99.3	8,694	1,560	31	5	145.3	98.6	10,402	3,418	18,654.8	18,559
72	0.3	37	3	62.2	99.6	4,916	1,249	51	5	173.3	99.2	11,046	2,911	1,187.9	802
72	0.4	22	3	149.5	99.5	11,679	1,922	29	5	176.3	98.7	12,950	3,667	18,347.1	14,435
72	0.5	28	4	122.7	99.4	9,923	2,094	106	6	206.9	98.6	13,965	3,736	1,565.3	3,022
90	0.1	11	3	149.7	99.6	11,597	2,251	31	3	163.0	99.0	13,666	3,271	108.0	17
90	0.2	101	12	314.7	99.4	24,126	3,176	121	6	157.3	98.6	13,847	3,503	2,531.1	3,323
90	0.3	128	8	179.5	99.2	13,966	2,422	109	6	230.9	98.4	19,223	3,840	5,859.6	8,702
90	0.4	70	7	228.1	99.5	16,989	2,650	253	7	244.5	97.5	20,300	4,793	1,792.2	2,702
90	0.5	28	4	107.7	99.6	8,309	1,386	28	5	185.6	99.2	14,684	3,538	2,468.9	3,647

TABLE 3.10: Comparison among **sepAll-bigM** BONMIN, **sepAll-bigM** CPLEX, and BONMIN Branch and Bound on configurations with 5 hydroplants and 30, 48, 60, 72, and 90 scenarios.

		Branch and Cut BONMIN						Branch and Cut CPLEX						BONMIN	
		B&B	Sep.		% time	NLP	Added	B&B	Sep.		% time	NLP	Added		
# scen.	α	nodes	nodes	Time (s)	NLP	solved	cuts	nodes	nodes	Time (s)	NLP	solved	cuts	Time (s)	Nodes
30	0.1	9	3	362.7	98.5	22,271	5,257	9	3	754.6	97.7	29,830	9,639	24.5	10
30	0.2	24	4	540.7	98.0	32,510	6,866	38	5	800.7	97.6	30,072	9,516	19.2	15
30	0.3	32	5	472.0	98.6	28,474	3,753	35	5	534.3	97.3	20,295	7,684	29.6	43
30	0.4	28	3	178.1	97.8	11,163	4,582	3	2	376.7	97.7	13,023	6,259	87.1	180
30	0.5	35	6	247.4	96.2	15,332	7,286	39	7	392.2	97.5	12,660	6,984	16.7	9
48	0.1	25	5	875.4	98.3	52,896	10,040	28	8	1,913.4	97.0	66,098	23,826	425.1	189
48	0.2	38	5	723.6	97.6	43,911	10,172	71	9	1,259.5	96.8	44,410	17,248	301.4	175
48	0.3	126	11	849.1	97.1	50,964	9,458	89	10	2,058.3	95.1	68,260	27,305	1,941.3	1,169
48	0.4	166	11	541.3	96.8	32,394	7,235	181	10	1,878.3	92.9	58,966	26,092	1,689.2	1,099
48	0.5	1	1	209.3	99.1	13,145	3,365	1	1	490.3	98.8	16,968	5,841	45.5	1
60	0.1	30	6	1,372.4	97.9	82,864	13,187	41	4	943.0	96.9	34,909	14,644	106.1	23
60	0.2	119	8	1,390.9	96.3	83,109	15,705	82	10	2,267.8	94.2	77,946	26,104	1,889.1	2,063
60	0.3	93	10	1,103.6	97.2	66,438	13,441	340	19	3,415.3	93.4	114,058	34,831	1,367.9	1,466
60	0.4	152	11	1,025.0	97.6	61,701	9,142	103	10	1,189.9	93.8	42,492	17,560	1,395.7	1,565
60	0.5	9	2	538.5	98.9	33,559	5,113	1	1	530.3	99.0	18,660	5,926	50.7	1
72	0.1	44	3	983.0	98.0	58,922	12,283	38	5	1,405.6	97.2	53,878	17,075	476.2	143
72	0.2	120	9	1,485.2	97.3	88,847	14,590	184	11	2,420.2	95.3	85,691	24,976	T.L.	16,039
72	0.3	157	10	1,571.1	96.6	93,986	17,617	91	8	1,670.1	95.9	57,966	25,648	2,572.8	1,205
72	0.4	149	13	1,131.9	97.5	68,315	13,386	164	12	2,681.2	94.4	91,467	31,718	7,745.0	2,975
72	0.5	196	15	1,319.2	95.7	77,908	16,556	149	11	2,379.7	90.6	78,660	27,969	205.4	102
90	0.1	45	4	1,744.6	98.3	104,781	13,898	42	5	1,216.1	96.2	50,353	17,448	205.2	18
90	0.2	217	15	3,675.0	96.8	218,367	26,881	208	11	4,053.7	91.8	145,394	42,084	5,716.7	3,756
90	0.3	203	16	2,000.0	96.2	118,623	17,354	295	13	3,394.1	90.8	118,098	37,627	1,615.8	1,051
90	0.4	272	21	2,794.0	96.1	163,791	21,959	270	16	3,758.6	92.0	128,264	39,488	4,846.4	2,953
90	0.5	341	22	1,956.5	94.7	114,643	19,276	536	45	3,232.8	93.7	113,101	35,454	2,893.1	1,846

Table 3.11: Comparison among **sepAll-bigM** BONMIN, **sepAll-bigM** CPLEX, and BONMIN Branch and Bound on configurations with 10 hydroplants and 30, 48, 60, 72, and 90 scenarios.

α	$E[\varphi]$	σ
0.00	561.0	198.9
0.05	595.3	203.1
0.10	600.3	211.5
0.20	588.3	252.5
0.30	594.0	257.4
0.40	582.6	257.7
0.50	518.7	330.1

TABLE 3.12: Expected profit in e M (second column) and standard deviation (third column) for different values of α.

3.4.4 The effect of α on the profit

We now discuss the trade-off between profit and risk allowed by our chance-constrained formulation for the mid-term hydro scheduling problem. Figure 3.4 shows, for several configurations of the system (1 to 10 hydroplants), the objective function value (quantile of the profit) of the solutions as a function of the level of risk α, restricted to the case of 90 scenarios. This allows the decision maker to easily evaluate not only the (minimum) profit they can obtain for a specified value of the risk, but also what profit they could expect by accepting a larger or smaller uncertainty. Of course, the objective function value obtained with a given α corresponds to the minimum profit that can be achieved with probability $1 - \alpha$, but the solution may be infeasible with probability α. In this section, $\alpha = 0.05$ is included in the comparison besides the α values tested above.

Once the problem is optimally solved for a specific level of risk α, the decision maker can also evaluate the distribution of the profits associated with the different scenarios. Indeed, a solution to the master problem specifies a value for the flow variables: this allows us to compute the associated profit for all satisfied scenarios, and also for those unsatisfied scenarios for which the flow variables define a physically feasible solution (i.e., those scenarios for which the water balance constraints are satisfied, but constraints (3.12) are not). Figure 3.5 depicts the inverse distribution function of the profit for the case of 10 hydroplants. We remark that here, and in the computation of expected profits below, we are assuming that the profit is zero whenever a solution violates the water conservation constraints. The solution obtained with $\alpha = 0$ (all scenarios are satisfied) achieved a profit that is consistently below the other solutions, except for scenarios when the other solutions are infeasible. As expected, there is a spike in each curve when the value on the x-axis corresponds to the level of risk α being optimized. It is interesting to note that even a risk-averse solution ($\alpha = 0.05$) achieves a profit that is relatively similar to the least risk-averse solution ($\alpha = 0.5$), although in the most favorable scenarios (right part of the graph), $\alpha = 0.5$ typically yields better profit than $\alpha = 0.05$. On the other hand, for the most unfavorable scenarios, up to a cumulative probability of almost 0.5, the solutions with $\alpha = 0.05$ and $\alpha = 0.1$ perform

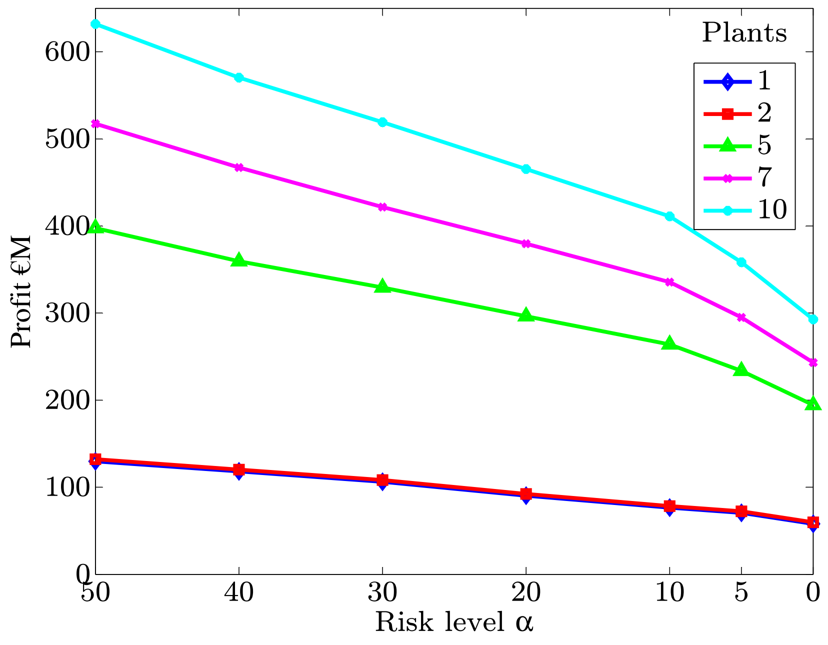


FIGURE 3.4: Trade-off between profit in €M and level of risk: the x-axis reports the risk level α, and the y-axis the corresponding objective function value.

much better than with $\alpha = 0.5$. Solutions obtained with $\alpha \in \{0.2, 0.3, 0.4\}$ are similar to each other, and they all perform worse than $\alpha = 0.1$ for a cumulative probability of up to 0.2, as expected, but perform better in the most favorable scenarios, achieving approximately 50M higher profit in some cases. Table 3.12 reports the expected profit and the standard deviation of the solutions corresponding to the tested values of α. We can see that relaxing some of the constraints with small probability (≤ 0.05) yields an increase of the expected profit by 6.1% as compared to the solution with $\alpha = 0$, although unsurprisingly this comes at the cost of a slightly larger standard deviation. The highest expected profit is achieved with $\alpha = 0.1$, where the increase is of 7% as compared to $\alpha = 0$. Allowing constraint violations with higher probability produces infeasible solutions in a larger number of scenarios, and the corresponding lack of profit decreases the expected gain. When α is very large ($\alpha = 0.5$), the solution obtained is infeasible for many scenarios, leading to an expected profit almost 10% lower than the conservative solution with $\alpha = 0$.

Summarizing, our computational experiments indicate that introducing a moderate amount of flexibility in the formulation, namely by allowing some constraints to be violated with small probability (0.05 or 0.1), can increase the expected profit by a significant amount. However, there are diminishing returns of increasing α, and when the allowed probability of violating the constraints becomes too large, the resulting trade-off between risk and rewards seems to be unfavorable, yielding a considerable drop in the expected profit.

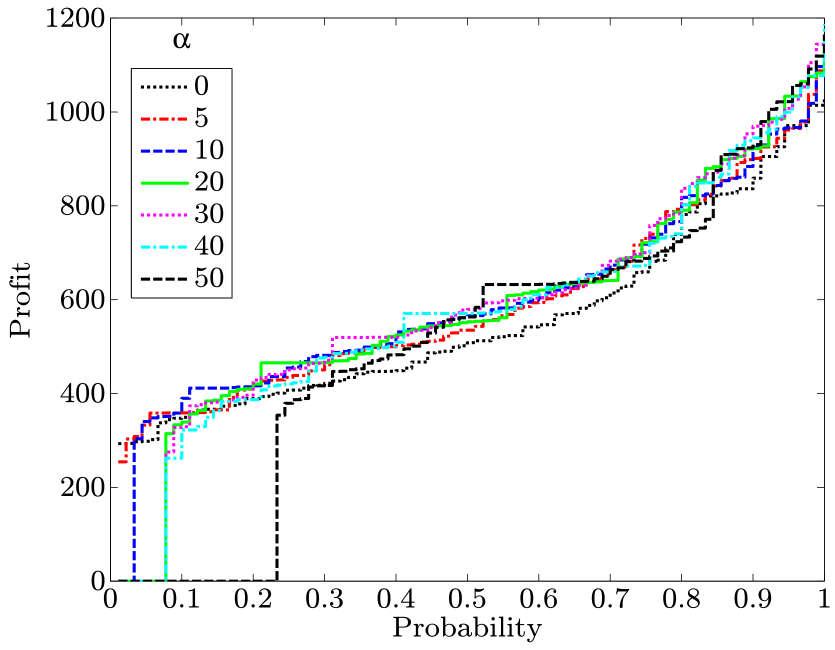


FIGURE 3.5: Inverse distribution function of the profit.

3.4.5 Other solvers

As remarked in Section 3.4.2, we tried to solve the MINLP (3.2) with additional solvers based on other solution methods, namely, the BONMIN Outer Approximation algorithm, the BONMIN hybrid algorithm and the FilMINT Branch-and-Cut algorithm. None of the mentioned solvers could consistently handle the MINLP (3.2), and all solvers were plagued by severe numerical issues; as a consequence, they could correctly solve only small instances or instances with simplified nonlinear functions. If a numerical issue is incurred, the corresponding value is marked with a $*$ in the tables.

In Table 3.13 we report detailed results for a subset of instances of increasing complexity, comparing the BONMIN Branch and Bound (B-BB) with the BONMIN Outer Approximation algorithm (B-OA), the BONMIN hybrid algorithm (B-Hyb) and the FilMINT Branch-and-Cut algorithm. All instances in the table have 90 scenarios and the time limit is set to 10 hours for every solver and instance. The table reports the number of hydroplants and the level of risk α in the first two columns. Subsequent columns report the total CPU time and the number of Branch-and-Bound nodes.

The BONMIN Outer Approximation algorithm incurs in numerical issues in the majority of the tested instances. Instead the BONMIN hybrid algorithm (B-Hyb) and the FilMINT Branch-and-Cut algorithm solve very few instances to optimality.

Similar considerations can be drawn from Tables 3.14 and 3.15, where all the instances are for the 5 and the 10 hydroplants configuration, and different number of scenarios, as reported in the first column.

		B_BB		B_OA	B_Hyb		FilMINT
Plants	α	Time (s)	nodes	Time (s)	Time (s)	nodes	Time (s)
1	90	155.6	365	368.5	T.L.	815,311	6,930.0
1	80	2,775.4	14,173	996.4	T.L.	355,751	T.L.
1	70	2,291.6	12,955	2.3*	2.7*	1	T.L.
1	60	4,331.8	26,144	6,813.4	2.7*	1	T.L.
1	50	3,691.6	22,856	0.0*	T.L.	1,430,981	T.L.
2	90	1,836.7	2,606	0.0*	0.0*	1	9.6*
2	80	14,279.8	24,749	0.0*	0.0*	1	13.0*
2	70	T.L.	29,606	0.0*	0.0*	1	880.1*
2	60	6,326.3	20,554	0.0*	T.L.	1,311,775	T.L.
2	50	7,996.1	27,761	238.5*	85.8	949	26,515.4*
5	90	2,471.5	3,060	0.0*	0.0*	1	2,509.1*
5	80	5,331.1	6,255	451.6	T.L.	657,584	14,811.7*
5	70	13,086.6	17,800	492.4*	T.L.	262,137	T.L.
5	60	9,376.6	11,745	67.7*	T.L.	265,862	T.L.
5	50	8,011.3	10,742	31,899.7*	T.L.	352,355	T.L.
7	90	9,554.3	7,001	0.0*	0.0*	1	1,253.2*
7	80	7,107.8	6,338	0.0*	17,625.6	127,802	T.L.
7	70	5,776.8	4,445	6.0*	T.L.	191,250	T.L.
7	60	16,619.1	13,397	4.9*	T.L.	191,292	T.L.
7	50	7,520.0	6,159	0.0*	T.L.	286,059	T.L.
10	90	4,520.6	2,189	0.0*	0.0*	1	10,111.1*
10	80	9,186.7	4,811	0.0*	T.L.	138,781	T.L.
10	70	14,136.2	6,380	18.4*	T.L.	114,764	T.L.
10	60	13,818.4	7,035	12.5*	T.L.	137,012	T.L.
10	50	T.L.	17,077	9.4*	T.L.	211,111	T.L.

Table 3.13: Comparison among the BONMIN Branch and Bound (B-BB), the BONMIN Outer Approximation algorithm (B-OA), the BONMIN hybrid algorithm (B-Hyb) and the FilMINT Branch-and-Cut algorithm on configurations of 1, 2, 5, 7, and 10 hydroplants and 90 scenarios.

		B_BB		B_OA	B_Hyb		FilMINT
# scen.	α	Time (s)	nodes	Time (s)	Time (s)	nodes	Time (s)
30	0.1	19.7	5	0.0*	0.0*	1	3.2*
30	0.2	71.6	276	38.9	38.7	651	14.5*
30	0.3	203.5	878	29.1	196.7	6,775	63.0*
30	0.4	51.9	191	22.8*	240.2	8,819	77.8
30	0.5	60.5	218	13.8*	570.1	15,602	27.3
48	0.1	238.1	221	0.0*	0.0*	1	8.8*
48	0.2	2,504.6	3,809	150.7	279.3	9,884	740.3*
48	0.3	1,748.8	2,609	2,768.3	95.2	4,520	2,003.7*
48	0.4	159.6	249	183.2*	7,973.2	298,097	4,760.7*
48	0.5	492.4	26	193.4*	T.L.	785,408	1,070.9*
60	0.1	132.0	57	0.0*	0.0*	1	26.0*
60	0.2	454.2	846	364.1	2,136.8	51,302	2,592.1*
60	0.3	4,351.6	2,605	22,297.3	20,694.7	266,543	3,501.8*
60	0.4	4,351.6	8,621	26.8*	T.L.	431,014	T.L.
60	0.5	377.3	720	2,390.2*	T.L.	449,348	8,103.7
72	0.1	408.0	279	0.0*	0.0*	1	79.5*
72	0.2	T.L.	29,114	408.0*	22,075.3	662,902	T.L.
72	0.3	T.L.	27,436	0.0*	T.L.	754,885	T.L.
72	0.4	T.L.	25,633	0.0*	T.L.	480,527	T.L.
72	0.5	1,812.1	2,836	48.8*	T.L.	253,616	T.L.
90	0.1	2,471.5	3,060	0.0*	0.0*	1	2,509.1*
90	0.2	5,331.1	6,255	451.6	T.L.	657,584	14,811.7*
90	0.3	13,086.6	17,800	492.4*	T.L.	262,137	T.L.
90	0.4	9,376.6	11,745	67.7*	T.L.	265,862	T.L.
90	0.5	8,011.3	10,742	31,899.7*	T.L.	352,355	T.L.

Table 3.14: Comparison among the BONMIN Branch and Bound (B-BB), the BONMIN Outer Approximation algorithm (B-OA), the BONMIN hybrid algorithm (B-Hyb) and the FilMINT Branch-and-Cut algorithm on configurations with 5 hydroplants and 30, 48, 60, 72, and 90 scenarios.

		B_BB		B_OA	B_Hyb		FilMINT
# scen.	α	Time (s)	nodes	Time (s)	Time (s)	nodes	Time (s)
30	90	27.2	7	5.0	12.2	13	43.4*
30	80	39.7	34	80.1	22.1	77	258.4*
30	70	85.0	117	2.9*	104.5	1,105	598.9
30	60	60.1	85	2.0*	79.9	811	391.6
30	50	36.0	30	1.9*	684.4	6,211	189.5
48	90	868.9	347	1.7*	16.2	13	340.0*
48	80	6,286.4	2,029	3.4*	661.9	10,250	6,603.0
48	70	2,301.4	1,169	5.5*	608.2	12,875	2,951.0*
48	60	5,785.1	2,907	2.9*	6,034.6	114,266	7,656.4
48	50	52.6	1	174.6	80.4	377	4,663.9
60	90	1,299.7	615	1.7*	123.3	371	3,031.9*
60	80	1,472.1	891	1.9*	3,155.3	22,617	21,306.8*
60	70	1,111.1	746	3.8*	14,440.1	103,331	T.L.
60	60	1,155.8	769	7.5*	T.L.	189,921	T.L.
60	50	406.7	329	241.9*	T.L.	257,795	T.L.
72	90	7,896.0	2,692	11,822.9	869.4	6,827	2,363.1*
72	80	T.L.	8,575	2.6*	3,736.7	46,970	T.L.
72	70	T.L.	11,505	11.0*	T.L.	468,884	T.L.
72	60	T.L.	10,471	12.2*	T.L.	304,701	T.L.
72	50	4,421.3	2,860	2,806.5*	T.L.	336,265	T.L.
90	90	4,520.6	2,189	0.0*	0.0*	1	10,111.1*
90	80	9,186.7	4,811	0.0*	T.L.	138,781	T.L.
90	70	14,136.2	6,380	18.4*	T.L.	114,764	T.L.
90	60	13,818.4	7,035	12.5*	T.L.	137,012	T.L.
90	50	T.L.	17,077	9.4*	T.L.	211,111	T.L.

Table 3.15: Comparison among the BONMIN Branch and Bound (B-BB), the BONMIN Outer Approximation algorithm (B-OA), the BONMIN hybrid algorithm (B-Hyb) and the FilMINT Branch-and-Cut algorithm on configurations with 10 hydroplants and 30, 48, 60, 72, and 90 scenarios.

3.5 Conclusions

We have proposed a Branch-and-Cut algorithm for a class of nonlinear chance-constrained mathematical optimization problems with a finite number of scenarios. The algorithm is based on an implicit Benders decomposition scheme, where we generate cutting planes as outer approximation constraints from the projection of the feasible region on suitable subspaces.

The algorithm has been theoretically analyzed and computationally evaluated on a mid-term hydro scheduling problem by using data from ten hydroplants in Greece. We have shown that the proposed methodology is capable of solving instances orders of magnitude faster than applying a general-purpose solver for convex mixed-integer nonlinear programming problems to the deterministic reformulation, and scales much better with the number of scenarios.

From the economical standpoint, our numerical experiments have shown that the introduction of a small amount of flexibility in the formulation, by allowing constraints to be violated with a joint probability $\leq 5\%$, increases the expected profit by 6.1% on our dataset.

Bibliography

K. Abhishek, S. Leyffer, and J. Linderoth. FilMINT: An outer approximation-based solver for convex mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 22(4):555–567, 2010.

R. Alvarez-Valdes, A. Parajon, and J. M. Tamarit. A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Comput. Oper. Res.*, 29(7):925–947, 2002.

C. Arbib, F. Marinelli, F. Rossi, and F. di Iorio. Cutting and reuse: an application from automobile component manufacturing. *Operation Research*, 50(6):923–934, 2002.

A. Atamtürk, G.L. Nemhauser, and M. W.P. Savelsbergh. The mixed vertex packing problem. *Mathematical Programming*, 89(1):35–53, 2000.

L. Bacaud, A. Lemarèchal, C. Renaud, and C. Sagastizábal. Bundle methods in stochastic optimal power management: A disaggregated approach using preconditioners. *Computational Optimization ans Applications*, 20(3):227–244, 2001.

G. C. Baslis and G. A. Bakirtzis. Mid-term stochastic scheduling of a price-maker hydro producer with pumped storage. *IEEE Transactions on Power Systems*, 26(4): 1856–1865, 2011.

S. Ben Messaoud, C. Chu, and M. Espinouse. Characterization and modelling of guillotine constraints. *European Journal of Operational Research*, 191(1):112–126, 2008.

J. A. Bennell, J. F. Oliveira, and G. Wäscher. Cutting and packing. *International Journal of Production Economics*, 145(2):449–450, 2013.

D. P. Bertsekas. *Nonlinear Programming, 2nd Edition.* Athena Scientific, Belmont, MA, 1999.

D. P. Bertsekas, G. S. Lauer, N. R. Jr. Sandell, and T. A. Posbergh. Optimal short-term scheduling of large-scale power systems. *IEEE Transactions on Automatic Control*, 28(1):1–11, 1983.

D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization (Athena Scientific Series in Optimization and Neural Computation, 6).* Athena Scientific, 1997.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.

J. A. Bloom. Solving an electricity generating capacity expansion planning problem by generalized Benders' decomposition. *Operations Research*, 31(1):84–100, 1983.

P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex Mixed Integer Nonlinear Programs. *Discrete Optimization*, 5:186–204, 2008.

P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for Mixed Integer Nonlinear Programs. *Mathematical Programming*, 119(2):331–352, 2009.

P. Bonami, A. Lodi, A. Tramontani, and S. Wiese. On mathematical programming with indicator constraints. *Mathematical Programming*, 151(1):191–223, 2015.

M. Boschetti, E. Hadjiconstantinou, and A. Mingozzi. New upper bounds for the two-dimensional orthogonal non guillotine. *IMA Journal of Management Mathematics*, 13(2):95–119, 2002.

E. Burke, R. Hellier, G. Kendall, and G. Whitwell. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operation Research*, 54(3):587–601, 2006.

A. Caprara and M. Monaci. On the two-dimensional knapsack problem. *Operations Research Letters*, 32(1):5–14, 2004.

A. A. F. M. Carneiro, S. Soares, and P. S. Bond. A large scale of an optimal deterministic hydrothermal scheduling algorithm. *IEEE Transactions on Power Systems*, 5 (1):204–211, 1990.

P. L. Carpentier, M. Gendreau, and F. Bastin. Midterm hydro generation scheduling under uncertainty using the progressive hedging algorithm. Technical Report 2012-35, CIRRELT, 2012.

H. C. Chang and P. H. Chen. Hydrothermal generation scheduling package: a genetic based approach. *IEE Proceedings - Generation, Transmission and Distribution*, 145 (4):451–457, 1998.

A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963.

A. Charnes, W. W. Cooper, and G. H. Symonds. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, 4(3):235–263, 1958.

Y. Chen. A recursive algorithm for constrained two-dimensional cutting problems. *Computational Optimization and Applications*, 41(3):337–347, 2008.

N. Christofides and E. Hadjiconstantinou. An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. *European Journal of Operational Research*, 83(1):21–38, 1995.

N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operational Research*, 25(1):30–44, 1977.

G. Cintra and Y. Wakabayashi. Dynamic programming and column generation based approaches for two-dimensional guillotine cutting problems. In CelsoC. Ribeiro and SimoneL. Martins, editors, *Experimental and Efficient Algorithms*, volume 3059 of *Lecture Notes in Computer Science*, pages 175–190. Springer Berlin Heidelberg, 2004.

F. Clautiaux, A. Jouglet, and A. Moukrim. A new graph-theoretical model for the guillotine-cutting problem. *INFORMS Journal on Computing*, 25(1):72–86, 2013.

E. G. Coffman, M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9 (4):808–826, 1980.

V-D. Cung, M. Hifi, and B. Le Cun. Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *nternational Transactions in Operational Research*, 7(3):185–210, 2000.

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

M. Duran and I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.

H. Dyckhoff. A new linear programming approach to the cutting stock problem. *Operations Research*, 29(6):1092–1104, 1981.

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

M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders,Äô cuts. *Mathematical Programming*, 124(1-2):175–182, 2010.

R. Fletcher and S. Leyffer. Solving Mixed Integer Nonlinear Programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994.

P. M. França and H. P. L. Luna. Solving stochastic transportation-location problems by generalized Benders decomposition. *Transportation Science*, 16(2):113–126, 1982.

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

F. Furini and E. Malaguti. A pseudo-polynomial size formulation for 2-stage 2-dimensional knapsack problems. In *Proceedings of the 45th International Conference on Computers & Industrial Engineering (CIE 45)*, 2015.

F. Furini, E. Malaguti, and D. Thomopulos. Modeling two-dimensional guillotine cutting problems via integer programming. Technical Report OR-14-25, DEI - University of Bologna, 2014.

D. Gade, S. Küçükyavuz, and S. Sen. Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64, 2014.

A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

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

P. C. Gilmore and R. E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965.

Glass Alliance Europe. The eu glass industry in 2014-2015, 2015. URL http://www.glass-international.com/contentimages/subscriber-pdf/Glass_Alliance.pdf.

O. Günlük and Y. Pochet. Mixing mixed-integer inequalities. *Mathematical Programming*, 90(3):429–457, 2001.

J. C. Herz. Recursive computational procedure for two-dimensional stock-cutting. *IBM Journal of Research Development*, 16(5):462–469, 1972.

M. Hifi. An improvement of viswanathan and bagchi's exact algorithm for constrained two-dimensional cutting stock. *Computers & Operations Research*, 24(8):727–736, 1997.

M. Hifi. Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems. *Journal of Combinatorial Optimization*, 8(1): 65–84, 2004.

M. Hifi and C. Roucairol. Approximate and exact algorithm for constrained (un)weighted two-dimensional two-staged cutting stock problems. *Journal of Combinatorial Optimization*, 5(4):465–494, 2001.

International Energy Agency. *Key World Energy Statistics 2015*. IEA, 2015.

M. Iori, J. J. Salazar González, and D. Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41(2): 253–264, 2007.

R. G. Jeroslow. Representability in mixed integer programming, I: characterization results. *Discrete Applied Mathematics*, 17(3):223–243, 1987.

C. Jordan. *Batching and Scheduling: Models and Methods for Several Problem Classes.* Lecture Notes in Economics and Mathematical Systems 437. Springer-Verlag Berlin Heidelberg, 1 edition, 1996.

M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey. *50 years of integer programming 1958-2008: From the early years to the state-of-the-art.* Springer-Verlag Berlin Heidelberg, 1 edition, 2010.

J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society of Industrial and Applied Mathematics*, 8(4):703–712, 1960.

M. P. N. C. R. Kelman. Long-term hydro scheduling based on stochastic models. *EPSOM*, 98:23–25, 1998.

M. A. Lejeune. Pattern-based modeling and solution of probabilistically constrained optimization problems. *Operations Research*, 60(6):1356–1372, 2012.

X. Liu, S. Küç̈kyavuz, and J. Luedtke. Decomposition algorithms for two-stage chance-constrained programs. *Mathematical Programming*, pages 1–25, 2014.

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

A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: a survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

A. Lodi, S. Martello, M. Monaci, C. Cicconetti, L. Lenzini, E. Mingozzi, C. Eklund, and J. Moilanen. Efficient two-dimensional packing algorithms for mobile wimax. *Management Science*, 57(12):2130–2144, 2011.

A. Lodi, E. Malaguti, G. Nannicini, and D. Thomopulos. Nonlinear chance-constrained problems with applications to hydro scheduling. Technical Report OR-15-9, DEI - University of Bologna, 2016.

J. Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, 146:219–244, 2014.

J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008.

D. G. Luenberger and Y. Ye. *Linear and nonlinear programming.* International series in operations research and management science. Springer, 3rd ed edition, 2008.

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

E. Malaguti, R. Medina Durán, and P. Toth. Approaches to real world two-dimensional cutting problems. *Omega*, 47:99–115, 2014.

S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations.* Wiley, 1990.

A. J. McNeil, R. Frey, and P. Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools (Revised Edition).* Princeton Series in Finance. Princeton University Press, Princeton, NJ, 2015.

OR-Library. Or-library. http://people.brunel.ac.uk/ mastjjb/jeb/info.html.

M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, unabridged edition, 1998.

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

A. Prekopa. On probabilistic constrained programmming. In H. W. Kuhn, editor, *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138, Princeton, NJ, 1970. Princeton University Press.

J. Puchinger and G. R. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304–1327, 2007.

F. Qiu, S. Ahmed, and L. A. Dey, S. S.and Wolsey. Covering linear programming with violations. *INFORMS Journal on Computing*, 26(3):531–546, 2014.

M. Russo, A. Sforza, and C. Sterle. An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems. *Computers & Operations Research*, 50:97–114, 2014.

S. Ružić, N. Rajaković, and A. Vučković. A flexible approach to short-term hydro-thermal coordination. I. Problem formulation and general solution procedure. *IEEE Transactions on Power Systems*, 11(3):1564–1571, 1996.

M. S. Salam, K. M. Nor, and A. R. Hamdan. Hydrothermal scheduling based lagrangian relaxation approach to hydrothermal coordination. *IEEE Transactions on Power Systems*, 13(1):226–235, 1998.

A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Inc., 1986.

S. Shen, J. C. Smith, and S. Ahmed. Expectation and chance-constrained models and algorithms for insuring critical paths. *Management Science*, 56(10):1794–1814, 2010.

E. Silva, F. Alvelos, and J. M. Valério de Carvalho. An integer programming model for two- and three-stage two-dimensional cutting stock problems. *Computers & Operations Research*, 205(3):699–708, 2010.

Y. Song, J. Luedtke, and S. Küçükyavuz. Chance-constrained binary packing problems. *INFORMS Journal on Computing*, 26(4):735–747, 2014.

M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra. Large-scale unit commitment under uncertainty. *4OR*, 13(2):115–171, 2015.

M. W. Tanner, L. Sattenspiel, and L. Ntaimo. Finding optimal vaccination strategies under parameter uncertainty using stochastic programming. *Mathematical Biosciences*, 215(2):144–151, 2008.

M. Trick. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research*, 118(1):73–84, 2003.

J. M. Valério de Carvalho. LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253–273, 2002.

W. van Ackooij. Decomposition approaches for block-structured chance-constrained programs with application to hydro-thermal unit commitment. *Mathematical Methods of Operations Research*, 80(3):227–253, 2014.

F. Vanderbeck. A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem. *Management Science*, 47(6):864–879, 2001.

Q. Wang, Y. Guan, and J. Wang. A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output. *Power Systems, IEEE Transactions on*, 27(1):206–215, 2012.

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

T. Watanabe and H. Ellis. Stochastic programming models for air quality management. *Computers & Operations Research*, 20(6):651–663, 1993.

T. Westerlund, H. Skrifvars, I. Harjunkoski, and R. Pörn. An extended cutting plane method for a class of non-convex MINLP problems. *Computers & Chemical Engineering*, 22(3):357–365, 1998.