Instituto de Matemática Pura e Aplicada

# Algorithms for Some Hierarchical Mathematical Optimization Problems

Pedro Henrique Borges de Melo

Advisor: Mikhail Solodov - IMPA
Co-advisor: Claudia Sagastizábal - UNICAMP

July 12, 2021

impa

Instituto de Matemática
Pura e Aplicada

This page is intentionally left blank.

# Abstract

Hierarchical optimization problems are those whose feasible sets and/or objective function involve solution mappings and/or value functions of other optimization problems. Such problems appear naturally when one is modeling games or defining decomposition algorithms for non-convex large-scale optimization.

Hierarchical problems have peculiar features that make them hard to solve with classical optimization because assumptions that are usually employed in the convergence analysis of algorithms are not satisfied in the hierarchical setting. For instance, neither convexity nor standard constraint qualification conditions can be relied upon. Also, smoothness of problem data cannot be assumed because value functions or solution mappings are not smooth for most of the cases of interest. Moreover, in general there are no explicit formulas for value functions or solution mappings, which makes the calculation of generalized derivatives harder or impossible with current results available in the literature.

An essential question that arises when trying to deal with hierarchical problems is the one of computing variations for value functions and solutions mappings in such a way that these variations can be put into an algorithmic pattern that actually solves the problem in practice. Given that high-quality and reliable optimization software is hard to develop, it would be better if we could solve these hierarchical problems as limits of classical problems or by extensions of classical techniques to leverage on existing high-quality classical methods.

The first part of this thesis approximates some hierarchical problems as limits of classical problems via a smoothing technique, which is applied for the solution of non-convex two-stage stochastic programs and for decomposing a class of stochastic equilibrium problems. The smoothing method is analyzed theoretically and compared numerically against state-of-the-art techniques.

The second part proposes an extension of the classical Benders cuts, termed *free floating cuts*, that are used to compute variations for value functions of multistage stochastic optimization problems. Such floating cuts are employed to evaluate efficiently the sensitivity of these multistage stochastic programs with respect to right-hand side uncertainty and with respect to the decisions of other players in a multistage stochastic game. The corresponding cuts are employed in a sequential sampling procedure and in a best-response algorithm for the stochastic game. Moreover, a solution technique for a deterministic trilevel game motivating the stochastic game is developed. The approaches are satisfactorily applied to the management of a cascade of water reservoirs used to produce electricity.

**Keywords:** Hierarchical Optimization, Non-smooth Optimization, Interior Penalty Methods, Tikhonov Regularization, Smoothing Methods, Benders Cuts, Cutting-Plane Methods, Stochastic Dual Dynamic Programming, Sequential Sampling

**AMS Subject Classification:** 90C15 65K10 90C39 90C31 90C06.

This page is intentionally left blank.

# Acknowledgments

I would like to thank my advisors Mikhail Solodov and Claudia Sagastizábal for their continued effort during these years. They provided me with the technical problems to which this thesis is dedicated as well as excellent guidance. They also pointed the right directions and made the corrections when needed. Without their professional attitude and commitment towards excellence, this work would not have been possible. Their working ethic is a reference to me.

I am also in debt with Leo Liberti and Claudia D'Ambrósio from École Polytechnique (France) and Asgeir Tomasgard from the Norwegian University of Science and Technology (Norway) for having received me as a visiting student. These stays were extremely helpful for the development of this thesis, as they gave me the opportunity to participate in more conferences and talk to more people in the field.

I am also grateful to Instituto Nacional de Matemática Pura e Aplicada (IMPA), where I first arrived for a summer course in 2014 and stayed since. From the beginning I was impressed with the challenging environment and the prosperous professional culture that the institute manages to foster on students and professors.

I would like to thank the members of the evaluation committee, which read this thesis carefully and suggested many small corrections. Specifically, I would like to thank Sandra Santos from UNICAMP for her contributions.

I also thank CNPq for the PhD scholarship provided, as well as École Polytechnique for the Gaspard Monge scholarship and the Norwegian University of Science and Technology for the Transatlantic Cooperation on Energy Markets Modeling (TACEMM) scholarship.

I would like to thank my mother, Emília Borges, for all her efforts and incredible support during all these years. I would also like to thank all my family and friends for their helping arms and companionship. Specially, I would like to thank my sisters Lara Borges and Bárbara Borges. I am extremely in debt with my grandfather Agostinho Borges, who I am sure would have been extremely proud of this thesis.

# Agradecimentos

Eu gostaria de agradecer aos meus orientadores Mikhail Solodov and Claudia Sagastizábal pelos seus esforços continuados ao longo desses anos de estudo. Eles forneceram os problemas técnicos abordados nessa tese, bem como uma excelente orientação. Eles também apontaram as direções corretas e fizeram correções quando foi preciso. A cultura de trabalho deles é uma referência pra mim.

Também sou agradecido ao Leo Liberti e à Claudia Ambrósio da École Polytechnique (França) e ao Asgeir Tomasgard da Norwegian University of Science and Technology (Noruega) por terem me recebido como um estudante visitante. Essas visitas foram extremamente úteis para o desenvolvimento dessa tese, pois me deram a oportunidade de participar de mais conferências e falar com mais pessoas da área.

Também sou extremamente grato ao Instituto Nacional de Matemática Pura e Aplicada (IMPA), onde eu cheguei pela primeira vez para um curso de verão em 2014 e fiquei desde então. Desde o início eu fiquei impressionado com o ambiente desafiador e a cultura profissional próspera que o instituto consegue fomentar nos estudantes e professores.

Gostaria de agradecer aos membros da banca avaliadora, os quais leram o trabalho cuidadosamente e sugeriram várias correções. Em especial, gostaria de agradecer à Sandra Santos da UNICAMP pelas suas contribuições.

Também sou agradecido ao CNPq pela bolsa de estudos, bem como à École Polytechnique pela bolsa Gaspard Monge e também à Norwegian University of Science and Technology pela bolsa Transatlantic Cooperation on Energy Markets Modelling (TACEMM).

Gostaria também de agradecer minha mãe, Emília Borges, por todos os esforços e apoio ao longo de todos esses anos. Também agradeço todos familiares e amigos pela mão firme e companhia. Especialmente, gostaria de agradecer minhas irmãs Lara Borges e Bárbara Borges. Sou extremamente grato ao meu avô, Agostinho Borges, que certamente estaria muito orgulhoso dessa tese.

This page is intentionally left blank.

# Contents

This page is intentionally left blank.

# Chapter 1

# Introduction

In this chapter we introduce the types of optimization problems around which the thesis is developed and motivate the need for the techniques proposed to solve them. The problems considered have features that make the direct application of smooth optimization methods either not possible or not practical. It would not be possible because convergence would not be achieved or there are no explicit formulas to differentiate, even in the generalized sense. It would not be practical because the resulting problems would not even fit into the memory of a computational cluster. Such problems arise in many situations, including when we are modeling games or when devising decomposition algorithms for large-scale stochastic optimization.

## 1.1 Motivation

To start, let us consider a decision process having two temporal stages: today and tomorrow. A here-and-now decision needs to be taken today, but the implied cost (also called future cost) associated with such decision can be measured only at the next stage (tomorrow). A commonly used paradigm is to take a decision today that minimizes the sum of the cost of taking the decision today and the mathematical expectation of the total cost measured tomorrow. Such paradigm, also referred to as Two-Stage Stochastic Optimization Problem (2TSP), and its extensions for more temporal stages, found numerous applications in energy planning [PP91; MMF11].

The considered setting has some major complications that need to be addressed so that the underlying formulation can actually be solved. First, a suitable representation for the future cost of taking the here-and-now decision needs to be obtained. For instance, in [Kel60; OSS11] the Benders cut is used because the future costs are convex with respect to the here-and-now decision. Second, the representation of the future cost needs to be such that the here-and-now decision can be chosen efficiently. In [OSS11] the representation of the future cost can be coupled into a linear or quadratic programming problem. Third, the representation of the expected future costs does not depend on the number of different uncertainty realizations that might take place tomorrow (the number of scenarios). Fourth, the future cost is usually not a smooth function of the here-and-now variable and therefore special solution methods need to be employed.

The situation changes substantially when the expected future cost is not convex with respect to the here-and-now decision, since, in principle, it is not clear how to obtain a suitable representation of the future cost such that the here-and-now variable can be chosen efficiently. The alternative provided by classical smooth non-convex optimization is to employ the so-called deterministic equivalent formulation [Wet66; DRS09]. However, the resulting problem is too large to actually be solved in practice when the number of scenarios is large.

Other hierarchical problems with a structure similar to the 2TSP, are the Deterministic and Stochastic Walrasian Equilibrium Problems (WEP). See [DJW17; JJBW02]. The deterministic formulation considers many agents that can observe the prices of the goods available for buying and according to the prices, decide how much of each good to buy. Each agent is assumed to have a known concave utility function modeling its preference for each good. For instance, if wine is expensive, some agents do not buy it, while most agents would buy rice. The problem consists in finding the prices of the goods such that the total demand for each good is equal to the offer. The stochastic version tries to find the prices in each scenario such that the demand for each product is equal to the supply for all scenarios.

The WEP can be thought of as having a here-and-now decision taken by a market coordinator and the effect of this decision is observed afterwards as a measurement of the imbalance between supply and demand for each good, where the imbalance represents the future cost. In contrast to the 2TSP, where the future cost is given by an average of optimal values, the future cost is now given as a function of the decisions of each agent (the demand). In summary, for one problem (2TSP) the future costs involve *value functions* and for the other (WEP) *solution mappings* of parametric optimization problems. However, the time structure of the decision process can be considered the same for both. Moreover, if one tries to employ classical smooth optimization, the size of the resulting problem is proportional to the number of agents taking decisions and to the number of scenarios. In addition, the reformulation needs to deal with the so-called complementarity conditions [IS14], which violates important assumptions commonly used.

The similarities and differences between the 2TSP and the WEP point to a solution technique that is effective for both, since it allows for a representation of the future costs with which we can compute approximations for the here-and-now decision efficiently. Moreover, it circumvents the difficulties associated to the large-scale nature of both problems as well as the non-smoothness of both solution mappings and value functions. The 2TSP and the WEP motivate the first part of the thesis.

The second part of this work is also about useful and efficient representations of value functions, but mostly in the context of multi-stage stochastic optimization problems and games.

The specific application that gives rise to the developments is the management of a cascade of water reservoirs that are used to produce electricity, called the Cascade Management Problem (CMP). Since this problem has many specific features, we discuss it carefully. The CMP can be formulated in two variants. The first casts the problem as a standard multi-stage stochastic program and considers that there is only one decision-maker managing alone the entire cascade. The second is a multi-stage stochastic game and considers that each reservoir is managed separately by an agent that maximizes its own profit. Each variant is explained more in the sequel and receives a different focus.

The first aspect of the CMP is that both variants have as input the scenarios for the rainwater that arrives to each reservoir. Such scenarios are obtained via the discretization of a continuous probability distribution believed to model realizations of the rainwater. When the number of reservoirs is huge (for instance, 85 for the Brazilian system) and we consider the first variant, two issues come into play, namely (i) the solution of a single instance of the problem may take hours even if employing tens of servers and (ii) since the solution of one instance is already hard, it is not possible to evaluate if the number of scenarios used is really meaningful, because solving for another set of scenarios is too time consuming. Since the efficient evaluation of the quality of the policies obtained with respect to rainwater scenarios is important for the first variant, it also has to be important for the second variant, which we explain now.

The second variant considers selfish players that care only about their own profit. Therefore, the agent at the top of the cascade usually tries to withhold water when electricity is cheap to release this water (for producing energy) when electricity is expensive. Such individualistic behavior decreases significantly the profits for the reservoirs located downstream. In principle, the society would like to put in place a mechanism such that the cascade managed by the selfish players produces the same amount of wealth as if it is managed jointly by one agent that tries to maximize the overall wealth obtained by the whole cascade. In other words, one has to monetize the collaboration efforts between the players such that they start collaborating effectively.

When the rainwater inflow is deterministic, the second variant can be solved easily as it can be modeled by a trilevel optimization problem. However, the monetization employed to measure the collaboration between players is relative to the profits they would obtain when acting without collaborating at all. Therefore, the computation of the individualistic strategies of each player is a key ingredient to monetize the collaboration effectively. It is important to be aware of this fact so that one can make sense of the developments for the stochastic setting (when rainwater is not deterministic).

The stochastic setting comes with its own complications since the problem is large-scale and harder to conceptualize from scratch, as not even the formulations are written somewhere else in the literature. However, the technical questions are (i) how to compute individualistic solutions and (ii) how to formulate mathematically the collaboration between players and solve the resulting model. We adress these points in the second part of the manuscript.

## 1.2 Relation with other works

In this section we give a non-exhaustive overview of how other works in the literature could address in different ways the specific problems under consideration (2TSP, WEP, CMP) and how other general techniques could be employed for their solution.

Smoothing techniques are useful for solving many problems. Smoothing methods for specific non-smooth functions have been employed in many circumstances [FJQ99; Che12]. The smoothing methods shown in [Nes04; BT12] are efficient among the ones that are not very specific for a non-smooth function (as in [FJQ99; Che12]) or given by a formula, in which case the situation is more concrete since generalized derivatives could be calculated by hand. Note that the structure of the non-smooth function considered in [Nes04; BT12] is less general than the one we consider, since the smoothed function in those references is always convex.

The 2TSP is an old problem considered repeatedly when the recourse functions are convex. Efficient solutions are given by cutting-plane methods [Kel60] or bundle methods [OSS11], for instance. We consider non-convex recourse functions. The works [XY13; XWY14; XYZ14; XYZ15] propose smoothing algorithms for very general classes of non-smooth and non-convex problems. However, their smoothing technique is based on computing multi-dimensional integrals. On the other hand, in [Liu+20] the authors consider a version of the 2TSP with less general recourse functions given by value functions of linear problems with parameters in the objective and in the right-hand side. Comparatively, we are able to consider recourse functions given by value functions of fully parameterized convex problems.

The WEP can be addressed by [FM99] without decomposition across agents or scenarios. Recently, other Newtonian methods with some decomposition properties are proposed in [LSS16; LSS12; CF10; GF10]. A derivative-free method with decomposition is considered in [DJW17]. Moreover, augmented Lagrangian methods can also be employed without decomposition [And+08; KS16; Sch12]. Comparatively, our proposal can solve the WEP with decomposition across agents and scenarios.

Regarding the CMP, there are two issues. First, the sequential sampling, which consists in solving the same multi-stage stochastic optimization problem for many sets of scenarios. Second, when there are selfish players managing each reservoir, one needs to formulate and solve deterministic and stochastic multi-stage equilibrium problems so that the cascade has its total wealth maximized. In our case, the solution of the deterministic equilibrium problem amounts to solving a trilevel optimization problem. In the stochastic case, it can be handled by some new forms of hierarchical multi-stage stochastic optimization.

Sequential sampling algorithms are presented in [BM06; BM06; KSM02; SM98; MMF11]. From a classical perspective, the sequential sampling is a statistical problem where obtaining samples is very expensive because stochastic optimization problems need to be solved repeatedly. These works employ statistical techniques to try to solve less stochastic optimization problems from scratch. We instead propose an efficient way to solve approximately as many multi-stage stochastic problems as desired.

Trilevel optimization problems are considered in [WC17; Yao+07; DKK20]. Essentially, all these trilevel models can be solved by a sequence of bilevel optimization problems. We apply a similar technique to our deterministic trilevel model.

Hierarchical multi-stage stochastic optimization problems have not been considered previously, with one exception. In the recent working paper [Sch+21], the authors consider a bilevel multi-stage stochastic optimization problem for a cascade of water reservoirs as well. The developments consider manual simplifications or analysis of the equations of the problem so that it can be solved.

## 1.3 Contributions

We comment on the contributions of this thesis from different perspectives. The work contains new models, new algorithms and new mathematical results. These contributions are related to two new techniques, namely, the smoothing method and the free floating cuts.

Both techniques can be employed to solve not only the problems already mentioned (2TSP, WEP, CMP). In this sense, they are not specific to the applications considered, but rather somewhat general ways for obtaining useful information about value functions and solution mappings. The results from the applications of both techniques to the specific problems suggest that the techniques are useful for computing the solutions in efficient ways. For instance, the smoothing methods are competitive against some start-of-the-art solvers for non-smooth problems. The free floating cuts allow for the solution of some really large-scale problems that currently cannot be solved efficiently by other means.

The smoothing method can be used to solve general non-smooth and non-convex optimization problems as long as the non-smooth functions can be represented either as solution mappings or value functions of fully parameterized convex problems. From a theoretical perspective, the epigraphical convergence of smoothed problems to the original problems is a satisfactory theoretical result that we obtain. The analysis of limits of stationary points of the smoothed problems turned out to be a hard problem, which we could solve only when the non-smooth value function is convex and the smoothing preserves the convexity. Moreover, the inequalities that we obtained while analyzing the smoothing method are useful for proving the local boundedness of the smoothed gradients. Nonetheless, we believe that more developments are needed for a general analysis of limits of stationary points of the smoothed problems (specially when solution mappings are the ones being smoothed). In summary, the contributions of the first part of the thesis are:

- The smoothing method itself, as a tool for solving a large class of non-smooth and non-convex optimization problems involving value functions and solution mappings, possibly taking advantage of the special structure of the problems.
- The introduction of a Tikhonov term in the smoothing, which makes it possible to handle unbounded solution mappings. This feature implies good theoretical properties and induces numerical stability.
- The application of the smoothing method for the Two-Stage Stochastic Optimization Problem and to the Walrasian Equilibrium Problem and the comparison of the results against state-of-the-art solvers.
- The theoretical analysis of some aspects of the smoothing, namely, (i) the epigraphical convergence of the smoothed problems to the original non-smooth problem, (ii) the local boundedness of gradients of smoothed value functions under weak assumptions, (iii) inequalities bounding the distance of the smoothed value functions and smoothed solution mappings to the non-smooth value function and solution mapping.

The contributions related to the smoothing methods [BSS20; BSS21] are published in:

- Borges, P., Sagastizábal, C. & Solodov, M. A regularized smoothing method for fully parameterized convex problems with applications to convex and nonconvex two-stage stochastic programming. Mathematical Programming, 2020. DOI: `https://doi.org/10.1007/s10107-020-01582-2`.
- Borges, P., Sagastizábal, C. & Solodov, M. Decomposition Algorithms for Some Deterministic and Two-Stage Stochastic Single-Leader Multi-Follower Games. Computational Optimization and Applications, 78, 675-704, 2021. DOI: `https://doi.org/10.1007/s10589-020-00257-0`.

Regarding the free floating cuts, they are an extension of the Benders cut that, when applied correctly to classical multi-stage stochastic optimization problems, gives an efficient way for approximating their value functions with respect to right-hand side parameters. In our context, parameters refer to exogenous values used in the formulation. For the CMP, it can be the inflow scenarios of rainwater, or the water released by the other water reservoirs. Moreover, correctly applying the floating cuts to multi-stage problems, means that the new cut needs to be used in non-trivial forms, which sometimes are quite intricate. In this sense, there are two contributions related to the floating cuts. First, the cut itself is new. Second, it is employed in useful ways to perform the approximations desired. In summary, the contributions on the second part of the thesis are:

- The free floating cut itself and its successful application for the sensitivities of value functions of standard multi-stage stochastic problems with respect to right-hand side parameters.
- A deterministic trilevel model to improve the total wealth obtained from a cascade with three reservoirs, an algorithm to solve such model to global optimality, its convergence analysis and an illustration of the numerical results.
- The formulation of the individualistic stochastic optimization problems modeling the flow of water on the cascade when there is no collaboration between agents. The formulation of the a profit-sharing mechanism that transfers profits between water reservoirs so that collaborations between agents can be monetized. The application of the sensitivities to compute individualistic stochastic policies for the CMP and to make possible the use of a best-response algorithm to obtain greedy policies for the agents of the cascade. The numerical illustration of the policies obtained and brief comments on the convergence properties of the stochastic algorithms.
- The application of the sensitivities of value functions of multi-stage problems to perform efficient sequential sampling of multi-stage problems for thousands of sets of scenarios.

The contributions of the second part of the thesis are currently being considered for publication:

- Borges, P., Sagastizábal, C., Solodov, M., D'Ambrósio, C. & Liberti, L. Profit Sharing Mechanisms in Multi-Owned Cascaded Hydro Systems. Submitted to Mathematical Programming, 2021.
- Borges, P. Cut Sharing Across Trees and Efficient Sequential Sampling for SDDP with Uncertainty in the Right-Hand Side. Second round revision in Computational Optimization and Applications, 2021.

## 1.4 Organization of the thesis

The thesis is organized in six chapters. The first chapter contains background material that is used frequently in the text. The next four chapters are divided in two parts. Each part employs similar ideas to solve different problems. The first part employs smoothing methods to solve some non-smooth and non-convex optimization problems involving value functions and solution mappings. The second part uses an extension of the Benders cuts to obtain efficient representations of value functions of multi-stage stochastic optimization problems. The final chapter contains some concluding remarks with suggestions for future developments.

In Chapter 3 we consider the Two-Stage Stochastic Optimization Problem with non-convex recourse functions that are optimal values of fully parameterized convex optimization problems. The problem is solved via a technique for smoothing the underlying recourse functions. We analyze some theoretical aspects of the smoothing technique, such as the distance of the smoothed value function to the original function, the local boundedness of the smoothed gradients and if limits of stationary points of the smoothed problems are stationary points for the original problem. We also compare the performance of the smoothing method against a state-of-the-art implementation of a bundle method.

In Chapter 4 we apply the same smoothing method of Chapter 3 to smooth solution mappings of fully parameterized convex problems. As a consequence we are able to solve some large-scale single-leader multi-follower games such as the Walrasian Equilibrium Problem. We analyze the epigraphical convergence of the smoothed problems to the original problem and extend some results from Chapter 3 to situations where the domains of the solution mappings are not the entire space, which turns out to be relevant for the situation. We compare our algorithm against a state-of-the-art implementation of a Newtonian method for complementarity problems.

In Chapter 5 we first solve a deterministic trilevel optimization problem that models the monetization of the collaborations between selfish profit-maximizing agents in a cascade of three water reservoirs used to produce electricity. We introduce the free floating cuts, which are an extension of the Benders cuts. The free floating cuts are used to solve stochastic variants of this trilevel problem by means of a best-response algorithm. The floating cuts are also used to find individualistic multi-stage stochastic policies when agents do not collaborate. We show the results obtained by the algorithms in a cascade with three reservoirs.

In Chapter 6 we use the free floating cuts to perform efficiently the sensitivity analysis of value functions of standard multi-stage stochastic optimization problems with respect to right-hand side uncertainty, since they give a lower bound on the optimal value for all possible right-hand side scenarios. We present numerical experiments illustrating the lower bounding property and the sensitivities obtained when the scenarios are drawn from a distribution that can be represented by finitely many scenarios. We also show the sensitivity when scenarios are drawn from a normal distribution, which cannot be represented by finitely many scenarios.

# Chapter 2

# Background Material

In this chapter we recall standard results frequently used in the thesis. Concepts from convex analysis and general variational analysis are recalled in Section 2.1. For analysing the approximating properties of the smoothed problems to the original non-smoooth problem we employ epigraphical convergence theory (Section 2.2) and gradient consistency theory (Section 2.3). The cutting-plane method and its convergence properties are recalled in Section 2.4. The Stochastic Dual Dynamic Programming method is recalled in Section 2.5.

The results in Section 2.1 and Section 2.2 can be found in [RW09, Chapters 2, 7 and 9]. The concept of gradient consistency and related results can be found in [Che12; BH16; BH13].

## 2.1 Elements of variational analysis

Variational analysis is an area of modern optimization theory that studies functions that do not have classical derivatives and set-valued mappings, which are mappings whose values are sets. The most frequent of such functions, with practical applications, are the value functions of optimization problems, their solution mappings and feasible sets. Convex analysis is a subfield of variational analysis that deals with convex functions, which is the most developed part of the theory.

Given a set-valued mapping $R : \mathbb{R}^{nx} \to \mathbb{R}^{ny}$, recall that the outer limit of $R$ at $x \in \mathbb{R}^{nx}$ is defined as

$$\limsup_{x\to\bar{x}} R(x) = \{y \in \mathbb{R}^{ny} : \exists x_k \to \bar{x}, \quad y_k \in R(x_k) \quad \text{s.t.} \quad y_k \to y\},$$

and its inner limit by

$$\liminf_{x\to\bar{x}} R(x) := \{y \in \mathbb{R}^{ny} : \quad \forall x_k \to \bar{x} \quad \exists y_k \in R(x_k) \quad \text{s.t.} \quad y_k \to y\}.$$

With a slight abuse of notation, for the value function we shall write $\limsup_{x\to\bar{x}} v(x)$ for $\limsup_{x\to\bar{x}}\{v(x)\}$, where $v$ is a function.

The map $R$ is outer semi-continuous at $\bar{x} \in \mathbb{R}^{nx}$ if $\limsup_{x\to\bar{x}} R(x) \subset R(\bar{x})$. The set-valued map $R$ is said to be inner semi-continuous at $\bar{x}$ if $\liminf_{x\to\bar{x}} R(x) \supset R(\bar{x})$.

A set-valued map $R$ is locally bounded at $\bar{x} \in \mathbb{R}^{nx}$ if there is an open set $V \subset \mathbb{R}^{nx}$ such that $\bar{x} \in V$ and $S(V) := \cup_{x\in V} S(x)$ is bounded. It is pointwise bounded if $R(x)$ is a bounded set for all $x \in \mathbb{R}^{nx}$.

The domain of the set-valued map $R$ is $\operatorname{dom} R := \{x \in \mathbb{R}^{nx} : R(x) \neq \emptyset\}$. If the set-valued map $R$ is a singleton over $\operatorname{dom} R$, we represent it by a function over its domain, using a lower case letter. For example, if $R(x) = \{r(x)\}$ for all $x$, we more often use $r(x)$.

Given a finite-valued convex function $v : \mathbb{R}^{nx} \to \mathbb{R}$, its subdifferential at $\bar{x}$ is given by

$$\partial v(\bar{x}) = \{u \in \mathbb{R}^{nx} : v(x) \geq v(\bar{x}) + u^\top (x - \bar{x}) \quad \forall x\}.$$

The regular subdifferential of $v : \mathbb{R}^{nx} \to \mathbb{R}$, assumed to be continuous, at $\bar{x} \in \mathbb{R}^{nx}$ is given by

$$\hat{\partial} v(\bar{x}) := \left\{ u \in \mathbb{R}^{nx} : \liminf_{x\to\bar{x}} \frac{v(x) - v(\bar{x}) - u^\top (x - \bar{x})}{\|x - \bar{x}\|} \geq 0 \right\},$$

the limiting subdifferential, which by convention has the same symbol as in the convex case, by

$$\partial v(\bar{x}) := \limsup_{x \to \bar{x}} \hat{\partial} v(x),$$

and the horizon (or singular Mordukhovich) subdifferential by

$$\partial^\infty v(\bar{x}) := \Big\{ u \in \mathbb{R}^{nx} : \exists x_k \to \bar{x}, \quad u_k \in \hat{\partial} v(x_k), \quad t_k \searrow 0 \quad \text{s.t.} \quad t_k u_k \to u \Big\}.$$

Denote by cl $D$ the closure of a set $D$, and by conv $D$ its convex hull. Then the Clarke subdifferential is given by

$$\partial_C v(x) = \text{conv cl } \{\partial v(x) + \partial^\infty v(x)\},$$

see [Mor18, Theorem 3.57]. If $v$ is locally Lipschitz, then $\partial_C v(x) = \text{conv } \partial v(x)$. To avoid confusion, we mention some alternative terminology widely used in the variational analysis literature: Clarke subdifferential is sometimes called convexified subdifferential (or generalized gradient, in the case of Lipschitz function), regular subdifferential is also known as Fréchet subdifferential, and limiting subdifferential as Mordukhovich subdifferential.

The regular subdifferential and the Clarke subdifferential are convex sets. The Clarke and the limiting subdifferentials are outer semi-continuous multi-functions. When $v$ is convex, it is locally Lipschitz and all these subdifferential notions coincide with the classical subdifferential of convex analysis.

The following proposition characterizes local boundedness of the value function subdifferential. It is a specialization of [RW09, Theorems 9.13 and 9.2] to our setting (note that in our case the value function is finite-valued).

**Proposition 2.1.1** (Subdifferential Characterization of Local Lipschitz Continuity). *Given an open set $V \subset \mathbb{R}^{nx}$, let $v : V \to \mathbb{R}$ be continuous. The following conditions are equivalent:*

1. *The function $v$ is locally Lipschitz at $\bar{x}$.*
2. *The regular subdifferential $\hat{\partial} v$ is locally bounded at $\bar{x}$.*
3. *The limiting subdifferential $\partial v$ is locally bounded at $\bar{x}$.*
4. *The horizon subdifferential $\partial^\infty v(\bar{x})$ contains only the zero vector.*

*Proof.* Theorem 9.13 from [RW09] depends on strict continuity of the value function, given in Definition 9.1, which combined with Theorem 9.2 of [RW09], yields the stated equivalences. Note that strict continuity means local Lipschitz continuity. □

The effective domain, or the domain, of a function $v : \mathbb{R}^{nx} \to \overline{\mathbb{R}}$ is dom $v := \{x \in \mathbb{R}^{nx} : v(x) < +\infty\}$. A function $v$ is said to be continuous relative to $V \subset \mathbb{R}^{nx}$ if for all $x \in V$ we have $v(x') \to v(x)$ whenever $x' \to x$ such that $x' \in V$. For instance, we could say that $v$ is continuous relative to its domain, or relative to the interior of its domain, or relative to the interior of the domain of some set-valued map.

## 2.2 Epigraphical convergence

For an extended-valued function $v : \mathbb{R}^q \to \overline{\mathbb{R}}$ and a scalar parameter $\varepsilon \geq 0$, we define the set-valued mapping of $\varepsilon$-approximate unconstrained minimizers of the function $v$ as

$$\varepsilon - \operatorname{Arg}\min_p v := \{p \in \mathbb{R}^q : v(p) \leq \varepsilon + \inf_p v\}.$$

A sequence of functions $v_k : \mathbb{R}^q \to \overline{\mathbb{R}}$ is said to converge epigraphically, see [RW09, Proposition 7.2], to $v : \mathbb{R}^q \to \overline{\mathbb{R}}$ if the following two conditions hold for all $p \in \mathbb{R}^q$:

$$\liminf_k v_k(p_k) \geq v(p) \quad \text{for all} \quad p_k \to p,$$

and

$$\limsup_k v_k(p_k) \leq v(p) \quad \text{for some} \quad p_k \to p.$$

The notion of epigraphical convergence of functions is tightly related to the convergence of minimizers thanks to the following theorem, adapted from [RW09, Theorem 7.31].

**Theorem 2.2.1.** *For a sequence of extended-valued functions $v_k$ converging epigraphically to $v$, the following holds.*

*(i)* $\inf v_k \to \inf v$ *if and only if for all* $\varepsilon > 0$ *there is a compact set* $B \subset \mathbb{R}^q$ *such that* $\inf_B v_k \leq \inf v + \varepsilon$ *for all k large enough,*

*(ii)* $\limsup_k \{\varepsilon_k - Arg\min v_k\} \subset Arg\min v$ *for any* $\varepsilon_k \to 0$.

## 2.3 Gradient consistency

Let $v : V \to \mathbb{R}$ be continuous on the open set $V \subset \mathbb{R}^q$. Assume that for $\varepsilon > 0$ we are given smooth (twice continuously differentiable, for instance) functions $v^\varepsilon : V \to \mathbb{R}$. Their lower semicontinuous closure is defined as

$$\text{lsc } v^\varepsilon(p) := \liminf_{\varepsilon \searrow 0, p' \to p} v^\varepsilon(p').$$

The sequence $v^\varepsilon$ is a smoothing for $v$ if

$$\text{lsc } v^\varepsilon(p) = v(p) \text{ for all } p \in V.$$

If $v^\varepsilon$ is a smoothing for $v$, it is well known, see [BHK13], that

$$\partial v(p) \subset \text{conv}\left\{ \limsup_{\varepsilon \searrow 0, p' \to p} \nabla v^\varepsilon(p') \right\} \text{ for all } p \in V. \tag{2.1}$$

Moreover, the smoothing functions $v^\varepsilon$ are said to be gradient consistent with a Lipschitz function $v$, when $\varepsilon \searrow 0$, if

$$\partial_C v(p) \supset \text{conv}\left\{ \limsup_{\varepsilon \searrow 0, p' \to p} \nabla v^\varepsilon(p') \right\} \text{ for all } p \in V. \tag{2.2}$$

Note that in the last statement we required $v$ to be Lipschitz, otherwise the definition would have to include information about the singular subgradients as well. For more details about smoothing functions, see [BHK13], [BH16].

## 2.4 Cutting-plane methods

Let $f : \mathbb{R}^{nx} \longrightarrow \mathbb{R}$ be a finite-valued convex function, for simplicity. The cutting-planes method [Kel60] is a well-known algorithm for solving problems of the form $\min_{x \in D} f(x)$, where $D$ is the feasible set. It assumes that for all $x \in \mathbb{R}^{nx}$ we can obtain a subgradient $\alpha \in \partial f(x)$. The cutting-planes method is given below.

**Algorithm 2.4.1** (Cutting-Planes Method (CPM))**.**

```
**Input:** $\varepsilon \geq 0$ and $x^1 \in \mathbb{R}^{nx}$.
**Initialization.** Set $k = 1$.
**Step 1: Obtain the Functional Value and a Subgradient.** Compute $f(x^k)$ and $\alpha^k \in \partial f(x^k)$.
**Step 2: Define the Model.** Define $f^k(x) = \max_{i=1,\dots,k}\{f(x^i) + (\alpha^i)^\intercal(x - x^i)\}$.
**Step 3: Next Iterate.** Compute $x^{k+1} \in \arg\min_{x \in D} \quad f^k(x)$
**Step 4: Termination.** If $f(x^{k+1}) - f^k(x^{k+1}) \leq \varepsilon$, then stop.
**Step 5: Loop.** Set $k = k+1$ and go back to Step 1.
```

The convergence analysis of the CPM is usually divided into two cases: $\varepsilon > 0$ and $\varepsilon = 0$. When $\varepsilon > 0$, the algorithm always stops with an approximate solution. In the other case, when $\varepsilon = 0$, it finds an exact solution if there is a finite set $\Lambda$ such that $(x^k, \alpha^k) \in \Lambda$ for all $k$. Such condition can be satisfied in practice for important cases explained after the next theorem, which justifies these convergence comments.

**Theorem 2.4.2** (Convergence Analysis of the CPM)**.** *Let* $f : \mathbb{R}^{nx} \longrightarrow \mathbb{R}$ *be a finite-valued convex function and let D be a non-empty compact set. It follows that:*

- *If $\varepsilon > 0$, there is k such that the CPM stops and $x^{k+1} \in \varepsilon - argmin_{x\in D} f$.*
- *If $\varepsilon = 0$ and there is a finite set $\Lambda$ such that $(x^k, \alpha^k) \in \Lambda$ for all k, then CPM finds an exact solution after finitely many iterations.*

*Proof.* For the first item, assume that the CPM never stops. Equivalently, assume that $f(x^{k+1}) - f^k(x^{k+1}) > \varepsilon$ for all $k$. Since $D$ is compact, there is a convergent subsequence $\{x^{k_i}\}_i$ of $\{x^k\}_k$ such that $x^{k_i} \to \bar{x}$. On the other hand, since $f$ is convex and finite valued, it is also a Lipschitz function, with constant $L$. Then, $\|\alpha^k\| \leq L$ for all $k$. Therefore,

$$\varepsilon < f(x^{k_{i+1}}) - f^k(x^{k_{i+1}}) \leq f(x^{k_{i+1}}) - f(x^{k_i}) - (x^{k_{i+1}} - x^{k_i})^\intercal \alpha^{k_i} \leq |f(x^{k_{i+1}}) - f(x^{k_i})| + \|\alpha_i^k\| \|x^{k_{i+1}} - x^{k_i}\|.$$

This is a contradiction, because $\|x^{k_{i+1}} - x^{k_i}\| \longrightarrow 0$ and $|f(x^{k_{i+1}}) - f(x^{k_i})| \longrightarrow 0$. Then, the CPM stops after finitely many iterations if $\varepsilon > 0$.

For the second item, let us note that because $\Lambda$ is a finite set, the iterate $(x^k, \alpha^k)$ repeats. Define $v = \min_{x\in D} f(x)$. Therefore, we know that

$$v \leq f(x^k) = f(x^k) + (x^k - x^k)^\intercal \alpha^k \leq \max_{i=1,\ldots,k} \{f(x^i) + (\alpha^i)^\intercal (x^k - x^i)\} = f^k(x^k).$$

Moreover, since $f^k \leq f$ for all $x$, it follows by minimizing over $D$ on both sides, that

$$f(x^k) \leq f^k(x^k) \leq v.$$

Therefore, $f(x^k) = v$. $\square$

Let us now discuss the finite set $\Lambda$ such that $(x^k, \alpha^k) \in \Lambda$ for all $k$. Take $f(x) = \min_y\{c^\intercal y : Ay = b - Tx, y \geq 0\}$ and assume that $f$ is finite-valued for all $x$. It is easy to check that the dual feasible set of the problem defining $f$ does not depend on $x$. Therefore, we can enumerate the vertices of the dual polyhedron. Moreover, the subgradients computed for $f$ are a function of these dual vertices, which imply that $\alpha^k$ can be forced to be finite. If, in addition, $D$ is a polyhedron, then the possible problems $\min_{x\in D} f^k$ are finitely many when reformulated as linear problems. Therefore, by forcing $x^k$ and $\alpha^k$ to be vertices, we can enforce the referred assumption.

## 2.5 Stochastic dual dynamic programming

The classical Stochastic Dynamic Programming (SDP) algorithm [Bel57] can be used to solve multistage stochastic optimization problems with a Markovian structure, but only when the amount of possible decisions and resulting number of states is small enough. The Stochastic Dual Dynamic Programming (SDDP) algorithm made possible the extension of the SDP for much bigger problems via a suitable representation of the cost-to-go functions via Lagrangian duality.

Let us recall the multistate stochastic optimization problem considered. We are given the parameterized feasible sets

$$D_{ts}(x_{t-1}) := \{x_{ts} \geq 0 \quad : \quad W_t x_{ts} = h_{ts} - B_t x_{t-1}\}.$$

The multistate problem under consideration, with statewise independent realizations, is given by

$$\min_{x_1} \quad c_1^\intercal x_1 + Q_2(x_1) \quad \text{s.t.} \quad x_1 \in D_1$$

where for $t = 2, \ldots, T$ and $s = 1, \ldots, S$ we have

$$Q_t(x_{t-1}) := S^{-1} \sum_{r=1}^{S} Q_{tr}(x_{t-1})$$

and

$$Q_{ts}(x_{t-1}) := \min_{x_{ts}} \quad c_t^\intercal x_{ts} + Q_{t+1}(x_{ts}) \quad \text{s.t.} \quad x_{ts} \in D_{ts}(x_{t-1})$$

with

$$Q_{T+1}(\cdot, \cdot) \equiv 0.$$

Whenever we consider an approximation $Q_t^k(x_{t-1})$ for aggregated cost-to-go function, it is given by the average of the approximations of the cost-to-go function for each scenario, as expressed by

$$Q_t^k(x_{t-1}) := S^{-1}\sum_{r=1}^{S} Q_{tr}^k(x_{t-1}). \tag{2.3}$$

The SDDP method is shown in Algorithm 2.5.1. Its convergence properties are discussed in [Sha11; CP99; DB06; LP05]. We do not discuss the proofs for simplicity. However, it is very much related to the cutting-planes method already explained.

**Algorithm 2.5.1** (SDDP Algorithm)**.**

**Initialization.** Take a tolerance $\varepsilon \geq 0$. Set $k = 1$. Take $M > 0$ large. Set $Q_{ts}^k(\cdot) \equiv -M$ for all $t = T, \dots, 2$ and all $s = 1, \dots, S$. For all $t = T, \dots, 2$ the functions $Q_t^k(\cdot)$ are given by formula (2.3).

**Step 1: Sample Scenario.** Sample a scenario $\omega_t^k \in \{1, \dots, S\} \quad \forall t = 2, \dots, T$.

**Step 2: Forward.** Compute

$$\hat{x}_1^k \in \arg\min_{x_1} \quad c_1^\intercal x_1 + Q_2^k(x_1) \quad \text{s.t.} \quad x_1 \in D_1$$

and for $t = 2, \dots, T$ and $s = \omega_t^k$ compute

$$\hat{x}_t^k \in \arg\min_{x_{ts}} \quad c_t^\intercal x_{ts} + Q_{t+1}^k(x_{ts}) \quad \text{s.t.} \quad x_{ts} \in D_{ts}(\hat{x}_{t-1}^k).$$

**Step 3: Backward.** For all $t = T, \dots, 2$ and all $s = 1, \dots, S$ compute

$$x_{ts}^k \in \arg\min_{x_{ts}} \quad c_t^\intercal x_{ts} + Q_{t+1}^{k+1}(x_{ts}) \quad \text{s.t.} \quad x_{ts} \in D_{ts}(\hat{x}_{t-1}^k)$$

where $Q_{T+1}^{k+1}(\cdot) \equiv 0$ and for $t = T-1, \dots, 1$ take $Q_{t+1}^{k+1}(\cdot)$ given by formula (2.3). For each $t = T-1, \dots, 1$ and $s = 1, \dots, S$ calculate subgradients

$$\alpha_{tsk} \in \partial Q_{t+1,s}(\hat{x}_t^k)$$

and take $Q_{t+1,s}^{k+1}(\cdot)$ as a maximum between $Q_{t+1,s}^k(\cdot)$ and the affine function

$$Q_{t+1,s}(\hat{x}_t^k) + \alpha_{tsk}^\intercal (x_t - \hat{x}_t^k).$$

**Step 4: Lower Bound.** Compute

$$x_1^k \in \arg\min_{x_1} \quad c_1^\intercal x_1 + Q_2^{k+1}(x_1) \quad \text{s.t.} \quad x_1 \in D_1. \tag{2.4}$$

Set $\underline{v}_k$ as the optimal value of (2.4) and $\bar{v}_k = \sum_{t=1}^{T} c_t^\intercal \hat{x}_t^k$.

**Step 5: Stopping Test.** Go to Step 2 if the lower bound $\underline{v}_k$ stabilized across $k$ or if the average forward cost $\bar{v}_i$ for $i = 1, \dots, k$ is close enough to the lower bound $\bar{v}_k$ as expressed by

$$\frac{1}{k}\sum_{i=1}^{k} \bar{v}^i \leq \varepsilon + \underline{v}_k.$$

**Step 6: Loop.** Set $k = k+1$ and go back to Step 1.1.

# Part I

# Smoothing Methods

# Chapter 3

# A Regularized Smoothing Method for Solutions of Fully Parameterized Convex Problems with Applications to Convex and Nonconvex Two-Stage Stochastic Programming

We present an approach to regularize and approximate solution mappings of parametric convex optimization problems that combines interior penalty (log-barrier) solutions with Tikhonov regularization. Because the regularized mappings are single-valued and smooth under reasonable conditions, they can be used to build a computationally practical smoothing for the associated optimal value function. The value function in question, while resulting from parameterized convex problems, need not be convex. One motivating application of interest is two-stage (possibly nonconvex) stochastic programming. We show that our approach, being computationally implementable, provides locally bounded upper bounds for the subdifferential of the value function of qualified convex problems. As a by-product of our development, we also recover that in the given setting the value function is locally Lipschitz continuous. Numerical experiments are presented for two-stage convex stochastic programming problems, comparing the approach with the bundle method for nonsmooth optimization.

## 3.1 Introduction and motivation

This part focuses on developing computationally implementable smoothing methods for a family of parametric convex programming problems, noting that all the functions are differentiable but the parametric dependence can be arbitrary.

As one motivating application, the approach provides approximations to (possibly nonconvex) stochastic programs, as long as they exhibit certain structure suitable to our theory. The setting can be illustrated by the following abstract stochastic programming problem formulation:

$$\min_{x\in X} f_0(x) := \mathscr{R}[F(x,\xi(\omega))], \tag{3.1}$$

where $\mathscr{R}$ is a risk measure [DRS09, Chap. 6], and $F(x,\xi)$ is a real-valued function of the decision variables $x \in X \subset \mathbb{R}^{nx}$. The random vector $\xi(\omega)$ has known probability distribution, with finite support described by scenarios $\xi_s$ and probabilities $p_s \in (0,1)$ for $s = 1,\dots S$.

We start with an example when problem (3.1) is convex. Consider a two-stage stochastic linear program

$$\left\{\begin{array}{ll} \min & c^\top x + \sum\limits_{s=1}^{S} p_s \mathscr{Q}_s(x) \\ \text{s.t.} & x \in X \end{array}\right. \quad \text{for } \mathscr{Q}_s(x) := \left\{\begin{array}{ll} \min & q_s^\top y \\ \text{s.t.} & Wy = h_s - T_s x \\ & y \geq 0\,, \end{array}\right. \tag{3.2}$$

where the involved vectors and matrices have suitable dimensions. Suppose the property of relative complete recourse [DRS09, Sect. 2.1.3] is satisfied. Then the format (3.1) is obtained by taking $\xi = (q_s, h_s, T_s)$, $F(x,\xi_s) = c^\top x + \mathscr{Q}_s(x)$ and $\mathscr{R} = \mathbb{E}$, the expected value function. Since in (3.2) the first-stage variable appears only in the right-hand side of the feasible set defining the second-stage problems, the corresponding recourse function $\mathscr{Q}_s$ is nonsmooth convex [DRS09, Prop. 2.2]. Hence, so is the associated objective in (3.1), which is given by

$$f_0(x) = c^\top x + \sum_{s=1}^{S} p_s \mathscr{Q}_s(x)\,. \tag{3.3}$$

It could be argued that one may get around the nonsmoothness of (3.2) simply by writing down the deterministic equivalent, a linear programming problem on variables $(x, y_1, \ldots, y_S)$. However, such a rewriting would preclude the possibility of *scenario decomposition* that is present in the nonsmooth formulation. The option to solve separate, easy, second-stage problems (one per scenario $s$) is very important, and often exploited in real-life applications; [Sag12]. Algorithms based on L-Shaped or bundle methods, [SW69] and [Bon+06, Part II], in particular, generate cuts for the nonsmooth recourse function using the second-stage solutions. The maximum of such cuts is a piecewise affine convex function which by convexity of $\mathscr{Q}_s$ approximates $f_0$ from below and is used as a proxy in the master program to generate a new first-stage iterate. For such schemes to converge, convexity is fundamental to ensure the generated cuts approximate well the recourse function in regions near the optimum, [OS14; OSL14].

In this part, we shall follow a different path, that is suitable for both convex and certain nonconvex objective functions in (3.1). The latter setting can occur even when the recourse function $\mathscr{Q}_s$ is convex, if the risk-measure is not convex. An example is [Ahm06, Lem.1], where it is shown that for a stochastic linear program with simple recourse the classical mean-variance criterion yields a piecewise-convex function $f_0$, which itself is not convex. Risk measures involving the variance are not the only possible source of nonconvexity in (3.1): the problems considered in [HBT18] have a probability distribution that depends affinely on the first-stage variable. In this case, the function $f_0$ in (3.3) is nonsmooth and also nonconvex. Finally, if the second-stage objective function in (3.2) depends on the first-stage variable, say instead of $q_s^\top y$ we have $q_s(x, y)$, then the recourse function itself can fail to be convex. More instances and examples of similar nature, referred to as programs with linearly bi-parameterized recourse, can be found in [Liu+20].

In order to handle nonsmooth nonconvex objectives, instead of building convex cutting-plane proxies for the recourse function, as in the L-Shaped and bundle methods, we define models that are *smooth and nonconvex*. This is done by adopting a parametric programming point of view, which for (3.2) amounts to considering the recourse $\mathscr{Q}_s(x)$ as a particular instance of the *value function* of a family of problems that are parameterized by the first-stage variable, $x$. The proposal replaces each (convex) second-stage problem by a well-behaved strictly or strongly convex (approximating) nonlinear programming problem (NLP), depending on a smoothing parameter $\varepsilon > 0$, and possibly also on a Tikhonov regularization parameter. This NLP unique solution $y_s^\varepsilon(x)$ is a differentiable mapping of $x$ that defines the following smoothed value function:

$$q_s^\top y_s^\varepsilon(x) \geq \mathscr{Q}_s(x)\,, \tag{3.4}$$

which approximates monotonically the recourse function *from above*. Rather than generating cuts, the master problem minimizes the smoothed objective function

$$c^\top x + \sum_{s=1}^{S} p_s q_s^\top y_s^\varepsilon(x)\,,$$

to define a new first-stage point. An important difference with the L-Shaped family is that now the master program is an NLP. Replacing a piecewise linear master program by a nonlinear version may appear as a handicap at first sight. However, with our scheme not only the solution mappings $y_s^\varepsilon(x)$ are smooth, but they also have *computable derivatives*, related to certain smooth dual mappings, the NLP multipliers computed when solving the approximating second-stage problems. This is a clear algorithmic advantage over the usual cutting-plane models, especially in a nonconvex setting. Additionally, not only (3.4) holds uniformly for all $x$ but also, under reasonable conditions, the smoothed recourse function is bounded above by $\mathscr{Q}_s(x)$ plus a term that tends to zero when so does the smoothing parameter; see the relation (3.11) below.

The smooth approximating solution mappings are defined by suitably combining a Tikhonov regularization with a logarithmic barrier. Regarding related (or somewhat related) works, clearly there are plenty smoothing techniques in the literature. For example, those of [Nes04] and [BT12], which solve convex nonsmooth optimization problems with complexity guarantees. However, complexity analysis is not our subject in this work. The other vastly studied topic concerns generalized and directional derivatives of the optimal value functions; see, e.g., [BS00; RW09] and references therein. Intensive sensitivity analysis of optimization and variational problems via generalized differentiation, including the Lipschitz stability of optimal value/marginal functions, was conducted in [Mor06; Mor18]. Differentiability properties of solution mappings of NLP problems can be traced back to [FM68; Fia83], where the linear independence constraint qualification, strict complementarity, and the second-order sufficient optimality condition are assumed. We must mention here that these works (see also [FI90]) have already considered computing some sensitivity information using approximating penalization schemes. We follow a similar path in the sequel, but with appropriate modifications, among which is adding a Tikhonov regularizing term to the classical interior penalization. Moreover, unlike [FI90], we do not assume satisfaction of strict complementarity or the second-order sufficient condition for the original problem. Accordingly, in our setting the primal solution set need not be a singleton; and can even be unbounded. Instead, we *induce* the second-order sufficient condition on certain approximating subproblems, via the specific regularization/penalization scheme employed to compute the approximations. As we shall show, our approach has many interesting theoretical properties, and is also computationally useful; for example, to preserve decomposability of stochastic programs.

As a matter of theory, our regularized penalization scheme provides estimates for the optimal value, as well as locally bounded upper bounds for the subdifferential of the value function. This, in turn, leads to the value function being locally Lipschitz. The latter result recovers, via our computationally-oriented approach and in our case, the locally Lipschitz property established in [Guo+14] (noting that the setting of [Guo+14] is much more general). Some other results on the locally Lipschitz behavior of optimal value functions are [MNY07] and [DM14]; but these assume inner semi-continuity of solution mappings (not assumed in this work).

Generally, we regard smoothing as the ability to generate, in computable ways, single-valued and smooth primal-dual solution mappings, which are "asymptotically correct" in some sense. Therefore, the topic of interest is how the constructed single-valued approximations relate in the limit to the possibly set-valued primal and dual solution mappings, and to the optimal value function.

For various other issues of parametric and sensitivity analysis of optimization and variational problems, see the monographs [FM68; Fia83; Ban+83; BS00; RW09; Mor06; Mor18], as well as [DGL12]. In [DGL12] the authors analyze certain optimization problems in Banach spaces involving an arbitrary amount of functions that are lower semicontinuous and an abstract constraint given by a closed set. They are mostly concerned with the respective lower and upper continuity aspects of optimal values and solution sets as well as a certain generalized Lipschitz property for the feasible set. To perturb the main optimization problem, they consider a metric on the space of all possible data of the main problem. This amounts to putting a metric on the space of lower semicontinuous functions concatenated with the space of closed sets. They prove that the space of all problem data is complete under their metric. Next, they see feasible set mappings, optimal value functions and solution mappings as mappings on the space of problem data and look at qualitative aspects as well as quantitative relations for these objects. For example, one nontrivial instance for the problems considered in [DGL12] is the master problem of the possibly nonconvex stochastic programming problem considered here. However, we do not deal directly with the underlying nonsmooth problem as opposed to [DGL12]. Instead, we want to show how to build well-behaved smooth approximations to nonsmooth and nonconvex value functions and to understand how these smooth approximations provide some useful information for the value functions. We capitalize on smooth optimization to solve easier approximations for a harder problem, leaving the theoretical issues concentrated solely on how the approximations relate to the original model.

The rest of the chapter is organized as follows. In Section 3.2, we fix notation and blanket assumptions, define the Tikhonov-regularized interior penalty scheme and associated smoothing of the value function. For a revision of some basic concepts in set-valued analysis we refer to Section 2.1. In Section 3.3, we specialize to our setting several results for the approximate optimal value function, including its parametric differentiability. Some technical bounds are gathered in Section 3.4. The final theoretical Section 3.5 shows that the gradients of the approximate value function are locally bounded and, as a by-product of our developments, we recover the result that optimal value functions of qualified convex problems are locally Lipschitz. The developed theory is general, not only applicable to stochastic programs; nevertheless, as two-stage stochastic programs is an important motivation for us, in Section 3.6 we go back to the issue of smoothing risk-averse variants of such problems. In the numerical Section 3.7 the approach is benchmarked on convex instances against a state-of-the-art bundle method software for nonsmooth optimization [Fra02]. Some concluding remarks are made in Section 4.6.

## 3.2 The setting and main ingredients of the approach

The family of problems in consideration is parameterized by $x \in \mathbb{R}^{nx}$ and has decision variable $y \in \mathbb{R}^{ny}$. The objective function is $f : \mathbb{R}^{nx} \times \mathbb{R}^{ny} \to \mathbb{R}$. Equality constraints are given by means of parametric mappings $A : \mathbb{R}^{nx} \to M(l \times ny)$ and right-hand side maps $b : \mathbb{R}^{nx} \to \mathbb{R}^{l}$, where $M(l \times nx)$ is the space of $l \times nx$-matrices. The parametric inequality constraints are $g_i : \mathbb{R}^{nx} \times \mathbb{R}^{ny} \to \mathbb{R}$, $i = 1, \ldots, m$. Accordingly, the parametric optimization problem is:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & f(x,y) \\ \text{subject to} \quad & A(x)y = b(x), \\ & g_i(x,y) \le 0, \quad i = 1, \ldots, m. \end{aligned} \tag{3.5}$$

Of course, not all functions need to really have a parametric dependence, and not all types of constraints need to be present. Special cases, like right-hand side and canonical perturbations are included implicitly. In particular, for two-stage linear stochastic programs (3.2), problem (3.5) represents the second-stage problems defining the recourse, and only the right-hand side mapping depends on $x$; specifically, in this case

$$f(x,y) = q_s^\top y\,,\; A(x) = W\,,\; b(x) = h_s - T_s x\,,\; \text{ and } g(x,y) = -y$$

(so $m = ny$).

### 3.2.1 Blanket assumptions and Tikhonov-regularized interior penalty scheme

Throughout we assume that in (3.5) the following holds for all $x \in \mathbb{R}^{nx}$ (of course, one could instead consider some subset of parameters in $\mathbb{R}^{nx}$):

1. The functions $f(x,\cdot)$ and $g_i(x,\cdot)$, $i = 1,\dots,m$, are convex.
2. The mappings $f$, $b$, $A$ and $g_i$ are at least twice continuously differentiable in both the parameter and the decision variable.
3. The $l \times nx$ matrix $A(x)$ has linearly independent rows.
4. The constraints in (3.5) satisfy the Slater condition: for every $x$ there exists $\mathring{y}(x)$ such that $A(x)\mathring{y}(x) = b(x)$, $g_i(x,\mathring{y}(x)) < 0$, $i = 1,\dots,m$.
5. For every $x$, problem (3.5) has at least one solution.

Let $S(x)$ denote the (nonempty, possibly unbounded) primal solution set of (3.5) and let

$$v(x) := f(x,y(x)), \quad \text{for } y(x) \in S(x), \tag{3.6}$$

be the value function of problem (3.5).

Recall that [RW09, Theorem 1.17] ensures that the value function (3.6) is continuous under uniform level-boundedness. While our blanket assumptions above do not imply the latter condition, we can still conclude continuity of the value function via our uniform approximation of the value function via smooth functions, and the condition (3.14) introduced in the sequel. We do not assume that the Slater points are uniform across parameters. They can change freely with $x \in \mathbb{R}^{nx}$. All the hypotheses about (3.5) and the associated problem data stated in this section, are not stated again and will be taken in the subsequent sections as granted.

Thus, the main object of our study are smooth parametric convex programming problems, with Slater points and without redundant equality constraints, that have nonempty solution sets for all parameters. The goal is to construct computable (and well-behaved) approximations to primal and dual solution mappings, and to value functions. To that end, define the following *Tikhonov-regularized interior-penalty* (log-barrier) function:

$$\phi(x,y) = -\sum_{i=1}^{m} \ln\{-g_i(x,y)\} + \frac{\mu}{2}\|y\|^2, \tag{3.7}$$

where $\mu \geq 0$ and $\|\cdot\|$ denotes the Euclidean norm. In our constructions, we use $\mu$ fixed, mainly because this turns out to be sufficient for our purposes. In particular, the size of this Tikhonov regularization is controlled by the penalty parameter $\varepsilon$ multiplying the full function $\phi$, see (3.8) below. But we could, at the expense of some extra notation, introduce a separate variable parameter $\mu_k$ for the Tikhonov regularization. Moreover, we could also regularize only some variables $y_i$ and not others, depending on the structure of the problem at hand. In particular, the variables that have nonnegativity constraints on them do not need to be regularized, in principle (this would be clear from the subsequent developments). But we shall not go into theoretical analysis of such modifications, as they will cause some technical complications, while the conceptual ideas are clear from our simpler presentation for (3.7). Note that for the two-stage stochastic linear programs (3.2), the corresponding penalty function would be $\phi(x,y) = -\sum_{i=1}^{ny} \ln y_i$ (if $\mu = 0$ is taken).

It is worth to point out that the Tikhonov term makes (3.7) different from the usual log-barrier penalties, but with some similar properties, to be recalled and/or established in the sequel, still holding. At the same time, as we shall explain next, the possibility of adding Tikhonov regularization brings some advantages.

### 3.2.2 Regularized approximate value function

For a penalty parameter $\varepsilon > 0$, the Tikhonov-regularized interior penalty approximation of problem (3.5) is defined by the NLP problem

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & f(x,y) - \varepsilon \sum_{i=1}^{m} \ln\{-g_i(x,y)\} + \varepsilon \frac{\mu}{2} \|y\|^2 \\ \text{subject to} \quad & A(x)y = b(x) \end{aligned} \tag{3.8}$$

(as usual, we use the convention that $\ln t = -\infty$ whenever $t \leq 0$, to drop from (3.8) the implicit interiority constraints $g_i(x,y) < 0$.)

Our main task is to relate the objects obtained from solving (3.8) to solutions of (3.5). To induce the differentiability properties of the interior penalty solutions of (3.8) we shall assume that either the constraints $y \geq 0$ are present among the inequality constraints in (3.5), and/or that the regularization parameter $\mu > 0$ is taken in (3.7). As a result, with our construction, it holds that:

the objective function in (3.8) is strictly or strongly convex, and its Hessian is positive definite everywhere. (3.9)

This leads to uniqueness of solutions and eventual differentiability of the solutions mappings. For this reason, when $y \geq 0$ is not present in (3.5), the Tikhonov term should be added. Otherwise, we do not need to use it, at least if we know that (3.8) has a solution for $\mu = 0$. The latter is closely related to the solution set of (3.5) being nonempty and bounded for the given $x$; see, e.g., [DS99; MZ98] for some results in this direction. But in any case, we can still use the regularization ($\mu > 0$) as well; for example, to make sure that (3.8) is solvable without any extra assumptions.

The unique solution to the regularized problem (3.8) defines the estimate of the value function in (3.6), as follows:

$$v^{\varepsilon}(x) := f(x, y^{\varepsilon}(x)), \quad \text{for } y^{\varepsilon}(x) \text{ solving (3.8)}. \tag{3.10}$$

We consider that $v(x) = v^0(x)$, which is justified by the fact that $v^{\varepsilon}(x) \searrow v(x)$ as $\varepsilon \searrow 0$ (see below for details). We shall also refer to $v^{\varepsilon}(x)$ as *upper smoothing* (of the value function $v(x)$), which would be justified once it is shown that the mapping $y^{\varepsilon}(x)$ is differentiable (then so is $v^{\varepsilon}(x)$). The derivative of $v^{\varepsilon}(x)$ involves the dual mapping $\lambda^{\varepsilon}(x)$, the Lagrange multiplier associated to the solution $y^{\varepsilon}(x)$ of (3.8). Note that this multiplier is well defined (by the linearity of the constraints in (3.8)) whenever so is $y^{\varepsilon}(x)$. In that case, the multiplier is also unique, because $A(x)$ has full row rank, by assumption.

## 3.3 The approximate optimal value function and approximating solution mappings differentiability

We now examine how the function (3.10) approximates the value function (3.6), and derive formulæ for the derivatives of the associated primal and dual solution mappings $y^{\varepsilon}(x)$ and $\lambda^{\varepsilon}(x)$, respectively.

### 3.3.1 Estimates for the optimal value

In this subsection, the analysis concerns a fixed parameter $x$.

Our penalty approximation (3.8) of the original problem (3.5) can be considered to be part of the larger class of interior penalty methods; see [FM68; Wri97]. That said, we are not aware of coupling interior penalties with the Tikhonov regularization, as we do here. Nevertheless, it can be checked directly that certain basic properties hold for this modification as well. In particular, take $0 < \varepsilon_2 < \varepsilon_1$. As in the classical setting ($\mu = 0$, as in [Wri97]), it can be seen that

$$v(x) \leq f(x, y^{\varepsilon_2}(x)) \leq f(x, y^{\varepsilon_1}(x)) \quad \text{and} \quad \phi(x, y^{\varepsilon_2}(x)) \geq \phi(x, y^{\varepsilon_1}(x)).$$

Also, as $\varepsilon \searrow 0$, the accumulation points of $y^{\varepsilon}(x)$ are solutions of (3.5), and $v^{\varepsilon}(x) = f(x, y^{\varepsilon}(x))$ decreases to $v(x)$, the optimal value of (3.5). Moreover, when $\mu = 0$, the following uniform estimate for the value function holds:

$$v(x) \leq v^{\varepsilon}(x) \leq v(x) + m\varepsilon, \quad (\text{ when } \mu = 0)$$

see, e.g., [IS06]. The next proposition generalizes the bound in question to the possibility of using Tikhonov regularization, as in (3.8).

**Proposition 3.3.1** (Value function bounds)**.** *For any $\mu \geq 0$ and any $\varepsilon > 0$, if $y^\varepsilon(x)$ exists then it holds that*

$$v(x) \leq v^\varepsilon(x) \leq v(x) + m\varepsilon + \varepsilon\frac{\mu}{2}\min_{y\in S(x)}\|y\|^2. \tag{3.11}$$

*If $\mu > 0$, then $y^\varepsilon(x)$ exists for any $\varepsilon > 0$, and it holds in addition that*

$$\frac{\mu}{2}\min_{y\in S(x)}\|y\|^2 + m \geq \frac{\mu}{2}\|y^\varepsilon(x)\|^2. \tag{3.12}$$

*Proof.* Recall that $x$ is fixed here. Let $\bar\eta_i := -\varepsilon/g_i(x,y^\varepsilon(x)) > 0$ for all $i = 1,\dots,m$. As is easily seen, the KKT optimality conditions for problem (3.8) characterize $y^\varepsilon(x)$ as a minimizer of

$$L(y) := f(x,y) + \varepsilon\frac{\mu}{2}\|y\|^2 + \sum_{i=1}^m \bar\eta_i g_i(x,y),$$

over the set defined by $A(x)y = b(x)$. Hence, for all $y \in \mathbb{R}^{ny}$ such that $A(x)y = b(x)$, it holds that

$$\begin{aligned} f(x,y) + \varepsilon\frac{\mu}{2}\|y\|^2 + \sum_{i=1}^m \bar\eta_i g_i(x,y) &= L(y) \geq L(y^\varepsilon(x)) \\ &= f(x,y^\varepsilon(x)) + \varepsilon\frac{\mu}{2}\|y^\varepsilon(x)\|^2 + \sum_{i=1}^m \bar\eta_i g_i(x,y^\varepsilon(x)) \\ &= v^\varepsilon(x) + \varepsilon\frac{\mu}{2}\|y^\varepsilon(x)\|^2 - m\varepsilon. \end{aligned} \tag{3.13}$$

Since $\bar\eta_i g_i(x,y) \leq 0$ and $f(x,y) = v(x)$ for all $y \in S(x)$, this means that

$$v^\varepsilon(x) \leq m\varepsilon + \inf_{y\in S(x)}\left\{f(x,y) + \varepsilon\frac{\mu}{2}\|y\|^2\right\} = v(x) + m\varepsilon + \varepsilon\frac{\mu}{2}\inf_{y\in S(x)}\|y\|^2,$$

which is the right-most inequality of (3.11); while the left-most inequality is obvious.

Similarly, but using also that $v^\varepsilon(x) \geq v(x)$, from (3.13) we obtain that

$$v(x) + \varepsilon\frac{\mu}{2}\inf_{y\in S(x)}\|y\|^2 = \inf_{y\in S(x)}\left\{f(x,y) + \varepsilon\frac{\mu}{2}\|y\|^2\right\} \geq v(x) + \varepsilon\frac{\mu}{2}\|y^\varepsilon(x)\|^2 - m\varepsilon.$$

Now dividing the latter inequality by $\varepsilon > 0$ and re-arranging terms results in (3.12). □

We do not claim that the estimate (3.11) is tight, but it will turn out to be sufficient for many purposes. To improve the bound, one would need to estimate how far $y^\varepsilon(x)$ is from the solution in $S(x)$ of minimal norm .

The example $\min\{-y : y \leq 1\}$ shows that that the classical bound $v^\varepsilon(x) \leq v(x) + m\varepsilon$ is not valid for $\mu > 0$. The claim can be checked explicitly (and is quite clear intuitively), because $\|y^\varepsilon(x)\|^2 < 1 = \min_{y\in S(x)}\|y\|^2$.

Now it is clear that although $\mu > 0$ has the advantage of guaranteeing the existence of $y^\varepsilon(x)$ (and this is without any assumptions), in the parametric context the price to pay is the loss of the uniform approximation of $v(x)$ given by $v^\varepsilon(x)$, because we have to deal with the term $\min_{y\in S(x)}\|y\|^2$ that now appears in the bound.

In the analysis below, for $\bar x \in \mathbb{R}^{nx}$ fixed, we want to know whether $\lim_{\varepsilon\searrow 0, x\to\bar x} v^\varepsilon(x) = v(\bar x)$. Observe that boundedness of the solution set $S(\bar x)$ is not necessarily relevant. For instance, consider $\min\{yx^2 : y \geq 0\}$. We have $S(x) = \{0\}$ for $x > 0$, $S(0) = \{y \in \mathbb{R} : y \geq 0\}$, and $\lim_{\varepsilon\searrow 0, x\to\bar x} v^\varepsilon(x) = v(\bar x)$ holds trivially.

Clearly, one way to ensure that $\lim_{\varepsilon\searrow 0, x\to\bar x} v^\varepsilon(x) = v(\bar x)$ is to guarantee (somehow) that there exists $K > 0$ such that $\min_{y\in S(x)}\|y\|^2 \leq K$ for $x \in \mathbb{R}^{nx}$ close to $\bar x$. For instance, if $S(\bar x)$ is locally bounded at $\bar x \in \mathbb{R}^{nx}$, then such $K > 0$ obviously exists. So this is not very restrictive. However, one of the advantages of considering $\mu > 0$ is that we can deal with unbounded solution sets. The constant $K > 0$ in question does not exist if and only if there is a sequence $x_k \to \bar x$ such that $\min_{y\in S(x_k)}\|y\|^2 \to \infty$. This is clearly something rare/pathological, and can be disregarded in a general approach, like the one we are presenting. Accordingly, where needed, we make the reasonable assumption that

$$\limsup_{x\to\bar x}\left\{\min_{y\in S(x)}\|y\|^2\right\} < +\infty. \tag{3.14}$$

Just note that (3.14) always holds if the solution sets are locally bounded, which in turn is automatic if the *feasible* sets in (3.5) are uniformly locally bounded (the latter being quite an acceptable assumption by itself, holding in many cases of interest).

**Remark 3.3.1.** [Consequences of assumption (3.14)] First, (3.14) and (3.11) imply that $v(x)$ is continuous. Under (3.14), the bound (3.11) implies $\limsup_{\varepsilon\searrow 0, x\to\bar{x}} y^\varepsilon(x) \subset S(\bar{x})$. However, this does not imply the existence of accumulation points of $y^\varepsilon(x)$. When $\mu > 0$ we can use (3.12) to conclude under (3.14) that $y^\varepsilon(x)$ remains uniformly bounded for small $\varepsilon > 0$ and $x$ close to $\bar{x} \in \mathbb{R}^{nx}$, even if $S(\bar{x})$ is unbounded. The case $\mu = 0$ is not as straightforward. If $\mu = 0$ we have to assume that $S(x)$ is bounded for all $x$, so that $y^\varepsilon(x)$ exists. Recalling that convex functions with one nonempty bounded level set are inf-compact and level-bounded, [RW09, Def. 1.8], we can be sure that $y^\varepsilon(x)$ remains bounded for fixed $x \in \mathbb{R}^{nx}$ when we change $\varepsilon > 0$. However, to deal with $x \to \bar{x}$ we shall focus on the case when $S(\bar{x})$ is at least locally bounded, if no regularization is used ($\mu = 0$). For that, we later refer to the condition

$$\limsup_{\varepsilon\searrow 0, x\to\bar{x}} \|y^\varepsilon(x)\| < +\infty \quad \forall \bar{x} \in \mathbb{R}^{nx}. \tag{3.15}$$

Satisfaction of (3.15) is ensured under (3.14) if $\mu > 0$, and under local boundedness of the feasible sets when $\mu = 0$. Condition (3.14) holds, for instance, if the feasible sets are uniformly bounded. An example (due to a referee) for which (3.14) fails is $\min x^2 y$ s.t. $xy \in [-1, 1]$. A more general form of condition (3.14) is also mentioned in [Guo+14] as the restricted inf-compactness condition. The difference is that [Guo+14] allows $S(x)$ to be empty.

## 3.3.2 Parametric differentiability

The regularized approximating problem (3.8) is explicitly set-up to satisfy the associated second-order sufficient optimality condition (by (3.9), either because of the Tikhonov regularization term with $\mu > 0$ or because of the log-barrier penalization of the $y \geq 0$ constraints when they are present). Then, given also the linear independence constraint qualification (by the full rank assumption on the matrices $A(x)$), the differentiability of the mappings $y^\varepsilon(x)$ and $\lambda^\varepsilon(x)$ can be obtained applying to (3.8) some classical results. We give some details of a direct proof in our case, because the calculations of the derivatives are needed for later developments in any case.

The KKT conditions for (3.8) give the following parametric system of nonlinear equations (in primal-dual variables):

$$\begin{aligned} &\nabla_y f(x, y^\varepsilon(x)) + \varepsilon \nabla_y \phi(x, y^\varepsilon(x)) - A(x)^\top \lambda^\varepsilon(x) = 0, \\ &A(x) y^\varepsilon(x) - b(x) = 0. \end{aligned} \tag{3.16}$$

(Note that we used the constraint in the form of $b(x) - A(x)y = 0$ to assign the Lagrange multiplier $\lambda^\varepsilon(x)$ at the solution $y^\varepsilon(x)$ in the first equation above, but then we reversed the sign of the constraint to "the original" in the second equation. This is quite common. Here, we opted for this form for a certain convenience later on.)

Differentiability of the primal-dual solution mappings depends on properties of the Jacobian of (3.16), which is given by

$$J^\varepsilon(x) := \begin{bmatrix} M^\varepsilon(x) & -A(x)^\top \\ A(x) & 0 \end{bmatrix}, \text{ for } M^\varepsilon(x) := \nabla^2_{yy} f(x, y^\varepsilon(x)) + \varepsilon \nabla^2_{yy} \phi(x, y^\varepsilon(x)). \tag{3.17}$$

This is shown below, together with some useful relations to compute the solution mapping derivatives.

**Theorem 3.3.1** (Smoothness of Solution Mappings)**.** *Let $\varepsilon > 0$ be fixed. For all $x \in \mathbb{R}^{nx}$, assume that $y^\varepsilon(x)$ exists (which is automatic if $\mu > 0$).*

*Then the following holds:*

*(i) The mappings $y^\varepsilon(x)$ and $\lambda^\varepsilon(x)$ are $C^1$-functions of the parameter $x \in \mathbb{R}^{nx}$.*

*(ii) For $j = 1, \ldots, nx$ the corresponding partial derivatives*

$$d^\varepsilon_j(x) := \frac{\partial y^\varepsilon(x)}{\partial x_j} \quad \textit{and} \quad \delta^\varepsilon_j(x) := \frac{\partial \lambda^\varepsilon(x)}{\partial x_j} \tag{3.18}$$

*can be computed by solving the linear system*

$$J^\varepsilon(x) \begin{bmatrix} d^\varepsilon_j(x) \\ \delta^\varepsilon_j(x) \end{bmatrix} = \begin{bmatrix} \theta^\varepsilon_j(x) + \varepsilon \varphi^\varepsilon_j(x) \\ \beta^\varepsilon_j(x) \end{bmatrix}, \tag{3.19}$$

*where $J^\varepsilon(x)$ is given by* (3.17)*, and the right-hand side terms are*

$$\begin{aligned}
\theta_j^\varepsilon(x) &:= -\left.\frac{\partial \nabla_y f(x,y)}{\partial x_j}\right|_{y=y^\varepsilon(x)} + \frac{\partial A(x)^\top}{\partial x_j}\lambda^\varepsilon(x)\,,\\
\varphi_j^\varepsilon(x) &:= -\left.\frac{\partial \nabla_y \phi(x,y)}{\partial x_j}\right|_{y=y^\varepsilon(x)}\\
\beta_j^\varepsilon(x) &:= \frac{\partial b(x)}{\partial x_j} - \frac{\partial A(x)}{\partial x_j}y^\varepsilon(x)\,.
\end{aligned} \tag{3.20}$$

*Proof.* To show the first item recall that, by construction, (3.9) holds, i.e., the matrix $M^\varepsilon(x)$ in (3.17) is positive definite. Take any $(u_1,u_2) \in \ker J^\varepsilon(x)$, so that

$$M^\varepsilon(x)u_1 - A(x)^\top u_2 = 0, \quad A(x)u_1 = 0\,.$$

Multiplying the first equation above by $u_1^\top$ and using $u_1^\top A(x)^\top = 0$, we conclude that $u_1^\top M^\varepsilon(x)u_1 = 0$. Positive definiteness of $M^\varepsilon(x)$ implies that $u_1 = 0$. Then, by the first equation above, $A(x)^\top u_2 = 0$. As $A(x)$ has full row rank, it follows that $u_2 = 0$. Thus $\ker J^\varepsilon(x) = \{0\}$, i.e., $J^\varepsilon(x)$ is nonsingular.

The conclusions follow from the (second-order) Implicit Function Theorem [Lan93, p. 364]. □

Note that the matrix in the linear systems (3.19) is the same for all $j$. This means that only one matrix factorization is required to solve all the linear systems in question.

The next result states that, once the mapping $y^\varepsilon(x)$ is smooth, so is the approximating value function $v^\varepsilon(x)$, and also gives the expressions for the corresponding derivatives. This justifies the name *upper smoothing* (not to be confused with smoothing in the sense of [Che12]; see Section 3.5) .

In what follows, for notational simplicity we drop the dependencies of some auxiliary quantities on $x$ and $\varepsilon$, as they are clear from the context.

**Corollary 3.3.1** (Smoothed value function derivatives)**.** *With the notation and assumptions in Theorem 3.3.1, for $i = 1,\dots,m$ and $j = 1,\dots,nx$, let*

$$\alpha_j := \nabla_y f(x,y^\varepsilon(x))^\top d_j\,, \quad \gamma_{ij} := \frac{\nabla_y g_i(x,y^\varepsilon(x))^\top d_j}{g_i(x,y^\varepsilon(x))}\,.$$

*Then it holds that*

*(i) For each $j = 1,\dots,nx$,*

$$\alpha_j = -\mu\varepsilon y^\varepsilon(x)^\top d_j + \varepsilon\sum_{i=1}^m \gamma_{ij} + \beta_j^\top \lambda^\varepsilon(x). \tag{3.21}$$

*(ii) The derivatives of the smoothed value function* (3.10) *are given by*

$$\frac{\partial v^\varepsilon(x)}{\partial x_j} = \alpha_j + \frac{\partial f(x,y^\varepsilon(x))}{\partial x_j}\,,$$

*for $j = 1,\dots,nx$.*

*Proof.* Multiplying the transpose of the first identity in (3.16) by $d_j^\top$ gives

$$\alpha_j + \varepsilon\nabla_y\phi(x,y^\varepsilon(x))^\top d_j - \lambda^\varepsilon(x)^\top A(x)d_j = 0\,.$$

For $i = 1,\dots,m$, define

$$\eta_i := \frac{-\varepsilon}{g_i(x,y^\varepsilon(x))}\,.$$

Taking into account that, by (3.7),

$$\varepsilon\nabla_y\phi(x,y^\varepsilon(x)) = \mu\varepsilon y^\varepsilon(x) + \sum_{i=1}^m \eta_i\nabla_y g_i(x,y^\varepsilon(x)),$$

and $A(x)d_j = \beta_j$ by (3.19), yields (3.21).

The second item is just the chain rule, combined with Theorem 3.3.1. □

Keeping in mind Proposition 2.1.1, formula (3.11), and the fact that the closure of the smoothed gradients provides an upper bound for the subdifferential of the value function, we would be able to conclude the local Lipschitz continuity of the value function $v$ at a point $\bar{x}$ from boundedness of the gradient of $v^\varepsilon$, by examining the limit of the latter as $x \to \bar{x}$ and $\varepsilon \searrow 0$ (see Section 3.5 for details). Here, we just point out that in view of Corollary 3.3.1, it will suffice to check boundedness of the terms defining the derivatives in item (ii). The right-most term will be dealt with by means of (3.15), and by smoothness of $f$ and of the regularized solution mapping $y^\varepsilon$. By contrast, bounding the terms $\alpha_j$ is far more involved, and this is the reason for singling out the expression (3.21) in item (i): the terms therein appear in various inequalities stated in the next section.

## 3.4 Technical bounds

The results in this section aim at showing that, although $d_j$ defined in (3.18) can blow up as $x \to \bar{x}$ and $\varepsilon \searrow 0$, under reasonable conditions the terms $\alpha_j$ defined in Corollary 3.3.1 stay bounded (see Theorem 3.5.1 below). The strategy we use to do such analysis is not complicated, but the technical details involve many calculations and bounds.

**Proposition 3.4.1.** *With the notation and assumptions in Theorem 3.3.1 and Corollary 3.3.1, the following relations hold for the matrix $M = M^\varepsilon(x)$ defined in* (3.17)*, and $\theta_j = \theta_j^\varepsilon(x)$, $\varphi_j = \varphi_j^\varepsilon(x)$, $\delta_j = \delta_j^\varepsilon(x)$ and $\beta_j = \beta_j^\varepsilon(x)$:*

*(i)* $Md_j = \nabla^2_{yy} f(x, y^\varepsilon(x))d_j + \sum_{i=1}^m \eta_i \nabla^2_{yy} g_i(x, y^\varepsilon(x))d_j - \sum_{i=1}^m \eta_i \gamma_{ij} \nabla_y g_i(x, y^\varepsilon(x)) + \varepsilon\mu d_j$.

*(ii)* $d_j^\top M d_j = d_j^\top \theta_j + \varepsilon d_j^\top \varphi_j + \delta_j^\top \beta_j$.

*(iii)* $d_j^\top M d_j \geq \varepsilon \sum_{i=1}^m \gamma_{ij}^2 + \varepsilon\mu \|d_j\|^2$.

*Proof.* Let $\mathbb{I}$ denote the identity matrix of order $ny$. By the definition of the penalty function in (3.7),

$$\begin{aligned}\varepsilon \nabla^2_{yy}\phi(x, y^\varepsilon(x)) \quad &= \quad \sum_{i=1}^m \frac{-\varepsilon}{g_i(x, y^\varepsilon(x))} \nabla^2_{yy} g_i(x, y^\varepsilon(x)) \\ &\qquad - \sum_{i=1}^m \frac{-\varepsilon}{g_i(x, y^\varepsilon(x))} \nabla_y g_i(x, y^\varepsilon(x)) \frac{\nabla_y g_i(x, y^\varepsilon(x))^\top}{g_i(x, y^\varepsilon(x))} + \varepsilon\mu\mathbb{I}.\end{aligned}$$

The expression in item (i) follows, after multiplying by $d_j$ and recalling the definitions of $\eta_i$, $\gamma_{ij}$, and of $M$.

Next, multiplying on the left (3.19) by the vector $(d_j^\top, \delta_j^\top)$; using the expression in (3.17) for the Jacobian matrix $J$, it follows that

$$(d_j^\top, \delta_j^\top) J \begin{bmatrix} d_j \\ \delta_j \end{bmatrix} = \begin{bmatrix} d_j^\top M d_j - d_j^\top A(x)^\top \delta_j \\ \delta_j^\top A(x) d_j \end{bmatrix} = \begin{bmatrix} d_j^\top \theta_j + \varepsilon d_j^\top \varphi_j \\ \delta_j^\top \beta_j \end{bmatrix}.$$

The first line gives item (ii), because $A(x)d_j = \beta_j$.

In the relation for $Md_j$ shown in item (i), the Hessians of $f$ and $g_i$ are positive semidefinite, by convexity of the objective and constraint functions (the implicit constraints $g_i(x,y) < 0$ make $\eta_i > 0$). Accordingly,

$$\begin{aligned} d_j^\top M d_j &\geq - \sum_{i=1}^m \frac{-\varepsilon}{g_i(x, y^\varepsilon(x))} d_j^\top \nabla_y g_i(x, y^\varepsilon(x)) \frac{\nabla_y g_i(x, y^\varepsilon(x))^\top d_j}{g_i(x, y^\varepsilon(x))} + \varepsilon\mu\|d_j\|^2 \\ &= \varepsilon \sum_{i=1}^m \gamma_{ij}^2 + \varepsilon\mu\|d_j\|^2, \end{aligned}$$

which completes the proof. □

The arguments that follow aim at finding upper bounds for the term $d_j^\top M d_j$ in Proposition 3.4.1(iii). This is done by bounding from above all the terms in the expression given in Proposition 3.4.1(ii). To this aim, our next result states boundedness of $\eta_i = -\varepsilon / g_i(x, y^\varepsilon(x))$, the Lagrange multiplier estimates for inequality constraints, obtained after solving the interior penalty subproblem (3.8). In the non-parametric case, such results (under appropriate constraint qualifications) are quite classical. Here, we give an extension to the parametric setting of this chapter.

But first, we shall need the following property.

**Lemma 3.4.1** (Continuity of Projections)**.** *Let $Y(x)$ be the feasible set of* (3.5) *for a parameter $x \in \mathbb{R}^{nx}$, and let $\mathring{y}(\bar{x}) \in \mathbb{R}^{ny}$ be a Slater point for the fixed parameter $\bar{x} \in \mathbb{R}^{nx}$.*

*It holds that the mapping $P(x)$ of orthogonally projecting the (fixed) point $\mathring{y}(\bar{x})$ onto $Y(x)$ is continuous around $\bar{x} \in \mathbb{R}^{nx}$.*

*Proof.* The assertion follows applying [FI90, Theorem 5.1] to the parametric optimization problem

$$\min \|y - \mathring{y}(\bar{x})\|^2 \text{ s.t. } y \in Y(x),$$

where $x \in \mathbb{R}^{nx}$ is the parameter.

Some details. The solution of this problem for the parameter $x = \bar{x}$ is obviously $\mathring{y}(\bar{x})$. By the Slater condition, $g(\bar{x}, \mathring{y}(\bar{x})) < 0$. Hence, the strict complementarity condition and the linear independence of active gradients are automatic for this problem with the parameter $x = \bar{x}$ (the latter because $A(\bar{x})$ has full rank). Finally, the second-order sufficient optimality condition holds by strong convexity of the projecting objective function. Then [FI90, Theorem 5.1] implies that the solution mapping $P(x)$ of the problem in question is smooth around $\bar{x}$. □

**Remark 3.4.1.** If the point $\mathring{y}(\bar{x})$ in Lemma 3.4.1 were to be changed to an arbitrary (but fixed) point, we could still use the inequality (3.11), written for the projection problem, to conclude that $P(x)$ is continuous if the projection $P(x)$ is locally bounded. This is possible because inequality (3.11) shows that there is a sequence of smooth functions converging locally uniformly to $P(x)$.

**Lemma 3.4.2** (Local Boundedness of Multiplier Estimates $\eta_i$)**.** *Assume that $y^\varepsilon(x)$ exists for all $x \in \mathbb{R}^{nx}$ (which is automatic if $\mu > 0$), and that* (3.15) *holds at $\bar{x} \in \mathbb{R}^{nx}$. Then for all $i = 1, \dots, m$,*

$$0 \leq \limsup_{\varepsilon \searrow 0, x \to \bar{x}} \eta_i < +\infty. \tag{3.22}$$

*Proof.* To show (3.22) suppose, for contradiction purposes, that there exist $\varepsilon_k \searrow 0$ and $x_k \to \bar{x}$ such that for some $i = 1, \dots, m$ it holds that $\{-\varepsilon_k / g_i(x_k, y^{\varepsilon_k}(x_k))\} \to +\infty$. Taking subsequences of $\{\varepsilon_k\}$ and $\{x_k\}$ we can get a partition of $\{1, \dots, m\} = I_0 \cup I_\infty$ where for all $i \in I_0$ the sequences $\{-\varepsilon_k / g_i(x_k, y^{\varepsilon_k}(x_k))\}$ remain bounded, while

$$-\varepsilon_k / g_i(x_k, y^{\varepsilon_k}(x_k)) \to +\infty \text{ for } i \in I_\infty. \tag{3.23}$$

Denote by $Y(x)$ the feasible set of (3.5) for a parameter $x \in \mathbb{R}^{nx}$. Let $\mathring{y}(\bar{x}) \in \mathbb{R}^{ny}$ be a Slater point for the parameter $\bar{x} \in \mathbb{R}^{nx}$ (which exists by the blanket assumptions). Define $y_k = \mathrm{P}_{Y(x_k)}(\mathring{y}(\bar{x}))$ to be the projection of $\mathring{y}(\bar{x})$ onto $Y(x_k)$. By Lemma 3.4.1, we have that $y_k \to \mathring{y}(\bar{x})$. Also by continuity, there exists some $\Gamma > 0$ such that, for all $k$ large enough,

$$g_i(x_k, y_k) \leq -\frac{\Gamma}{2} < 0 \quad \text{for all } i \in I_0 \cup I_\infty, \tag{3.24}$$

because $g_i(\bar{x}, \mathring{y}(\bar{x})) \leq -\Gamma < 0$ for all $i \in I_0 \cup I_\infty$.

Define

$$u_k := y_k - y^{\varepsilon_k}(x_k),$$

and note that

$$A(x_k) u_k = 0.$$

Take $i \in I_\infty$. By (3.23), we have that $g_i(x_k, y^{\varepsilon_k}(x_k)) \to 0$. By convexity,

$$g_i(x_k, y_k) \geq g_i(x_k, y^{\varepsilon_k}(x_k)) + [\nabla_y g_i(x_k, y^{\varepsilon_k}(x_k))]^\top u_k.$$

Using $g_i(x_k, y^{\varepsilon_k}(x_k)) \to 0$ and (3.24), we can assume that for all $k$ large enough it holds that

$$\nabla_y g_i(x_k, y^{\varepsilon_k}(x_k))^\top u_k \leq -\frac{\Gamma}{4} < 0 \text{ for all } i \in I_\infty. \tag{3.25}$$

Multiplying on the left by $u_k^\top$ the KKT condition (3.16) written with $(\varepsilon_k, x_k)$, we see that

$$u_k^\top \nabla_y f(x_k, y^{\varepsilon_k}(x_k)) + \varepsilon_k \nabla_y \phi(x_k, y^{\varepsilon_k}(x_k))^\top u_k = u_k{}^\top [A(x_k)^\top \lambda^{\varepsilon_k}(x_k)] = 0,$$

because $A(x_k) u_k = 0$. Since by (3.7),

$$\varepsilon_k \nabla_y \phi(x_k, y^{\varepsilon_k}(x_k)) = \sum_{i \in I_0 \cup I_\infty} \eta_i \nabla_y g_i(x_k, y^{\varepsilon_k}(x_k)) + \mu \varepsilon_k y^{\varepsilon_k}(x_k),$$

it follows that

$$u_k^\top \nabla_y f(x_k, y^{\varepsilon_k}(x_k)) - \varepsilon_k \sum_{i \in I_0 \cup I_\infty} \frac{\nabla_y g_i(x_k, y^{\varepsilon_k}(x_k))^\top u_k}{g_i(x_k, y^{\varepsilon_k}(x_k))} + \mu \varepsilon_k u_k^\top y^{\varepsilon_k}(x_k) = 0\,.$$

Hence,

$$\begin{aligned} & -\varepsilon_k \sum_{i \in I_\infty} \frac{\nabla_y g_i(x_k, y^{\varepsilon_k}(x_k))^\top u_k}{g_i(x_k, y^{\varepsilon_k}(x_k))} \\ = \quad & -u_k^\top \nabla_y f(x_k, y^{\varepsilon_k}(x_k)) + \varepsilon_k \sum_{i \in I_0} \frac{\nabla_y g_i(x_k, y^{\varepsilon_k}(x_k))^\top u_k}{g_i(x_k, y^{\varepsilon_k}(x_k))} - \mu \varepsilon_k u_k^\top y^{\varepsilon_k}(x_k)\,. \end{aligned}$$

The left-hand side in the equality above tends to $-\infty$ as $k \to \infty$, by (3.23) and (3.25). The sequence $\{y^{\varepsilon_k}(x_k)\}$ is bounded because of (3.15). Then, $\{u_k\}$ is also bounded, as well as all the terms in the right-hand side of the equality above. Thus, we have a contradiction. This proves (3.22). □

We then obtain the following.

**Corollary 3.4.3** (Local Boundedness of the Smoothed Dual Solution Mapping)**.** *Under the assumptions of Lemma 3.4.2, for any $\bar{x} \in \mathbb{R}^{nx}$ there exist $\rho > 0$ and $C > 0$ such that, for all $\varepsilon \in (0, \rho)$ and $x \in B(\bar{x}, \rho)$, it holds that*

*(i)* $\|Md_j\| \le C\|d_j\| + C\sum_{i=1}^{m} |\gamma_{ij}|\,.$

*(ii) The set $\{\lambda^\varepsilon(x) : x \in B(\bar{x}, \rho)\,, \varepsilon \in (0, \rho]\}$ is bounded.*

*Proof.* By Proposition 3.4.1(i),

$$\begin{aligned} \|Md_j\| \le & \|\nabla^2_{yy} f(x, y^\varepsilon(x))\| \|d_j\| + \sum_{i=1}^{m} \eta_i \|\nabla^2_{yy} g_i(x, y^\varepsilon(x))\| \|d_j\| + \varepsilon\mu \|d_j\| \\ & + \sum_{i=1}^{m} \eta_i \left|\gamma_{ij}\right| \|\nabla_y g_i(x, y^\varepsilon(x))\|\,. \end{aligned}$$

Item (i) follows, by (3.15) and (3.22).

As the matrices $A(x)$ have full rank, from the KKT conditions (3.16) we obtain, in a standard way, that

$$\lambda^\varepsilon(x) = [A(x)A^\top(x)]^{-1} A(x) \left\{ \nabla_y f(x, y^\varepsilon(x)) + \varepsilon\mu y^\varepsilon(x) + \sum_{i=1}^{m} \eta_i \nabla_y g_i(x, y^\varepsilon(x)) \right\}.$$

Item (ii) follows, again by (3.15) and (3.22). □

Keeping Proposition 3.4.1(ii) in mind, we next estimate the behavior of the right-hand side terms in the linear system (3.19).

**Proposition 3.4.2.** *Under the assumptions of Lemma 3.4.2, for all $j = 1, \ldots, nx$ and $\bar{x} \in \mathbb{R}^{nx}$, the quantities $\theta_j = \theta_j^\varepsilon(x)$ and $\varphi_j = \varphi_j^\varepsilon(x)$, defined in* (3.20)*, satisfy the following relations:*

*(i)* $\limsup_{\varepsilon \searrow 0, x \to \bar{x}} \varepsilon^2 \|\varphi_j\| < +\infty$ *and* $\limsup_{\varepsilon \searrow 0, x \to \bar{x}} \|\theta_j\| < +\infty$.

*(ii)* $\varepsilon^2 |d_j^\top \varphi_j| \le \varepsilon K(\|d_j\| + \sum_{i=1}^m \left|\gamma_{ij}\right|)$ *for $\varepsilon \in (0, \delta)$, $x \in B(\bar{x}, \delta)$ and some constant $K = K(\delta, \bar{x}) > 0$.*

*Proof.* Recalling the definition of $\varphi_j$, we obtain that

$$\begin{aligned} \varepsilon^2 \varphi_j &= -\sum_{i=1}^{m} \frac{\partial\left(\varepsilon \eta_i \nabla_y g_i(x, y)\right)}{\partial x_j}\Big|_{y = y^\varepsilon(x)} \\ &= -\varepsilon \sum_{i=1}^{m} \frac{\partial \eta_i}{\partial x_j} \nabla_y g_i(x, y^\varepsilon(x)) - \varepsilon \sum_{i=1}^{m} \Big( \eta_i \frac{\partial \nabla_y g_i(x, y^\varepsilon(x))}{\partial x_j} \Big)\,. \end{aligned}$$

We now bound the right-hand side terms, as follows. First notice that by (3.15) and (3.22), for some $K_1 > 0$ (depending on $\bar{x}$) it holds that for all $x$ close enough to $\bar{x}$ and all $\varepsilon$ close enough to zero,

$$\left\| \varepsilon \sum_{i=1}^{m} \Big( \eta_i \frac{\partial \nabla_y g_i(x, y^\varepsilon(x))}{\partial x_j} \Big) \right\| \leq K_1 \,. \tag{3.26}$$

Regarding the terms in the first summation, recalling the definition of $\eta_i$,

$$\begin{aligned} \varepsilon \frac{\partial \eta_i}{\partial x_j} &= -\varepsilon^2 \frac{\partial \Big( 1/g_i(x,y)) \Big)}{\partial x_j} \Big|_{y=y^\varepsilon(x)} \\ &= \frac{\varepsilon^2}{(g_i(x, y^\varepsilon(x)))^2} \frac{\partial g_i(x,y)}{\partial x_j} \Big|_{y=y^\varepsilon(x)} \\ &= (\eta_i)^2 \frac{\partial g_i(x,y)}{\partial x_j} \Big|_{y=y^\varepsilon(x)} . \end{aligned}$$

Using once more (3.22), together with smoothness of $g_i$ and (3.15), we conclude that the term above is bounded. Therefore, there exists some constant $K_2 > 0$ such that

$$\left\| \varepsilon \sum_{i=1}^{m} \frac{\partial \eta_i}{\partial x_j} \nabla_y g_i(x, y^\varepsilon(x)) \right\| \leq K_2 \,. \tag{3.27}$$

Combining (3.26) with (3.27) gives the first assertion in item (i).

The second assertion in item (i) follows using the smoothness assumptions on $f$ and $A$, (3.15), and item (ii) of Corollary 3.4.3.

Item (ii) follows multiplying the expression above for $\varepsilon^2 \varphi_j$ by $d_j$, and re-examining the terms involved. □

The final estimate of this section is the following.

**Proposition 3.4.3.** *Under the assumptions of Lemma 3.4.2, for all* $j = 1, \dots, nx$ *and* $\bar{x} \in \mathbb{R}^{nx}$ *there exist* $\rho > 0$ *and a constant* $L > 0$ *such that, for all* $\varepsilon \in (0, \rho)$ *and* $x \in B(\bar{x}, \rho)$,

$$\varepsilon^2 \mu \|d_j\|^2 + \varepsilon^2 \sum_{i=1}^{m} \gamma_{ij}^2 \leq \varepsilon L \|d_j\| + \varepsilon L \sum_{i=1}^{m} |\gamma_{ij}| + L. \tag{3.28}$$

*Proof.* Throughout we consider $\varepsilon > 0$ sufficiently small and $x$ close enough to $\bar{x} \in \mathbb{R}^{nx}$. We also drop the dependencies on $\varepsilon$ and $x$, as they are clear from the context. For example, in what follows $M := M^\varepsilon(x)$, $A := A(x)$, as well as $d_j := d_j^\varepsilon(x)$, $\delta_j := \delta_j^\varepsilon(x)$, etc.

By items (ii) and (iii) in Proposition 3.4.1, we have that

$$\varepsilon^2 \mu \|d_j\|^2 + \varepsilon^2 \sum_{i=1}^{m} \gamma_{ij}^2 \leq \varepsilon d_j^\top \theta_j + \varepsilon^2 d_j^\top \varphi_j + \varepsilon \delta_j^\top \beta_j \,. \tag{3.29}$$

To establish (3.28), we proceed to bound the terms in the right-hand side of (3.29).

By items (i) and (ii) in Proposition 3.4.2, for some constant $L_1 > 0$,

$$\varepsilon d_j^\top \theta_j + \varepsilon^2 d_j^\top \varphi_j \leq \varepsilon \|\theta_j\| \|d_j\| + \varepsilon^2 |d_j^\top \varphi_j| \leq \varepsilon L_1 (\|d_j\| + \sum_{i=1}^{m} |\gamma_{ij}|) \,. \tag{3.30}$$

To bound the last term in the right-hand side of (3.29), we first show that, for some constant $L_2 > 0$,

$$\varepsilon \|\delta_j\| \leq \varepsilon L_2 \|d_j\| + \varepsilon L_2 \sum_{i=1}^{m} |\gamma_{ij}| + L_2 \,. \tag{3.31}$$

By the first equation in (3.19), $M d_j - A^\top \delta_j = \theta_j + \varepsilon \varphi_j$ . Multiplying this equation by $A$, as the matrix $AA^\top$ is non-singular, we obtain that

$$\delta_j = (AA^\top)^{-1} A (M d_j - \theta_j - \varepsilon \varphi_j).$$

Since the matrices $(AA^\top)^{-1}A$ (which depend on $x$) are bounded for all $x$ close to $\bar{x}$, for some $L_2 > 0$

$$\begin{aligned} \varepsilon\|\delta_j\| &\leq L_2(\|\varepsilon M d_j\| + \varepsilon\|\theta_j\| + \varepsilon^2\|\varphi_j\| \\ &\leq L_2\varepsilon\|M d_j\| + L_3, \end{aligned}$$

where the second inequality follows from Proposition 3.4.2(i), taking $L_3 > 0$ large enough.

By item (i) in Corollary 3.4.3, for some constant $C > 0$,

$$\|M d_j\| \leq C\|d_j\| + C\sum_{i=1}^{m} |\gamma_{ij}| .$$

Combining the latter relation with (3.32) and taking $L_2 > 0$ large enough, gives (3.31).

By the definition of $\beta_j$ and (3.15), $\|\beta_j\|$ stays bounded as $\varepsilon \searrow 0$ and $x \to \bar{x}$. By (3.31), it then holds that, for some $L_4 > 0$,

$$\varepsilon\delta_j^\top\beta_j \leq \varepsilon\|\delta_j\|\|\beta_j\| \leq \varepsilon L_4\|d_j\| + \varepsilon L_4\sum_{i=1}^{m} |\gamma_{ij}| + L_4 .$$

Combining the latter relation with (3.30), the assertion (3.28) follows from (3.29). □

## 3.5 Boundedness of the smoothing gradients and Lipschitz-continuity of the value function

We shall now discuss some consequences of our analysis above, including boundedness of the derivatives of the proposed smoothing, as well as some issues related to gradient consistency [Che12; BHK13; BH16; BH13], and Lipschitz-continuity of the value function [MNY07; DM14; Guo+14].

We are now in position to combine the various inequalities in Section 3.4 to bound the upper smoothing derivatives given in Corollary 3.3.1.

**Theorem 3.5.1** (Local Uniform Boundedness of Smoothed Gradient)**.** *Assume that the smoothing is built with a fixed $\mu > 0$ and that* (3.14) *holds at $\bar{x} \in \mathbb{R}^{nx}$. Then there exist $\rho > 0$ and $L > 0$ such that*

$$\|\nabla v^\varepsilon(x)\| \leq L \quad \textit{for all } \varepsilon \in (0,\rho) \textit{ and } x \in B(\bar{x},\rho).$$

*Proof.* Take any $j \in \{1,\dots,nx\}$. Recalling Corollary 3.3.1, we have that

$$\frac{\partial v^\varepsilon(x)}{\partial x_j} = -\mu\varepsilon y^\varepsilon(x)^\top d_j + \varepsilon\sum_{i=1}^{m}\gamma_{ij} + \beta_j^\top\lambda^\varepsilon(x) + \frac{\partial f(x,y^\varepsilon(x))}{\partial x_j} . \tag{3.33}$$

The last term in the right-hand side of (3.33) is locally bounded by the assumption (3.15), and the smoothness properties of $f$ and of $y^\varepsilon$ (the latter established in Theorem 3.3.1). As already used before, $\beta_j$ is also bounded, by the same reasons. The mapping $\lambda^\varepsilon$ is locally bounded, as established in Corollary 3.4.3(ii). Hence, the last two terms in the right-hand side of (3.33) are bounded. It remains to analyze the first two terms.

Suppose that the term $\varepsilon\|d_j\|$ is unbounded as $\varepsilon \searrow 0$ and $x \to \bar{x}$. By (3.28), it holds that

$$\varepsilon^2\mu\|d_j\|^2 \leq \varepsilon L\|d_j\| + \varepsilon L\sum_{i=1}^{m} |\gamma_{ij}| + L.$$

As $\mu > 0$, this inequality implies that if $\varepsilon\|d_j\|$ is unbounded, then the term $\varepsilon\sum_{i=1}^{m}|\gamma_{ij}|$ must be unbounded (otherwise the inequality in question yields a contradiction). But both $\varepsilon\|d_j\|$ and $\varepsilon\sum_{i=1}^{m}|\gamma_{ij}|$ being unbounded clearly contradicts (3.28), recalling again that $\mu > 0$. We conclude that $\varepsilon\|d_j\|$ is bounded. Then (3.28) implies that so is $\varepsilon\sum_{i=1}^{m}|\gamma_{ij}|$.

The proof is completed, because we showed that all the terms in the right-hand side of (3.33) are bounded. □

Note, in passing, that the analysis above shows that the following bound on the possible blow-up rate of the derivatives of $y^\varepsilon$ holds:

$$\limsup_{\varepsilon\searrow 0, x\to\bar{x}} \varepsilon\|\nabla y^\varepsilon(x)\| < +\infty.$$

We next make some comments on other notions appearing in the literature on smoothing, and in particular on the property known as *gradient consistency*, as defined in Section 2.3. Gradient consistency was introduced in [CQS98] as Jacobian consistency, and further studied in [BHK13] and [Che12]; see also [QSZ00; RX05].

In the following definitions, a continuous function, possibly nonsmooth, $v : \mathbb{R}^{nx} \to \mathbb{R}$ is given, as well as a differentiable function $\sigma : (0,\infty) \times \mathbb{R}^{nx} \to \mathbb{R}$. The smooth function $\sigma$ is said to be a *smoothing of v in the sense of [Che12]* if

$$\lim_{\varepsilon \searrow 0, x \to \bar{x}} \sigma(\varepsilon, x) = v(\bar{x}). \tag{3.34}$$

The last condition is stronger than the one mentioned in Section 2.3. In our context $v$ is the optimal value function of problem (3.5) while, for the given regularization/penalization parameter $\varepsilon > 0$ appearing in (3.8), we have $\sigma(\varepsilon, x) = v^\varepsilon(x) = f(y^\varepsilon(x), x)$, with $y^\varepsilon(x)$ being the solution of problem (3.8).

When gradient consistency was defined in [Che12], the motivating smoothing functions considered there were explicit, and so local boundedness of the gradients was something granted, in a sense. In our case, the situation is different (as our smoothing function is implicit), and indeed we had to prove that its gradients remain bounded. In general, boundedness/unboundedness is relevant, because of the formula for the Clarke subdifferential that involves the horizon subdifferential (see Section 2.1). This information is not present ("missing") in (2.2) and (2.1), as those conditions are intended for bounded sequences of gradients. In this chapter (as a side issue, not our principal concern) we prove that the horizon subdifferential of $v$ and horizon closure of the smoothed gradients ($\limsup^\infty$) are equal, which is part of what is needed in the general gradient consistency theory, beyond the current definition (2.2) for locally Lipschitz functions (where the smoothing function has locally bounded gradients). For instance, consider the problem $v(x) = \min_y xy$ s.t. $y \geq 0$ and the smoothing of the value function with $\mu > 0$. There are unbounded smoothed gradients around $x = 0$. Clearly, our assumptions do not hold for this $v$ because $S(x) = \emptyset$ if $x < 0$.

In the statements below, we make use of the condition (3.14), whose consequences were discussed in Remark 3.3.1.

**Lemma 3.5.1** (Gradient Consistency). *The following holds true:*

*(i) If $\mu = 0$, or if $\mu > 0$ and* (3.14) *holds, then the function $v^\varepsilon$ is a smoothing for $v$ in the sense of [Che12] (i.e.,* (3.34) *holds for $\sigma(\varepsilon, x) = v^\varepsilon(x)$).*

*(ii) If $\mu > 0$ and* (3.14) *holds, then $w^\varepsilon(x) := v^\varepsilon(x) + \varepsilon\phi(x, y^\varepsilon(x))$ is also a smoothing in the sense of [Che12], and it has locally bounded gradients.*

*(iii) If problem* (3.5) *has parameters only on the map $b$ and $b$ is affine, (i.e.,* (3.5) *has only right-hand side linear perturbations), then $v$ and $w^\varepsilon$ are convex.*

*(iv) Under the assumptions of Theorem 3.5.1, if $v$ is convex and $w^\varepsilon$ is convex for $\varepsilon > 0$ small enough, then $w^\varepsilon$ is gradient consistent with $v$ (i.e.,* (2.2) *holds for $\sigma(\varepsilon, x) = w^\varepsilon(x)$).*

*Proof.* When $\mu = 0$, or (3.14) holds for $\mu > 0$, the relation (3.11) in Proposition 3.3.1 and the continuity of $v$ imply that

$$\lim_{\varepsilon \searrow 0, x \to \bar{x}} v^\varepsilon(x) = \lim_{x \to \bar{x}} v(x) = v(\bar{x}).$$

Item (i) follows.

Let us now prove item (ii). In view of item (i), to show that $w^\varepsilon$ is a smoothing of $v$ we have to verify that $\varepsilon\phi(x', y^\varepsilon(x')) \to 0$ when $\varepsilon \searrow 0$ and $x' \to x$. Recall that $\varepsilon\mu\|y^\varepsilon(x')\| \to 0$ when $\varepsilon \searrow 0$ and $x' \to x$, due to (3.14) and (3.12). Next, using Lemma 3.4.2 and (3.14), we can conclude that $\varepsilon \ln\{-g_i(x', y^\varepsilon(x'))\} \to 0$ since there is $C > 0$ such that $\varepsilon \leq -Cg_i(x', y^\varepsilon(x'))$ and $y^\varepsilon(x')$ is bounded. We conclude that $\varepsilon\phi(x', y^\varepsilon(x')) \to 0$, and thus $w^\varepsilon$ is a smoothing of $v$.

Computing the gradient of $w^\varepsilon$ via the chain rule we see that it is locally bounded, because $\varepsilon d_j$ and $\varepsilon\gamma_{ij}$ are bounded (see the proof of Theorem 3.5.1), and the other terms can be bounded using (3.14), (3.12) and Lemma 3.4.2.

We proceed to item (iii). Consider the problem $\tilde{v}(x) = \min_y \tilde{f}(y)$ s.t. $Ay = b(x)$, where $\tilde{f}(y)$ is a convex extended-valued function, and the problem has solutions for all $x$. For $v$ we shall have $\tilde{f} = f + I_D$, where $I_D$ is the indicator function of the set $D = \{x : g(x) \leq 0\}$, and for $w^\varepsilon$ we have $\tilde{f} = f + \varepsilon\phi$. Denote by $\tilde{y}(x)$ any solution for a fixed $x$. Taking any $x_1, x_2$ and $t \in (0,1)$, the point $t\tilde{y}(x_1) + (1-t)\tilde{y}(x_2)$ is feasible for the problem at parameter $x = tx_1 + (1-t)x_2$. It follows by the convexity of $\tilde{f}$ that $\tilde{v}(tx_1 + (1-t)x_2) \leq \tilde{f}(t\tilde{y}(x_1) + (1-t)\tilde{y}(x_2)) \leq t\tilde{v}(x_1) + (1-t)\tilde{v}(x_2)$. This shows convexity of $\tilde{v}$, i.e., of $v$ and $w^\varepsilon$.

To establish item (iv), fix $\bar{x} \in \mathbb{R}^{nx}$. By the convexity of $w^{\varepsilon}$, for any $x$ and $x'$ it holds that

$$w^{\varepsilon}(x') \geq w^{\varepsilon}(x) + \nabla w^{\varepsilon}(x)^{\top}(x' - x).$$

Note that $\lim_{\varepsilon \searrow 0} w^{\varepsilon}(x') = v(x')$ and $\lim_{\varepsilon \searrow 0, x \to \bar{x}} w^{\varepsilon}(x) = v(\bar{x})$. Then, passing onto the limits $\varepsilon \searrow 0$ and $x \to \bar{x}$ in the inequality above, for any $u \in \limsup_{\varepsilon \searrow 0, x \to \bar{x}} \nabla w^{\varepsilon}(x)$ we see that it must hold that

$$v(x') \geq v(\bar{x}) + u^{\top}(x' - \bar{x}).$$

(Note that such $u$ exists, by item (ii).) This shows that $u$ is a subgradient of the convex function $v$ at $\bar{x}$, implying gradient consistency. □

**Remark 3.5.1.** Note that our smoothing can also be seen in the context of Attouch's Theorem [Att77]. Then, Lemma 3.5.1 could be viewed as an "implementation" of Attouch's Theorem, in the sense that our approximating functions are computable.

We finish by showing that the optimal value function $v$ is locally Lipschitz under our assumptions. We note that a more general result is available in [Guo+14]. However, here we obtain the locally Lipschitz property of $v$ as a simple by-product of our algorithmic smoothing approach.

**Theorem 3.5.2** (Local Lipschitz Continuity of the Value Function)**.** *In addition to the blanket assumptions stated in Section 4.2, assume that condition* (3.14) *holds for* $\bar{x} \in \mathbb{R}^{nx}$.

*Then the optimal value function v is locally Lipschitz continuous in the neighborhood of* $\bar{x}$.

*Proof.* Take any $\mu > 0$ and consider (3.8), which in this case always has solution $y^{\varepsilon}(x)$, for every $\varepsilon > 0$. As (3.14) is assumed, by Lemma 3.5.1 we know that the corresponding $v^{\varepsilon}$ is a smoothing of $v$. Hence, by (2.1), it holds that

$$\partial v(\bar{x}) \subset \operatorname{conv} \left\{ \limsup_{\varepsilon \searrow 0, x \to \bar{x}} \nabla v^{\varepsilon}(x) \right\}. \tag{3.35}$$

Next, as explained in Remark 3.3.1, when $\mu > 0$, condition (3.14) implies (3.15) (because of (3.12)). Then, by Theorem 3.5.1, $\nabla v^{\varepsilon}(x)$ is locally bounded. Hence, by (3.35), so is $\partial v(\bar{x})$.

The conclusion now follows from Proposition 2.1.1. □

Note that in Theorem 3.5.2, taking $\mu > 0$ is useful for providing a locally bounded upper bound for $\partial v(x)$, and the resulting theoretical argument. This is not related to choosing $\mu$ in any computational implementation of the smoothing approach.

## 3.6 Smoothing risk-averse two-stage stochastic programs

We now explain how to cast in our setting a two-stage convex stochastic program, to be considered in our computational experiments in Section 3.7.

Given a risk-aversion parameter $\kappa \in [0,1]$ and a confidence level $\alpha \in (0,1)$, we combine expected value with average-value-at-risk functionals to define

$$\mathscr{R}[Z] := \kappa \mathbb{E}[Z] + (1-\kappa)\mathtt{AVaR}_{\alpha}(Z),$$

for a random variable $Z$ representing a loss ($\kappa = 1$ is the risk-neutral variant). Letting $c(x)$ and $q_s(y)$ denote convex first and second-stage objective functions, the risk-averse two-stage stochastic program of interest is

$$\left\{\begin{array}{ll} \min & c(x) + \mathscr{R}[q_1(y_1), \ldots, q_S(y_S)] \\ \text{s.t.} & x \in X \\ & \text{and, for } s = 1, \ldots, S \\ & y_s \geq 0\,, T_s x + W y_s = h_s\,, \end{array}\right.$$

where we assume once more that the recourse is relatively complete, so that the second-stage problems have nonempty feasible sets. Using the expression

$$\mathtt{AVaR}_{\alpha}[Z] := \min_{x_u \in \mathbb{R}} \left\{ x_u + \frac{1}{1-\alpha} \mathbb{E}[\max(Z - x_u, 0)] \right\}$$

from [RU02], we obtain the following risk-averse version of the two-level problem (3.2):

$$\begin{cases} \min & c(x)+(1-\kappa)x_u+\sum_{s=1}^{S} p_s\mathscr{Q}_s(x,x_u) \\ \text{s.t.} & x\in X\,, x_u\in\mathbb{R}\,, \end{cases} \quad \text{for } \mathscr{Q}_s(x,x_u):=\begin{cases} \min & \kappa q_s(y)+\dfrac{1-\kappa}{1-\alpha}z \\ \text{s.t.} & Wy=h_s-T_s x \\ & q_s(y)-z\le x_u \\ & y\ge 0, z\ge 0 \end{cases} \tag{3.36}$$

By construction, the second-stage objective function and constraints in (3.36) are convex on $(y,z)$, while the recourse function is finite-valued, nonsmooth and convex on $(x,x_u)$. Furthermore, the optimal multipliers of the constraints involving $(x,x_u)$, say $\eta_x,\eta_{x_u}$ with $\eta_{x_u}\ge 0$, provide the subgradient $(T_s^\top\eta_x,-\eta_{x_u})^\top$.

For $s=1,\dots,S$, the smoothed second-stage solutions, denoted $\Big(y^\varepsilon(x,u),z^\varepsilon(x,u)\Big)$, are computed by solving the smoothed second-stage problems

$$\begin{cases} \min & \kappa q_s(y)+\dfrac{1-\kappa}{1-\alpha}z+\varepsilon\phi_s(x_u,y,z) \\ \text{s.t.} & Wy=h_s-T_s x, \end{cases} \quad \text{for } \phi_s(x_u,y,z):=-\sum_{i=1}^{ny}\ln(y_i)-\ln(z)-\ln(z-q_s(y)+x_u)\,. \tag{3.37}$$

The approximate first-stage problem is

$$\begin{cases} \min & c(x)+(1-\kappa)x_u+\sum_{s=1}^{S} p_s\Big(\kappa q_s(y^\varepsilon(x,x_u))+\dfrac{1-\kappa}{1-\alpha}z^\varepsilon(x,x_u)\Big) \\ \text{s.t.} & x\in X\,, x_u\in\mathbb{R}\,, \end{cases} \tag{3.38}$$

which, by the definition of $\mathtt{AVaR}_\alpha$, is not necessarily the same as the objective of the problem below:

$$\begin{cases} \min & c(x)+\mathscr{R}\left[q_1(y_1^\varepsilon(x,x_u)),\dots,q_S(y_S^\varepsilon(x,x_u))\right] \\ \text{s.t.} & x\in X\,. \end{cases}$$

**Corollary 3.6.1** (Specializing the Results to Two-Stage Risk-Averse Stochastic Linear Programs)**.** *Consider the particular instance of the abstract stochastic problem* (3.1) *given by* (3.36) *and its smooth approximation* (3.38). *Suppose that the matrix $W$ has linearly independent rows. Assume also that for all $x\in X$ the recourse problems, without risk measures, satisfy the Slater condition and have nonempty solution sets. Then the following holds when building the smoothing with $\mu=0$ as in* (3.37)*:*

*(i) For $s=1,\dots,S$,*

$$Q_s(x,x_u)\le\kappa q_s(y^\varepsilon(x,x_u))+\frac{1-\kappa}{1-\alpha}z^\varepsilon(x,x_u)\le Q_s(x,x_u)+\varepsilon C_s$$

*for an explicit and known constant $C_s>0$.*

*(ii) The objective function of* (3.38) *decreases monotonically and uniformly to the objective function of* (3.36) *as $\varepsilon\searrow 0$.*

*(iii) If $\bar{x}^\varepsilon$ is a global solution to* (3.38) *then $\bar{x}^\varepsilon$ is an approximate global solution to* (3.36) *with explicit and known quality of approximation.*

*Proof.* To prove item (iii), look at item (i). Start multiplying (i) by $p_s$, and then summing across the scenarios. After that, add the first-stage cost $c(x)+(1-\kappa)x_u$ and take the infimum on the resulting inequality over $x\in X$. □

From item (iii) of Corollary 3.6.1, we know that every accumulation point of $\bar{x}^\varepsilon$ is a global solution of (3.36). In practice, the result applies because it is possible to compute $\bar{x}^\varepsilon$ as global solutions and, for this setting, smoothing preserves the original convexity of the problem. In general, for non-convex value functions, item (iii) is still true, but one may not be sure of global optimality in computation. Whenever gradient consistency holds, limits of $\bar{x}^\varepsilon$ are stationary points of the original problem. In particular, the local boundedness of the smoothed gradients proved in this chapter ensures that the singular subdifferential of the value function and the singular closure of the smoothed gradients agree, which is the gradient consistency result.

## 3.7 Numerical experiments

We now benchmark our proposal against the state-of-the-art bundle solver [Fra02] in terms of decrease in the objective function values along the iterations, using data profiles [MW09].

The experiments were performed on an Intel Core i7 computer with 1.9 GHz, 8 cores and 15.5 GB RAM, running under Ubuntu 18.04.3 LTS.

### 3.7.1 Instances and solvers considered in the benchmark

The test set was created by using four functions from I. Deák's collection [Deá06], having $nx = 20$ first-stage variables and $ny = 30$ second-stage variables per scenario. For each scenario, $l = 20$ affine equality constraints couple the two stages and there are 10 affine equality constraints defining the fist-stage feasible set $X$. All the assumptions in Corollary 3.6.1 are satisfied.

In order to define new, more challenging, instances, the stochastic linear programs from [Deá06] were modified by adding a quadratic term to the linear second-stage cost. Accordingly, given $\mathrm{q}_s$, the linear cost in the original problems and a scalar parameter $r \geq 0$, in (3.36) we set

$$q_s(y) := \mathrm{q}_s^\top y + \frac{1}{2} r y^\top y.$$

The instances in the benchmark are obtained by varying the number of scenarios and the quadratic parameter

$$S \in \{1, 2, \dots, 20\} \quad \text{and} \quad r \in \{0, 0.01, 0.1, 1\}.$$

In (3.36) the risk-aversion parameter is $\kappa \in \{0.5, 1\}$, and the confidence level is set to $\alpha = 0.9$, noting that the risk-neutral version ($\kappa = 1$) has no variables $x_u, z$ and related constraints. Accordingly, the considered second-stage problems are increasingly more difficult, being linear programs if $r = 0$ and $\kappa = 1$, quadratic programs if $r > 0$ and $\kappa = 1$, and problems with quadratic objective and quadratic constraints (QCQP) if $r > 0$ and $\kappa \in [0, 1)$. We used CPLEX 12.8 and an optimized build of Ipopt 3.12.10 with the linear solver Pardiso as described in the manual; see also [WB05]. Both packages were configured to employ only one thread per run.

To solve the corresponding problems (3.36), we consider two methodologies, listed below.

– BM, a decomposition method for the first-stage problem, based on the bundle algorithm by A. Frangioni, [Fra02], one of the best solvers in the area. The method parameters were tuned for best performance, particularly regarding the management of the bundle size (keeping only active bundle elements). At each iteration, say $(x^k, x_u^k)$, the algorithm uses certain oracle information, obtained by evaluating the nondifferentiable convex objective function

$$c(x^k) + (1-\kappa) x_u^k + \sum_{s=1}^{S} p_s \mathscr{Q}_s(x^k, x_u^k).$$

In addition to this value, the bundle method uses a subgradient of the form

$$\Big( \nabla c(x^k) + \sum_{s=1}^{S} p_s T_s^\top \eta_{x^k}, (1-\kappa) - \sum_{s=1}^{S} p_s \eta_{x_u^k} \Big)^\top,$$

for multipliers $(\eta_{x^k}, \eta_{x_u^k})$ obtained when computing the value of the recourse function $\mathscr{Q}_s(x^k, x_u^k)$, for each scenario $s$. Depending on the instance, computing such value amounts to dealing with a linear program, a quadratic program, or a QCQP problem, solved with the packages CPLEX or Ipopt. As CPLEX currently does not provide directly multipliers for quadratic constraints, we could not use it for the risk-averse quadratic runs.

– ST, our smoothing with log-barrier and Tikhonov regularization approach, solving the approximate first-stage master problem (3.37) with Ipopt. In this setting, the oracle information for the smoothed objective function

$$c(x^k) + (1-\kappa) x_u^k + \sum_{s=1}^{S} p_s \Big( \kappa q_s(y^\varepsilon(x^k, x_u^k)) + \frac{1-\kappa}{1-\alpha} z^\varepsilon(x^k, x_u^k) \Big)$$

requires the solution of one problem (3.37) written with $(x, x_u) = (x^k, x_u^k)$ per scenario $s$. For all the considered instances, this is a problem with nonlinear objective function and affine constraints solved with Ipopt, giving the objective function gradient as callback information, computing its value according to Theorem 3.3.1(ii). The performance reported below relies heavily on the availability of an optimized build of Ipopt. In particular, the regularized solution mappings in Theorem 3.3.1(i), $y^\varepsilon(x)$ and $\lambda^\varepsilon(x)$, are an output of Ipopt, once certain *mu-target* option is activated (such Ipopt parameter corresponds to $\varepsilon$). For simplicity,

$\varepsilon$ was kept constant along iterations. However, note that the bounds given in item (i) in Corollary 3.6.1 justify interpreting this parameter as a direct measure of precision when $\mu = 0$. Recall that when $\mu > 0$, as shown in Proposition 3.3.1 and further discussed in Remark 3.3.1, the determination of the quality of the smoothing depends on bounds for the scenario subproblem solutions, a knowledge that is hardly available in practice. For numerically hard problems taking $\mu > 0$ can be advantageous to improve the chances that derivatives of the regularized solution mappings are sufficiently precise. As in (3.8), in this case the Tikhonov term involves a factor $0.5\varepsilon\mu$ that is kept constant along iterations. The range chosen for these parameters is

$$\varepsilon \in \{0.01, 0.1, 1\} \quad \text{and} \quad \mu \in \{0, 0.1, 1\}.$$

As we deal with random instances, each experiment is repeated three times, yielding 540 or 2160 different runs, respectively if $r = 0$ and $r > 0$. Table 3.1 summarizes all the variants considered in the benchmark.

| Problem type | (in (3.36)) | BM-CPLEX | BM-Ipopt | ST-Ipopt |
|---|---|---|---|---|
| Risk-neutral linear | $(\kappa = 1, r = 0)$ | x | x | x |
| Risk-neutral quadratic | $(\kappa = 1, r > 0)$ | x | x | x |
| Risk-averse linear | $(\kappa \in [0,1), r = 0)$ | – | x | x |
| Risk-averse quadratic | $(\kappa \in [0,1), r > 0)$ | – | x | x |

Table 3.1: Benchmark configuration.

### 3.7.2 Comparing the solvers with data profiles

To report the results of the experiments we use data profiles as introduced in [MW09]. Specifically, for a given instance, the maximum running time of a given set of methods is used to normalize all the running times, so that in the graph abscissa the range for all methods is between 0 and 1 (the value of 1 can be thought of as the maximum time budget given to the solvers). The ordinate in the data profiles corresponds to the probability of each method delivering the best iterate plus a gap until a time given in the abscissa. The gap corresponds to 5% of the largest decrease obtained for a given instance by all methods, that were given the same starting point.

The results are analyzed by considering the different groups in Table 3.1, starting with the risk-neutral instances ($\kappa = 1$), in both its linear ($r = 0$) and quadratic ($r > 0$) variants. The corresponding profiles are given in Figure 3.1.

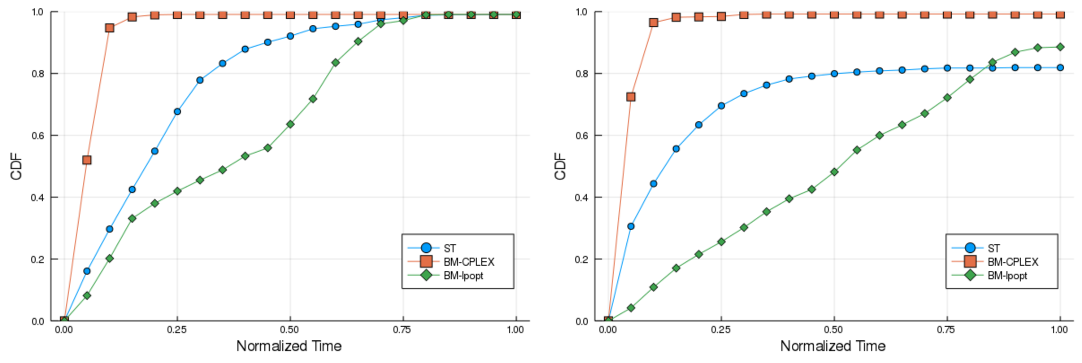

Figure 3.1: Performance for linear (left) and quadratic (right) instances without risk.

In both graphs BM-CPLEX is a clear winner, followed by ST and with BM-Ipopt performing worst. For the linear group, in 70% of the runs ST obtained the largest functional decrease using a slightly more than a quarter of the time budget: on the left graph the abscissa 0.25 has ordinate 0.7 for ST (the dot). All the solvers succeeded in solving all of the linear instances (the ordinate value of 1 is attained by the three lines). By contrast, the quadratic instances clearly put Ipopt in trouble, as both ST and BM-Ipopt failed in about 20% of the runs. Notice that for this simplest test set (no risk) there is a big difference in the performance of BM-CPLEX and BM-Ipopt. This illustrates well the impact that subproblem solution times can have on a decomposition method. Considering that BM subproblems are all linear or quadratic programs for these groups of instances, the profiles can be seen as a cautionary tale on the importance of using a specific solver (CPLEX) rather than a general purpose one (Ipopt) whenever possible. Incidentally, this behavior also indicates that the difference of performance between ST and BM-CPLEX might be explained by the time each solver spent in the respective subproblems (nonlinear for ST).

The next profiles in Figure 3.2, potentially more challenging in terms of subproblem solution, consider risk aversion ($\kappa \in [0,1)$), again with linear ($r = 0$) instances on the left and quadratic ones ($r > 0$) on the right.

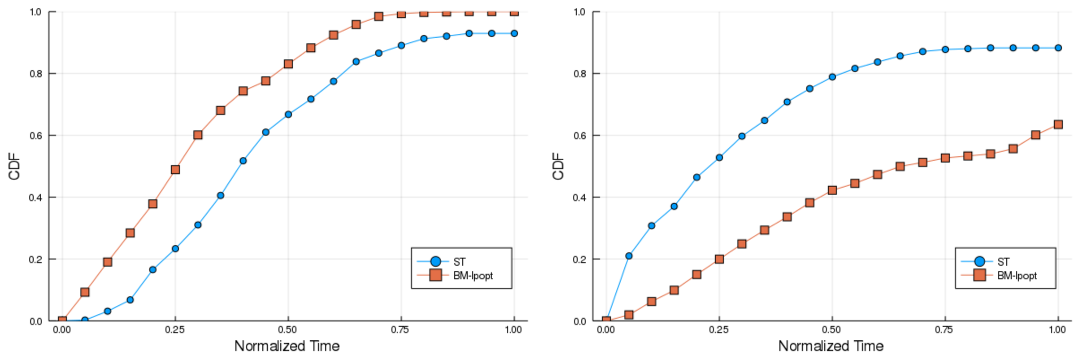

Figure 3.2: Performance for linear (left) and quadratic (right) instances with risk.

The left graph gives BM-Ipopt as a winner, followed closely by ST. The situation is reversed for the risk-averse quadratic set of instances. Specifically, on the right graph ST performance is far superior than BM-Ipopt's (recall that for these instances the comparison with BM-CPLEX is not possible). The fact that these are the hardest problems is evident in the profile on the right, showing a percentage of failures of 10% and 40%, for ST and BM-Ipopt, respectively.

In our final profiles we confirm the impact in terms of solution times of introducing a quadratic term in the second-stage subproblems, particularly when there is risk aversion. The top profiles in Figure 3.3 show that, when $r$ varies in $\{0.01, 0.1, 1.0\}$ both BM-CPLEX and ST (left and right top graphs) perform alike for the instances without risk aversion. The situation is substantially different for the bottom profiles, with the performance of BM-Ipopt and ST for the same three values of $r$, now considering risk. On the left bottom graph, as $r$ gets smaller, the improvement in BM-Ipopt's performance is noticeable, as well as a reduction in the percentage of failures: about 30% for $r = 0.01$ and $r = 0.1$, and 60% for $r = 1.0$. For both BM and ST, smaller values of $r$ make the problem solution easier. The right bottom graph, with ST runs, has much fewer failures than BM's, and, more remarkably, the three ST lines look alike for the three different values of $r$.

For the considered test set, it appears that BM-CPLEX should be preferred for the linear instances, while ST is the winner for problems with risk aversion and quadratic objective function in the second stage. Regarding ST's failures, by solving the deterministic equivalent with CPLEX, we could check that ST had found good estimates of the optimal value without reaching the threshold of 5% error in some instances. We expect that a dynamical management of $\varepsilon$ would help in eliminating such failures. This is a topic of future research.

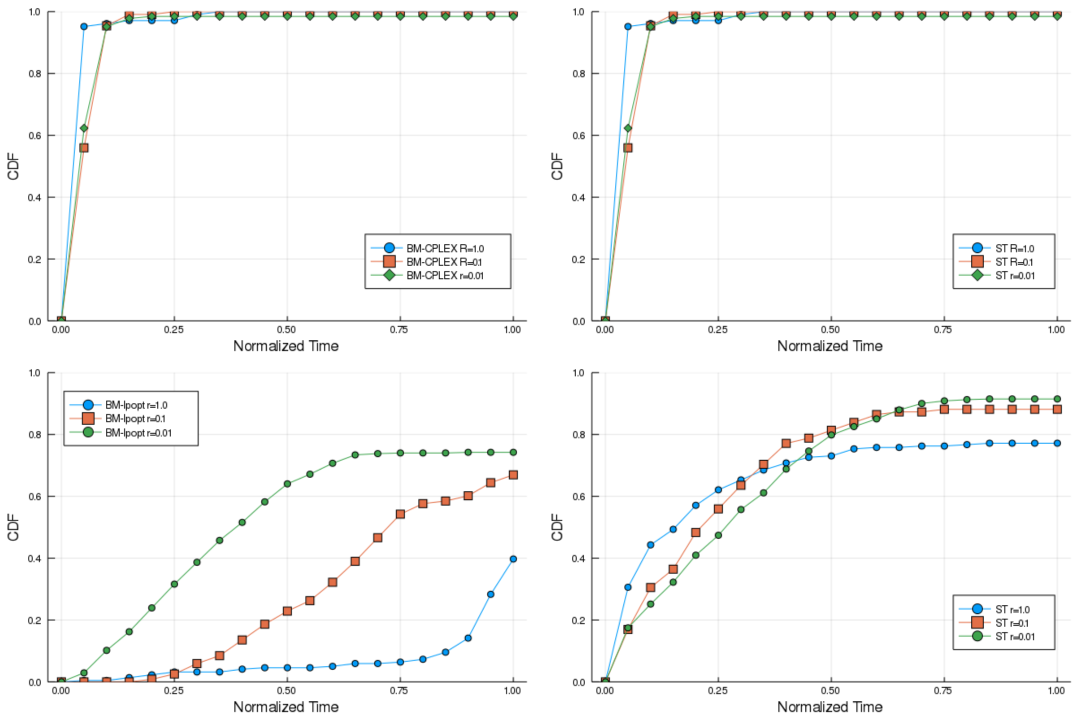

Figure 3.3: Effect of the quadratic term on BM (left) and ST (right) without (top) and with (bottom) risk.

## 3.8 Conclusions

The theory of sample average approximations (SAA) focuses on how problems with finitely many scenarios approximate a two-stage stochastic optimization problem in general probability spaces [XY10a; XY10b; PA20]. Having this goal in mind, in [XY10a] the authors consider a setting akin to our master problem (3.36), with additional equilibrium constraints, and in a general probability space. With respect to our analysis, the convergence theory in [XY10a] requires the computation of stationary points for the general non-convex version of (3.36). Instead, we rely on epigraphical convergence, a weaker form of convergence recently investigated in the setting of general probability spaces in [PA20]. In this last work, the almost sure constraints in [XY10b] are employed to study certain reformulations of two-stage stochastic equilibrium problems with complementarity conditions.

# Chapter 4

# Decomposition Algorithms for Some Deterministic and Two-Stage Stochastic Single-Leader Multi-Follower Games

We consider a certain class of hierarchical decision problems that can be viewed as single-leader multi-follower games, and be represented by a virtual market coordinator trying to set a price system for traded goods, according to some criterion that balances supply and demand. The objective function of the market coordinator involves the decisions of many agents, which are taken independently by solving convex optimization problems that depend on the price configuration and on realizations of future states of the economy. One traditional way of solving this problem is via a mixed complementarity formulation. However, this approach can become impractical when the number of agents and/or scenarios becomes large. This chapter concerns agent-wise and scenario-wise decomposition algorithms to solve the equilibrium problems in question, assuming that the solutions of the agents' problems are unique, which is natural in many applications (when solutions are not unique, the approximating problems are still well-defined, but the convergence properties of the algorithm are not established). The algorithm is based on the content of the previous chapter, where a suitable regularization of solution mappings of fully parameterized convex problems is developed. Here, we show one specific strategy to manage the regularization parameter, extend some theoretical results to the current setting, and prove that the smooth approximations of the market coordinator's problem converge epigraphically to the original problem. Numerical experiments and some comparisons with the complementarity solver PATH are shown for the two-stage stochastic Walrasian equilibrium problem.

## 4.1 Introduction and motivation

The so-called setting of *multiple optimization problems with equilibrium constraints* (MOPEC) serves as a broad framework to encompass various types of hierarchical problems that arise often in applications, most notably in energy optimization; see [PFW16], [Sag12], and references therein. Here, we consider a certain class of MOPECs that can be viewed as single-leader multi-follower games.

We are interested in problems where, for a given parameter $p \in \mathbb{R}^q$, agents $a$ in a set $\mathscr{A}$ determine their decisions $x_{\mathscr{A}}(p) = (x_a(p) \in \mathbb{R}^{n_a}, a \in \mathscr{A})$ by solving independently convex optimization problems of the form

$$x_a(p) = \arg\min_x \{f_a(x,p) : B_a(p)x = b_a(p), \quad g_a(x,p) \leq 0\}. \tag{4.1}$$

The convex objective function $f_a$ and the affine equality and convex inequality constraints are such that in (4.1) the minimizer $x_a(p)$ is unique. The goal is to find the optimal parameter $p^*$, a price signal that is observed when coupling all the agents' decisions, by minimizing a criterion $F : \mathbb{R}^N \to \mathbb{R}$ where $N = \sum_a n_a$ over a set $\Pi$:

$$p^* \in \operatorname{Arg}\min_p \{F(x_{\mathscr{A}}(p)) : p \in \Pi\}. \tag{4.2}$$

The notation Arg min in (4.2) refers to a set, while arg min in (4.1) is a singleton.

Finding the optimal price $p^*$ and decisions $x_{\mathscr{A}}(p^*)$ that solve (4.1)-(4.2) gives a particular instance of a MOPEC. We do not address the MOPEC setting in all of its generality, in which the agents behave strategically, taking into account the other agents' decisions in their individual optimization problems, as in a generalized Nash game. But the solution approach presented below, based on a special smoothing of the solution mappings of the agents' problems, should still be applicable for a general MOPEC. To simplify the presentation, we focus on the specific framework (4.1)-(4.2), suitable for the applications we have in mind. One example is the stochastic

Walrasian Equilibrium Problem (WEP); see [DJW17; JJBW02].

As it occurs often in hierarchical optimization, in the WEP the agents' decisions are only defined for parameters in the set $\Pi$. This is because for such problems the feasible set is given by certain *price system*, specific to the considered economy. The willingness of the agents to trade the goods available in the economy is cost-minimizing, based on the price of the goods and possibly subject to budget constraints. Agents' decisions are taken only for positive prices, and the solution set in (4.1) is empty if $p \notin \Pi$. In order to determine the equilibrium price, in problem (4.2) a virtual market coordinator minimizes the mismatch between supply and demand.

Even in the setting of (4.1), less general than MOPEC, the global problem (4.2) can be nonsmooth and nonconvex, and computing a local minimizer is a difficult task. Our proposal considers the agent's problem (4.1) as a convex smooth program parameterized by the price, and applies the smoothing and regularization procedure of [BSS20] to the possibly nonsmooth solution mappings of (4.1). The idea, simple to explain (but not so simple to analyze theoretically), boils down to replacing in (4.2) the agents' decisions $x_a(p)$ (which may fail to have classical derivatives at all points) with approximating functions that are smooth. Accordingly, for the given smoothing parameter $\varepsilon > 0$ and regularization parameter $\mu = \mu(\varepsilon) \geq 0$, we build functions $x_a^\varepsilon(p) = x_a^{\varepsilon,\mu(\varepsilon)}(p)$ that are well-defined at least on $\Pi$, are smooth on a certain related set $\tilde{\Pi}$, and converge in the following sense:

$$\lim_{\varepsilon \searrow 0, p' \in \Pi, p' \to p} x_a^\varepsilon(p') = x_a(p) \quad \text{for all } p \in \Pi \text{ and } a \in \mathscr{A}. \tag{4.3}$$

The regularization parameter $\mu = \mu(\varepsilon) \geq 0$ is, in general, a function of the smoothing parameter $\varepsilon > 0$, bounded around zero. There are three important cases: $\mu = 0$, $\mu > 0$ but fixed, and variable $\mu > 0$ tending to zero as $\varepsilon \searrow 0$ (e.g., $\mu = \sqrt{\varepsilon}$). The crucial property (4.3) connects our approach with some classical concepts from variational analysis. Epigraphical convergence [RW09, Chapter 7] implies approximation properties for solutions of problem (4.2). In particular, when (4.3) holds, the approximating functions define a smoothing, in the sense of [Che12].

An important feature of our approach is that the derivatives of the approximating functions can be computed numerically. This is important to efficiently solve the approximations of the global problem (4.2). For an appropriate sequence of parameters $(\varepsilon_k \searrow 0, \mu_k = \mu(\varepsilon_k) \geq 0)$ and smooth functions $x_{\mathscr{A}}^{\varepsilon_k}(p) = x_{\mathscr{A}}^{\varepsilon_k,\mu_k}(p)$, our method computes a sequence $p_k$ of approximate local solutions for the smooth problems

$$\min_p \{F(x_{\mathscr{A}}^{\varepsilon_k}(p)) : p \in \Pi \cap \tilde{\Pi}\}. \tag{4.4}$$

Depending on how the approximating functions are built, the smoothness set $\tilde{\Pi}$ can be larger or smaller than $\Pi$. Having access to the derivatives of $x_{\mathscr{A}}^{\varepsilon_k}(p)$, first-order information for the objective function in (4.4) is available and a stationary point can be computed, for example using Ipopt [WB05]. In our numerical experiments in Section 4.5.2 we identified two issues that impact the performance of the method. First, a low accuracy in the output of the smoothed problems results in low quality derivatives for the smooth mappings $x_{\mathscr{A}}^{\varepsilon_k}(p)$ and this sometimes hinders the solution process. Second, the parameter $\varepsilon_k > 0$ defining the smoothing needs to be carefully chosen, so that the objectives of (4.2) and (4.4) are sufficiently close across iterations.

When compared with the method in [BSS20] for nonconvex two-stage stochastic programming problems, the most important conceptual difference of the current proposal is the following. In [BSS20], there is a master problem akin to (4.2) and subproblems like (4.1). The smoothing [BSS20] inserts in the master problem the *optimal value function* of the smoothed subproblems. By contrast, in (4.4), we rather use the smoothed *solution mappings*. This makes the situation much different for the convergence analysis of (4.2), because approximation properties for solution mappings are weaker than those available for optimal value functions. There are also important differences for computational implementations. For example, a single value for the smoothing parameter $\varepsilon > 0$ (sufficiently small), was often sufficient in the computational experience of [BSS20]. By contrast, for problem (4.2) a proper management of the sequence of the values of $\varepsilon_k$ becomes crucial.

Our approach is particularly well suited for decomposition. When solving (4.4), an agent-wise decomposition is readily available because the approximating functions are defined independently across agents. Furthermore, when the agents' problems (4.1) are two-stage stochastic programs, our construction allows for a decomposition method across both agents and scenarios. Decomposition methods for variational inequalities with Dantzig-Wolfe and Benders-type structure were developed in [LSS13], [LSS12], [LSS16]; extending [FC05], [CF10], [GF10]. For generalized Nash games, we refer to [FPS11] and [KS12]; and for sparse affine variational inequalities, see [KHF17] and [KF19]. Other agent-wise decomposition methods with applications in communications engineering are given in [ASP14; Scu+13; Scu+11].

Regarding the direct solution of a problem like (4.1)-(4.2), there are at least four major classes of methods. These are the complementarity-based or the variational inequality algorithms using PATH [FM99], the augmented

Lagrangian methods [And+08], [Sch12], [KS16], the more recent derivative-free approach in [DJW17], and the smoothing techniques [XY13], [XWY14], [XYZ14], [XYZ15], relying on a smoothing given by an integral in a multidimensional space. Although not clear how to implement them efficiently in practice, integral-based smoothings are gradient consistent in the sense of [Che12], [BHK13]. This property guarantees that limits of stationary points of the smoothed problems are stationary points of the original problem (with stationarity being understood in the generalized sense of nonsmooth analysis). We do not show that our method is gradient consistent, which is the reason we focus only on the epigraphical convergence of the approximating problems. However, the property of gradient consistency is not a necessary condition for the method to work well in practice, as it is also clear from the theory.

The rest of the chapter is organized as follows. In Section 4.2, we give some preliminary results and notation. In Section 4.3, we adapt and extend some properties of the smoothing in [BSS20] to the current equilibrium setting. In Section 4.3.3, we describe our agent-wise decomposition method. Section 4.4 discusses the same two-stage WEP found in [DJW17]. Numerical experiments for the decomposition across agents are reported in Section 4.5.2. The agent-wise and scenario-wise decomposition method is shown in 4.5.3. Concluding remarks and comments are given in Section 4.6.

## 4.2 Background material

Notation-wise, we mostly follow [RW09], with $\overline{\mathbb{R}}$ being the extended real numbers, $B(u,\delta)$ the ball around $u$ of radius $\delta > 0$, and with all norms being Euclidean (the respective spaces are always clear from the context). The symbol $o(t)$ denotes any expression such that $t^{-1}o(t) \to 0$ whenever $t \searrow 0$. The symbol conv $D$ stands for the convex hull of the set $D$.

### 4.2.1 Properties of smoothing functions and epigraphical convergence

Let $v : V \to \mathbb{R}$ be continuous on the open set $V \subset \mathbb{R}^q$, and assume that for $\varepsilon > 0$ we are given smooth functions $v^\varepsilon : V \to \mathbb{R}$ such that, as in Section 2.3, their lower semicontinuous closure satisfies the identity

$$\operatorname{lsc} v^\varepsilon(p) = v(p) \text{ for all } p \in V\,. \tag{4.5}$$

For more details about smoothing functions, see Section 2.3, [BHK13] and [BH16]. The next result, that holds independently of the property of gradient consistency, shows that "unexpected" things may happen only when the Lipschitz constants of the smoothing gradients diverge. The result gives an indication that the parameter $\varepsilon > 0$ should be managed carefully. Notice also that, at points of nonsmoothness of the function, the Lipschitz constants of the gradient of the smoothings indeed "explode".

**Proposition 4.2.1.** *Let $v$ and $v^\varepsilon$ be given as above. Fix $p \in V$. Assume that there exists a constant $L > 0$, called a uniform Lipschitz constant for the gradients of $v^\varepsilon$ at $p$, and there exists $\delta > 0$, such that*

$$\|\nabla v^\varepsilon(p_1) - \nabla v^\varepsilon(p_2)\| \le L\|p_1 - p_2\| \quad \textit{for all} \quad p_1, p_2 \in B(p,\delta) \quad \textit{and} \quad \varepsilon \in (0,\delta).$$

*Then, $\nabla v(p)$ exists at $p \in V$ and*

$$\lim_{\varepsilon \searrow 0, p' \to p} \nabla v^\varepsilon(p') = \nabla v(p).$$

*In particular, if $v$ is not differentiable at $p$, there is no uniform Lipschitz constant for the gradients of $v^\varepsilon$ at $p$, independently of how the smoothing sequence $v^\varepsilon$ is constructed.*

*Proof.* Using the uniform Lipschitz constant, the Newton-Leibniz formula (e.g., [IS14, Lemma A.11]) implies, for $t \in (-\frac{\delta}{2}, \frac{\delta}{2})$ and $p'$ close to $p$ and $d \in \mathbb{R}^q$ such that $\|d\| \le 1$, that

$$|v^\varepsilon(p' + td) - v^\varepsilon(p') - t\nabla v^\varepsilon(p')^\top d| \le \frac{Lt^2}{2} \quad \text{for all}\, \varepsilon \in (0,\delta). \tag{4.6}$$

Also, because there is a local uniform Lipschitz constant, the smoothing gradients $\nabla v^\varepsilon(p')$ are locally bounded for small $\varepsilon > 0$ and $p'$ close to $p$. This implies that there is $\bar{v} \in \limsup_{\varepsilon \searrow 0, p' \to p} \nabla v^\varepsilon(p')$. Taking limits on (4.6) when $\varepsilon \searrow 0$, $p' \to p$ and using (4.5), we conclude that all partial derivatives of $v$ exist at $p$ and that $\bar{v}$ is unique and it is the gradient vector of $v$ at $p$. □

## 4.3 Decomposition method induced by our smoothing technique

The approach outlined in this section was introduced in [BSS20] as a tool to smooth, approximate, and regularize value functions of fully parameterized convex optimization problems. Here, we revisit/adapt those results under the light of smoothed solution mappings (rather than the value function), considering a prototypical problem, dropping the subindex $a$ in (4.1) and the assumption about uniqueness of solutions. Thanks to the continuity and differentiability properties of the smoothing, the decomposition method presented below generates an epigraphically convergent sequence, as in Theorem 2.2.1.

### 4.3.1 Defining the smoothed problems

The conditions necessary for the theory in [BSS20] are gathered below; for their precise role and use see [BSS20].

**Assumption 4.3.1.** In the prototypical problem

$$\mathscr{X}^*(p) := \operatorname{Arg}\min_x \{f(x,p) : B(p)x = b(p), \quad g_i(x,p) \le 0 \text{ for } i = 1,\dots,m\}, \tag{4.7}$$

the functions $f(\cdot,p)$ and $g_i(\cdot,p)$, $i = 1,\dots,m$, are convex for all $p \in \mathbb{R}^q$, and all the data is sufficiently smooth both in $x$ and in $p$. Also, the rows of the matrices $B(p)$ are linearly independent for all $p$ and the Slater condition holds for all $p$: for each $p$ there exists $\hat{x}(p)$ such that $B(p)\hat{x}(p) = b(p)$ and $g_i(\hat{x}(p),p) < 0$ for $i = 1,\dots,m$.

Since problem (4.1) is not assumed to have a solution for all $p$, we may have dom $\mathscr{X}^* \subsetneq \mathbb{R}^q$. Note that while we allow the solution set $\mathscr{X}^*(p)$ to be empty, the feasible set of problem (4.7) cannot be empty.

In what follows, we adopt the convention that $\log\alpha = -\infty$ if $\alpha \le 0$. Given the parameters $\varepsilon > 0$ and $\mu \ge 0$, a parametric Tikhonov-regularized log-barrier penalty function is used to approximate problem (4.7) by

$$x^{\varepsilon,\mu}(p) := \arg\min_x \{f(x,p) + \varepsilon\phi^\mu(x,p) : B(p)x = b(p)\}, \tag{4.8}$$

where

$$\phi^\mu(x,p) := -\sum_{i=1}^m \log\{-g_i(x,p)\} + \frac{\mu}{2}\|x\|^2.$$

The regularized solution $x^{\varepsilon,\mu}(p)$ is written in lower case, because it is unique when it exists (under our assumptions). If $\mu = 0$, the solution $x^{\varepsilon,\mu}(p)$ exists if $\mathscr{X}^*(p)$ is bounded [DS99], [MZ98]. The solution always exists if $\mu > 0$ (even when $\mathscr{X}^*(p) = \emptyset$), because in that case the objective function of (4.8) is strongly convex and problem (4.8) is assumed to have a nonempty feasible set.

**Theorem 4.3.2** (Proposition 2 in [BSS20])**.** *Let $v(p)$ and $v^{\varepsilon,\mu}(p)$ be the optimal value functions of problems* (4.7) *and* (4.8)*, respectively. Under Assumption 4.3.1, for all $p \in D :=$ dom $\mathscr{X}^* \cap$ dom $x^{\varepsilon,\mu}$ it holds that:*

$$v(p) \le v^{\varepsilon,\mu}(p) \le v(p) + m\varepsilon + \frac{\varepsilon\mu}{2}\min_{x\in\mathscr{X}^*(p)}\|x\|^2, \tag{4.9}$$

*and*

$$\frac{\mu}{2}\min_{x\in\mathscr{X}^*(p)}\|x\|^2 + m \ge \frac{\mu}{2}\|x^{\varepsilon,\mu}(p)\|^2. \tag{4.10}$$

**Remark 4.3.1.** The inequality $v(p) \le v^{\varepsilon,\mu}(p)$ always holds, even if $x^{\varepsilon,\mu}(p)$ is not well-defined or $\mathscr{X}^*(p) = \emptyset$. If $x^{\varepsilon,\mu}(p)$ is not well-defined, then $v^{\varepsilon,\mu}(p)$ is understood as an infimum. It should also be noted that the domain $D$ is independent of both $\varepsilon > 0$ and $\mu \ge 0$.

We are naturally interested in conditions under which the lower semicontinuous closure of the smoothing coincides with $v$, i.e., (4.5) holds. Theorem 4.3.2 is instrumental to identify a path to follow. We would first have to guarantee boundedness of the term $\min_{x\in\mathscr{X}^*(p)}\|x\|^2$. Second, we would need to show that $v$ is continuous relative to subsets of dom $\mathscr{X}^*$. Then, condition (4.5) would follow from taking limits in (4.9).

The first requirement depends on the following weak assumption, called restricted inf-compactness condition in [Guo+14] (later rediscovered independently in [BSS20], through an unrelated computationally oriented approach):

$$\limsup_{p'\in\text{dom }\mathscr{X}^*, p'\to p} \left\{\min_{x\in\mathscr{X}^*(p')}\|x\|^2\right\} < +\infty \quad \text{for all } p \in \text{dom } \mathscr{X}^*. \tag{4.11}$$

Condition (4.11) means that the solution mapping $\mathscr{X}^*$ has a locally bounded selection over its domain. Note that (4.11) holds automatically if the feasible sets in (4.7) are uniformly bounded for $p \in \operatorname{dom} \mathscr{X}^*$ (which by itself is a rather natural assumption, holding in many cases of interest).

When the regularization parameter $\mu \geq 0$ is taken as a function of the smoothing parameter $\varepsilon$, we shall use the computational version of (4.11), given by

$$\limsup_{\varepsilon \searrow 0, p' \in D, p' \to p} \|x^{\varepsilon,\mu(\varepsilon)}(p')\| < +\infty \quad \text{for all} \quad p \in D := \operatorname{dom} \mathscr{X}^* \cap \operatorname{dom} x^{\varepsilon,\mu(\varepsilon)}. \tag{4.12}$$

Note that assumption (4.12) does not need $\varepsilon\mu(\varepsilon) \to 0$ to make sense. It refers only to boundedness, not to convergence of the regularized solutions to the actual solution.

Regarding the second issue above, related to continuity of the value functions, the following simple example shows the behaviour of the lower semicontinuous closure of the smoothing for parameters outside of $\operatorname{dom} \mathscr{X}^*$.

**Example 4.3.3.** Consider the problem

$$\mathscr{X}^*(p) = \operatorname*{Arg\,min}_x \{px : x \geq 0\} = \begin{cases} \emptyset & \text{if } p < 0, \\ \{x : x \geq 0\} & \text{if } p = 0, \\ \{0\} & \text{if } p > 0. \end{cases}$$

The value function $v(p) = \inf_x \{px : x \geq 0\}$ fails to be lower semicontinuous at $p = 0$. Let us consider $\mu > 0$ fixed. Regarding the smoothing, it holds that

$$\operatorname{lsc} v^{\varepsilon,\mu}(p) = \begin{cases} -\infty & \text{if} \quad p \leq 0, \\ 0 & \text{if} \quad p > 0. \end{cases}$$

As it is easy to see, if $\mu > 0$, for all problems satisfying Assumption 4.3.1 we have that

$$\operatorname{lsc} v^{\varepsilon,\mu}(p) = -\infty \quad \text{for all} \quad p \in \operatorname{int} (\mathbb{R}^q \setminus \operatorname{dom} \mathscr{X}^*).$$

The lower semicontinuous closure lsc $v$ of the value function coincides with $v$ at all points except at $p = 0$, where lsc $v(0) = -\infty$.

Since under Assumption 4.3.1 the value function $v$ can fail to be lower semicontinuous, it may not coincide with its lower semicontinous closure. Analyzing how smoothing behaves for the example above led us to the following interesting result, that reveals the role of our smoothing as a tool to change a problem like $\min_{p \in P} v(p)$, which can fail to have a solution, to a sequence of problems $\min_{p \in P} v^{\varepsilon,\mu}(p)$, all having solutions.

**Theorem 4.3.4** (On lower semicontinuous closures)**.** *Under Assumption 4.3.1 and* (4.11)*, for $\mu > 0$ fixed, it holds that*

$$\operatorname{lsc} v^{\varepsilon,\mu}(p) = \operatorname{lsc} v(p) := \liminf_{p' \to p} v(p') \quad \textit{for all} \quad p \in \mathbb{R}^q.$$

*Proof.* If $p \in \operatorname{int} \operatorname{dom} \mathscr{X}^*$ or $p \in \operatorname{int} (\mathbb{R}^q \setminus \operatorname{dom} \mathscr{X}^*)$, the statement is trivial because the following stronger property holds:

$$\lim_{\varepsilon \searrow 0, p' \to p} v^{\varepsilon,\mu}(p') = v(p) = \operatorname{lsc} v(p).$$

Take $p$ on the boundary of $\operatorname{dom} \mathscr{X}^*$. Note that lsc $v(p) = -\infty$. Take sequences $\varepsilon_k \searrow 0$ and $p_k \to p$. We trivially have that $\liminf_k v^{\varepsilon_k,\mu}(p_k) \geq -\infty = \operatorname{lsc} v(p)$. Now note that there is a sequence $p_k \to p$ such that $v(p_k) = -\infty$. Recall that for all fixed $p \in \mathbb{R}^q$ we have $v^{\varepsilon,\mu}(p) \to v(p)$ when $\varepsilon \searrow 0$. Therefore, for each $k$ we can find $\varepsilon_k > 0$ such that $v^{\varepsilon_k,\mu}(p) < -k$. We just proved that there are sequences $\varepsilon_k \searrow 0$ and $p_k \to p$ such that $\limsup_k v^{\varepsilon_k,\mu}(p_k) = -\infty \leq v(p)$. $\square$

Our next two examples illustrate the fact that non-convex value functions may not be continuous on the interior of their domain.

**Example 4.3.5.** Take $v(p) = \min_x \{px : x \geq 0\}$. Note that $v(p) = 0$ if $p \geq 0$ and $v(p) = -\infty$ if $p < 0$ and that $-v$ is a convex function. Now note that the effective domain of $v$ is $\mathbb{R}$, while the one of $-v$ is $\{p : p \geq 0\}$. The function $-v$ is continuous on the interior of its domain, because it is convex. But the example shows that the same statement is false for a concave function. However, looking at the solution mapping $\mathscr{X}^*(p) = \operatorname{Arg\,min}_x \{px : x \geq 0\}$ we can realize that both $v$ and $-v$ are continuous relative to the domain of the solution mapping, which is $\{p : p \geq 0\}$. In general, we are only able to prove Lipschitz continuity locally, on $\operatorname{int} \operatorname{dom} \mathscr{X}^*$, and continuity relative to compact subsets of $\operatorname{dom} \mathscr{X}^*$.

**Example 4.3.6.** Consider the problem $\min\{x : x \geq 0\}$. Let us look at the dual function $\psi(p)$ as a value function. Then, $\psi(p) = \inf_x\{(1-p)x\}$ is such that $\psi(p) = 0$ if $p = 1$ and $\psi(p) = -\infty$ otherwise. Again, $\psi$ is not continuous on the interior of its domain, but it is so relative to the domain of the solution mapping $\mathscr{X}^*(p)$ of the dual problem, which is $\{1\}$. For this example, $\text{int dom } \mathscr{X}^* = \emptyset$, while the claim about continuity on compact subsets of the domain is useful.

### 4.3.2 Continuity and differentiability of the objects induced by smoothing

We proceed by stating further continuity/differentiability properties of our smooth approximations.

**Theorem 4.3.7** (Continuity of smoothed value function and solution mapping)**.** *Under Assumption 4.3.1, we have:*

*(i) If condition* (4.11) *holds, the value function of problem* (4.7) *is locally Lipschitz continuous relative to* $\text{int dom } \mathscr{X}^*$*, and continuous relative to any compact subset of* $\text{dom } \mathscr{X}^*$*.*

*(ii) Suppose the solution mapping of* (4.7) *is a singleton (*$\mathscr{X}^*(p) = x(p)$ *for all* $p \in \text{dom } \mathscr{X}^*$*). Assume, in addition, that:*

*(a) either* $\mu = 0$ *and the feasible set of problem* (4.7) *is uniformly bounded on* $\text{dom } \mathscr{X}^*$*,*

*(b) or* $\mu > 0$ *is fixed and condition* (4.11) *holds,*

*(c) or, most generally, conditions* (4.11) *and* (4.12) *hold, and* $\mu(\varepsilon)$ *is bounded for small* $\varepsilon > 0$*.*

*Then for any compact set* $D \subset \text{dom } \mathscr{X}^* \cap \text{dom } x^{\varepsilon,\mu(\varepsilon)}$*, it holds that*

$$\lim_{\varepsilon \searrow 0, p' \in D, p' \to p} x^{\varepsilon,\mu(\varepsilon)}(p') = x(p) \quad \textit{for all } p \in D. \tag{4.13}$$

*Proof.* Let us prove the first assertion in item (i). Fix any $\mu > 0$. Note that $\text{dom } v^{\varepsilon,\mu} = \mathbb{R}^q$, and that the local boundedness of $\nabla_p v^{\varepsilon,\mu}(\cdot)$ for small $\varepsilon > 0$ on $\text{int dom } \mathscr{X}^*$ follows from [BSS20, Theorem 2]. Then the claim follows from [Che12] and Proposition 2.1.1, noting that for $p \in \text{int dom } \mathscr{X}^*$ (see Section 2.3) we have

$$\partial_C v(p) \subset \text{conv}\left\{ \limsup_{\varepsilon \searrow 0, p' \to p} \nabla_p v^{\varepsilon,\mu}(p') \right\}.$$

For the second assertion in item (i), fix again any $\mu > 0$. Note that $\text{dom } v^{\varepsilon,\mu} = \mathbb{R}^q$ and $v^{\varepsilon,\mu}$ is a smooth function on $\mathbb{R}^q$ (see [BSS20, Corollary 1]). Formulas (4.9) and (4.11) imply that $v^{\varepsilon,\mu}(p)$ converges uniformly to $v(p)$ over compact subsets of $\text{dom } \mathscr{X}^*$. Because $v(p)$ is the uniform limit of the smooth functions $v^{\varepsilon,\mu}(p)$ on compact regions of $\text{dom } \mathscr{X}^*$, it has to be continuous relative to the compact subset of $\text{dom } \mathscr{X}^*$ taken.

Regarding item (ii), the first step is to show that (4.12) holds. This is the case if (a) holds, or if (b) holds via (4.10) or if (c) holds. In other words, the conditions (a), (b) or (c) imply that no subsequence of $x^{\varepsilon,\mu(\varepsilon)}(p')$ becomes unbounded when $\varepsilon \searrow 0$ and $p' \to p$ for $p' \in D$. Then, condition (4.9) and the continuity of $v$ on $D$ (shown in item (i)) imply that all subsequences of $x^{\varepsilon,\mu(\varepsilon)}(p')$ accumulate on the singleton $\mathscr{X}^*(p)$. In other words, (4.9) implies that

$$v(p') \leq f(x^{\varepsilon,\mu(\varepsilon)}(p'), p') \leq v(p') + m\varepsilon + \frac{\varepsilon\mu(\varepsilon)}{2} \min_{x \in \mathscr{X}^*(p')} \|x\|^2.$$

Taking limits on the last inequality under (a), (b) or (c) shows that $x^{\varepsilon,\mu(\varepsilon)}(p')$ converges as claimed. Note that the continuity of $v$ over $D$ is used. □

The multipliers of the affine equality constraints in (4.7) and (4.8), denoted respectively by $\lambda(p)$ and $\lambda^{\varepsilon,\mu}(p)$, play an important role in the calculations of the first-order derivatives of the value function and solution mapping. Because $B(p)$ has linearly independent rows, $\lambda^{\varepsilon,\mu}(p)$ exists and is unique whenever $x^{\varepsilon,\mu}(p)$ exists.

Differentiability properties of the regularization, relevant to solving problems (4.8) (and thus eventually (4.4)) computationally, are summarized below. In particular, explicit formulas for the derivatives of the primal-dual solutions of (4.8) are given. To this end, in the optimality conditions for (4.8) the sign of the Lagrange multiplier $\lambda^{\varepsilon,\mu}(p)$ is taken so that the following identity holds:

$$\nabla_x f(x^{\varepsilon,\mu}(p), p) + \varepsilon \nabla_x \phi^\mu(x^{\varepsilon,\mu}(p), p) - B(p)^\top \lambda^{\varepsilon,\mu}(p) = 0.$$

**Theorem 4.3.8** (Theorem 3.3.1 in Chapter 3, Theorem 1 in [BSS20])**.** *Under Assumption 4.3.1, the following holds.*

*(i) If $\mu > 0$, then $x^{\varepsilon,\mu}(p)$ and $\lambda^{\varepsilon,\mu}(p)$ are well-defined and are continuously differentiable in p, for all $p \in \mathbb{R}^q$.*

*(ii) If $\mu = 0$ and the constraints $x \geq 0$ are present in problem* (4.7)*, suppose that $\mathscr{X}^*(p)$ is pointwise bounded on* dom $\mathscr{X}^*$*. Then $x^{\varepsilon,\mu}(p)$ and $\lambda^{\varepsilon,\mu}(p)$ are well-defined on* dom $\mathscr{X}^*$*, and are continuously differentiable in p, for all $p \in$* int dom $\mathscr{X}^*$*.*

*(iii) Whenever $x^{\varepsilon,\mu}(p)$ and $\lambda^{\varepsilon,\mu}(p)$ are well-defined and their partial derivatives with respect to the r-th coordinate of $p \in \mathbb{R}^q$ exist, they are given as the solution of the linear system below:*

$$\begin{bmatrix} J^{\varepsilon,\mu}(p) & -B(p)^\top \\ B(p) & 0 \end{bmatrix} \begin{bmatrix} \partial_r x^{\varepsilon,\mu}(p) \\ \partial_r \lambda^{\varepsilon,\mu}(p) \end{bmatrix} = \begin{bmatrix} \theta_r^{\varepsilon,\mu}(p) + \varepsilon \varphi_r^{\varepsilon,\mu}(p) \\ \beta_r^{\varepsilon,\mu}(p) \end{bmatrix}, \tag{4.14}$$

*where, for $r = 1, \ldots, q$,*

$$\begin{aligned} J^{\varepsilon,\mu}(p) :=& \nabla^2_{xx} f(x,p) + \varepsilon \nabla^2_{xx} \phi^\mu(x,p) \Bigg|_{x=x^{\varepsilon,\mu}(p)}, \\ \theta_r^{\varepsilon,\mu}(p) :=& -\frac{\partial \nabla_x f(x,p)}{\partial p_r}\Bigg|_{x=x^{\varepsilon,\mu}(p)} + \frac{\partial B(p)^\top}{\partial p_r} \lambda^{\varepsilon,\mu}(p), \\ \varphi_r^{\varepsilon,\mu}(p) :=& -\frac{\partial \nabla_x \phi^\mu(x,p)}{\partial p_j}\Bigg|_{x=x^{\varepsilon,\mu}(p)} \\ \beta_r^{\varepsilon,\mu}(x) :=& \frac{\partial b(p)}{\partial p_r} - \frac{\partial B(p)}{\partial p_r} x^{\varepsilon,\mu}(p). \end{aligned}$$

### 4.3.3 Decomposition method across the agents

When in our problem (4.1)-(4.2) the total number $|\mathscr{A}|$ of agents is large, a direct solution approach can become too time consuming, and possibly even impossible. Having laid out the elements of our proposal, we are now in position to give our solution method, that converges epigraphically and allows for decomposition across agents. The method is first described conceptually, and then each step is explained further. Details of the actual implementation are not discussed, for now.

**Algorithm 4.3.9** (Smoothing decomposition of equilibrium problems)**.**

**Input and initialization.** Choose $p_1 \in \mathbb{R}^q$, $\sigma_1 > 0$, $\varepsilon_1 > 0$ and $\mu_1 \geq 0$. Set $k := 1$.

**Step 1: Price Iterate.** Find $p_{k+1}$ as an approximate local solution of

$$\operatorname{Arg}\min_p \{F(x_{\mathscr{A}}^{\varepsilon_k,\mu_k}(p)) : p \in \Pi \cap \tilde{\Pi}\},$$

taking $p_k$ as the starting point for the solver applied (for instance, Ipopt). During this solve, when the value and gradient of the objective function $F$ are needed (at points $\hat{p} \neq p_k$), e.g., when the callbacks for the objective are made, solve the subproblem below to get the required information.

**Step 1.1: Smoothed Subproblems.** For each $a \in \mathscr{A}$ solve (for instance, with Ipopt)

$$\min_x \left\{ f_a(x,\hat{p}) - \varepsilon_k \sum_{i=1}^m \log\{-g_{ai}(x,\hat{p})\} + \frac{\varepsilon_k \mu_k}{2} \|x\|^2 : B_a(\hat{p})x = b_a(\hat{p}) \right\}.$$

Let $x_a^{\varepsilon_k,\mu_k}(\hat{p})$ denote the unique minimizer in this problem, with the associated unique Lagrange multiplier $\lambda_a^{\varepsilon_k,\mu_k}(\hat{p})$. Compute the Jacobians of $x_a^{\varepsilon_k,\mu_k}(\hat{p})$ and $\lambda_a^{\varepsilon_k,\mu_k}(\hat{p})$, and use this information to compute the gradient of $F(x_{\mathscr{A}}^{\varepsilon_k,\mu_k}(\hat{p}))$.

**Step 2: Stopping Test.** If $F(x_{\mathscr{A}}^{\varepsilon_k,\mu_k}(p_{k+1}))$ stabilized relative to previous iterations, Stop.

**Step 3: Update Smoothing Parameters.** Determine $\sigma_{k+1}$, $\varepsilon_{k+1} > 0$ and $\mu_{k+1} \geq 0$, so that

$$|F(x_{\mathscr{A}}^{\varepsilon_{k+1},\mu_{k+1}}(p_{k+1})) - F(x_{\mathscr{A}}(p_{k+1}))| \leq \sigma_{k+1}.$$

**Step 4: Loop.** Set $k := k+1$ and go back to Step 1.

In practice, the rule to choose the parameters $\varepsilon_k, \mu_k$ has to be computationally inexpensive. In our case, once $\varepsilon_k$ is determined, possible options are taking a fixed $\mu_k = \mu > 0$ or setting $\mu_k = \sqrt{\varepsilon_k}$, or some similar simple choice. We use a one-dimensional bisection procedure to select $\varepsilon_k$, and therefore having a one-dimensional relation for $\mu_k$ is useful. For simplicity, in what follows we take $\mu_k = \mu > 0$. Other situations are dealt with by means of the parameterization just explained. In order to define a proper management of the sequence $\varepsilon_k \searrow 0$, we define the quantity

$$d^{\varepsilon}(p) := |v^{\varepsilon,\mu}(p) - v(p)|, \quad \mu = \mu(\varepsilon),$$

where

$$v^{\varepsilon,\mu}(p) := \begin{cases} F(x_{\mathscr{A}}^{\varepsilon,\mu}(p)) & p \in \Pi, \\ +\infty & p \notin \Pi, \end{cases} \quad \text{and} \quad v(p) := \begin{cases} F(x_{\mathscr{A}}(p)) & p \in \Pi, \\ +\infty & p \notin \Pi. \end{cases}$$

The need to handle dynamically $\varepsilon_k$ appeared because the output can be very poor when using some fixed sequence determined beforehand. This fact was confirmed by some runs in which $\varepsilon_k = 1/k$ resulted in slow convergence, while setting $\varepsilon_k = 1/(k^2)$ decreased the parameter too fast, making numerical errors dominate the iterative process. Fixing a priori exogenous values for that sequence appeared not to be suitable, particularly regarding accuracy. In some experiments, for a given $p_k$, we noticed that $\varepsilon_k = 10^{-6}$ is the maximal value for which $d^{\varepsilon_k}(p_k) \leq 10^{-2}$. As a result, setting $\varepsilon_k = 1/k$ would require $10^6$ iterations to bring the regularized agent's problem sufficiently close to the original one.

For these reasons, in Step 3 of Algorithm 4.3.9 we use the available information, and manage the parameter so that $\varepsilon_k$ is about the largest value possible for which $d^{\varepsilon_k}(p_k) \leq \sigma_k$, for $\sigma_k$ a decreasing sequence going to zero, depending only on certain available past information. Of course, we also do not want to spend an unreasonable amount of time calibrating $\varepsilon_k$ based on the last point $p_k$. The values $p_1 \in \Pi$ and $\varepsilon_1, \mu_1, \sigma_1 > 0$ are inputs of the algorithm. We take $\varepsilon_1$ "large". Given $p_{k+1} \in \Pi$ and $\varepsilon_k > 0$, for $k \geq 1$ we set

$$\sigma_{k+1} = \min\left\{ \frac{0.5 d^{\varepsilon_k}(p_{k+1})}{k+1}, \sigma_k \right\}.$$

The value $\varepsilon_{k+1} > 0$ is obtained as follows. We know that setting $\varepsilon = 0$ yields $d^{\varepsilon}(p_{k+1}) = 0$ and want to find $\varepsilon_{k+1}$ such that $d^{\varepsilon_{k+1}}(p_{k+1}) \leq \sigma_{k+1}$ taking into account that $d^{\varepsilon}(p)$ is continuous in $\varepsilon$. The interval to make this search is $[0, C_{k+1}]$ with $C_{k+1} = 1.3\varepsilon_k$. We allow for $\varepsilon_{k+1} > \varepsilon_k$. Then, we start a bisection procedure on $[0, C_{k+1}]$ trying to match the value $\sigma_{k+1}$, and stop once we find a value $\varepsilon_{k+1}$ such that $d^{\varepsilon_{k+1}}(p_{k+1}) \leq \sigma_{k+1}$, or the maximal number of trials is reached, or $d^{\varepsilon_{k+1}}(p_{k+1})$ is close enough to $\sigma_{k+1}$. Note that close enough here is understood loosely. For example, with distance between $0.1\sigma_{k+1}$ and $0.3\sigma_{k+1}$. The point is that this calculation does not need to be precise. However, it has to be precise enough to guarantee that $d^{\varepsilon_{k+1}}(p_{k+1})$ decreases to zero, hence ensuring that the regularized models get closer and closer to the true model near an accumulation point of the sequence $p_{k+1}$.

Another relevant issue for implementation is when to stop the minimization process when solving problems in Step 1. In order not to blindly rely on the stopping criteria of the solver, the focus should be put on robust decrease. If the price sequence converges, then $d^{\varepsilon_k}(p_k)$ yields a current estimate of how close the regularized model is to the true model. If while solving (4.4) consecutive iterates have objective function values differing in less than $0.5d^{\varepsilon_k}(p_k)$, this hints that the current $\varepsilon_k, \mu_k$ may not be meaningful in providing solutions for the original problem. These rules are based on comparison of functional values and not on gradient information (with our epigraphical convergence approach to the problem, the unknown property of gradient consistency cannot be exploited algorithmically).

We finish with some useful practical considerations for the implementation. In practice we want to avoid being subject to numerical instabilities associated with $\varepsilon_k > 0$ being too small. To circumvent this issue, we add triggers to store the record (the best iterate while solving one instance of the price problem in Step 1) as well as the best iterates between different instances of that problem. Also, while building the matrix of the linear system (4.14), we have to make sure that the iterate lies in the interior of the feasible set with some safeguards, because factors like $g_{ai}(x_a^{\varepsilon,\mu}(p), p)^{-2}$ appear in the diagonal (and so numerical errors are amplified to the square). For instance, if $g_{ai}(x_a^{\varepsilon,\mu}(p), p) = -10^{-5} < 0$, the term in the diagonal is $10^{10}$. Currently, the management of these numerical errors is done the simplest way possible, because it is enough for the applications we tried. However, as Algorithm 4.3.9 may need to solve thousands of optimization problems, being robust to failure is essential. These safeguards are not detailed in our description, but the guiding principles are listed below:

1. use an adaptive rule for $\varepsilon_k$, $\mu_k$ and $\sigma_k$ so that problems in Step 1 are closer and closer to the model (4.2),

2. early stop the subproblem solution in Step 1 if the difference in objective function values on consecutive iterations is smaller than a fraction of $d^{\varepsilon_k}(p_k)$, and

3. carefully manage numerical errors and failures, taking into account that the algorithm is solving thousands of optimization problems, and that even if a fraction of those fail, the algorithm has to keep running.

The convergence result below is based on relation (4.13) holding for the solution mappings $x_{\mathscr{A}}^{\varepsilon,\mu}(p)$ and $x_{\mathscr{A}}(p)$. There are two qualitatively distinct cases to keep in mind, these are: $\liminf \mu_k > 0$ and $\liminf \mu_k = 0$. The assumption that the feasible sets of the problems are uniformly bounded is enough for both cases. Note that if $\liminf \mu_k > 0$, we can allow for unbounded solution sets as well, enforcing condition (ii)(b) or (ii)(c) of Theorem 4.3.7.

**Theorem 4.3.10** (Convergence result for Algorithm 4.3.9)**.** *Fix a compact set $K \subset \Pi$. Let Assumption 4.3.1 for the agents' problems* (4.1) *hold, and assume in addition that*

*(i) either the feasible sets of problems* (4.1) *are uniformly bounded for all $p \in K$,*

*(ii) or if in Algorithm 4.3.9 we have $\liminf \mu_k > 0$, the solution mappings of problems* (4.1) *have locally bounded selections for all $p \in K$,*

*(iii) or, most generally, conditions* (4.12) *and* (4.11) *hold, and $K \subset \operatorname{dom} \mathscr{X}_a^* \cap \operatorname{dom} x_a^{\varepsilon_k,\mu_k}$ for all $a \in \mathscr{A}$.*

*Then, the sequence of functions $v^{\varepsilon_k,\mu_k}$ converges epigraphically to $v$ over $K$.*

*Proof.* Recall that $\{\mu_k\}$ is bounded. If (i) holds, then conditions (4.12) and (4.11) hold. Therefore, (ii)(c) of Theorem 4.3.7 holds and the conclusion follows. If (ii) holds, then (4.12) holds because of (4.10) and the fact that solution mappings have locally bounded selections, which is condition (4.11). Therefore, the conclusion follows again. Finally, (iii) easily implies (ii)(c) of Theorem 4.3.7. □

The importance of the set $D$ considered in (4.13) lies in ensuring epigraphical convergence on a feasible region containing an accumulation point of the sequence $p_k$. For instance, for the WEP defined below the epigraphical convergence result does not apply on the boundary of $\Pi$. Also note that condition in item (ii) of the theorem above is equivalent to requiring satisfaction of (4.11) for problems (4.1), uniformly over $p \in K$.

## 4.4 Solving deterministic equilibrium problems

As mentioned in the introduction, PATH [FM99] is an established code to solve equilibrium problems. Being a Newton-type method (see, e.g., [IS14, Chapter 5.2.2]), when PATH works, it tends to get higher precision than our smoothing. However, when solving Walrasian equilibrium problems, it appears that one needs to calibrate PATH parameters very carefully, because certain utility functions in the agent's problems (4.1) are ill-conditioned and degeneracies occur (singular-basis, using PATH-related language). Even with careful tuning, this leads to failures with a certain frequency, especially for larger instances.

We next describe a first family of problems employed in our experiments, and then benchmark the performance of Algorithm 4.3.9 against PATH.

### 4.4.1 Deterministic Walrasian equilibrium problems

A Walrasian Equilibrium Problem is defined for an economy with agents in a set $\mathscr{A}$, of cardinality $|\mathscr{A}|$. There are $n$ goods whose prices form a vector $p \in \mathbb{R}^n$. The agent's consumption is a vector $x \in \mathbb{R}^n_+$, so in (4.1)-(4.2) the dimensions are $q = n$ and $n_a = n$ for all $a \in \mathscr{A}$. Each agent has at its disposal an initial amount of goods $e_a \in \mathbb{R}^n$, called endowment, that is worth $p^\top e_a$. The consumption benefit is measured using some strictly concave utility function $u_a(\cdot)$. Accordingly, given a price $p$, the agent's optimal decisions are

$$x_a(p) := \arg\max_{x \geq 0} \{u_a(x) : p^\top x \leq p^\top e_a\}. \tag{4.15}$$

With respect to problem (4.1), the objective function therein is $f_a(x,p) := -u_a(x)$. As explained below, for some utilities the budget constraint can be replaced by an equality constraint, so the only inequalities in (4.1) refer to non-negativity of the decision variable. These are handled by the penalty $\phi^\mu$ introduced in (4.8).

The global problem (4.2) minimizes the excess supply, which results in the following:

$$p^* \in \arg\min_{p\in\Delta} \frac{1}{|\mathscr{A}|^2} \left\| \sum_{a\in\mathscr{A}} x_a(p) - \sum_{a\in\mathscr{A}} e_a \right\|^2, \text{ where } \Delta := \{\tau > 0 : \sum_{j=1}^{n} \tau_j = 1\}. \tag{4.16}$$

In particular, $\Delta$ is the unit simplex of strictly positive prices. Notice that letting

$$\bar{x}_{\mathscr{A}}(p) := \frac{1}{|\mathscr{A}|} \sum_{a\in\mathscr{A}} x_a(p) \quad \text{and} \quad \bar{e}_{\mathscr{A}}(p) := \frac{1}{|\mathscr{A}|} \sum_{a\in\mathscr{A}} e_a(p), \tag{4.17}$$

the objective function in (4.16) amounts to measuring the distance between consumption and endowment, averaging over all agents in the economy. If the price succeeds in clearing the market, the optimal value in (4.16) is zero, the information that we employ to assess the quality of the output of the solution methods compared in our experiments.

As already mentioned above, the solution set in (4.15) is empty for some $p \notin \Delta$. This depends on a property of the utility function, said to be *non-satiable*. In economics, non-satiation is the assumption that a consumer will always benefit from additional consumption. In consumer theory, the utility refered to as having constant elasticity of substitution (CES) satisfies the property of non-satiation. This strictly concave function combines in one number the preference of consuming $n$ types of goods, assuming they have an elasticity of substitution $1 \neq b > 0$ and that a number $\gamma_j > 0$ indicates the preference for the $j$-th good:

$$\textsc{ces}(x) := \left( \sum_{j=1}^{n} \gamma_j^{\frac{1}{b}} x_j^{\frac{b-1}{b}} \right)^{\frac{b}{b-1}} \quad \text{for all} \quad x > 0. \tag{4.18}$$

When $b \to \infty$ the goods behave like perfect substitutes and as $b \searrow 0$ they behave like perfect complements. The property that $\textsc{ces}(\alpha x) = \alpha\textsc{ces}(x)$ for $\alpha > 0$ and $x > 0$ reflects the belief that the intrinsic utility of a vector of goods lies on the proportion between the goods, rather than on their magnitude. This is confirmed by the fact that the consumption $x_a(p)$ solving problem (4.15) written with utility $\textsc{ces}$ satisfies the relation $x_a(\alpha p) = x_a(p)$ for any $\alpha > 0$ and $p \in \Delta$. The domain of the CES utility function is the whole space for $b > 1$.

The deterministic WEP model (4.15)-(4.16) represents an exchange economy, without production. Agents cannot spend more than the worth of the initial endowment. When the price of some good becomes non-positive, the feasible set in (4.15) gets unbounded. In this case, if the utility $u_a(x)$ is of non-satiable type, the agent will try to spend more and more, yielding decisions $x_a(p)$ that are not well-defined.

These issues with the utility functions would result in failure for methods that need the problem data to be defined, and have Lipschitz derivatives, on the whole space. By contrast, they do not affect our method, because the smooth regularized solutions of problem (4.8) use the utility derivatives only locally, near the smoothed consumption of the agents. Another advantage of our approach is that, in practice, larger values of $\varepsilon > 0$ tend to imply less abrupt changes on the functions involved in (4.20). As a result, the agents' problems are somehow easier to solve. Computationally, as the method moves $\varepsilon > 0$ from larger to smaller values, the agent problem (4.15) goes from easier to harder.

## 4.4.2 First numerical benchmark

For non-satiable utilities, the budget inequality constraint in (4.15) can be changed by an equality without changing the solution set. The agents' problems take the form

$$x_a(p) = \arg\max_{x\geq 0} \{\textsc{ces}_a(x) : p^\top x = p^\top e_a\},$$

and so our smoothing penalizes the non-negativity constraint. The problems in Step 1.1 of Algorithm 4.3.9 are given by

$$x_a^{\varepsilon_k,\mu_k}(p_k) := \arg\min_x \{-\textsc{ces}_a(x, p_k) - \varepsilon_k \sum_{i=1}^{n} \log(x_i) + \frac{\varepsilon_k \mu_k}{2} \|x\|^2 : p_k^\top x = p_k^\top e_a\}.$$

These problems are solved with Ipopt [WB05], setting the mu-target option available for the solver so that $x_a^{\varepsilon_k,\mu_k}(p_k)$ is automatically computed instead of $x_a(p)$. An optimized build of Ipopt with state-of-the-art linear algebra software Pardiso [KFS18] is essential for reproducing performance. Otherwise, speed may be sacrificed. See [KFS18] for one of the advanced applications of Pardiso.

Our smoothing method, denoted by Alg. 4.3.9 in the tables, was coded in CPP (g++ 7.5.0) with the initial value $\varepsilon_1 = 1$ and keeping $\mu_k = 1$ fixed. All experiments are run on an Intel i7 1.90GHz machine, but using only one thread when comparing with PATH. The operating system is Ubuntu 18.04.3 LTS.

We start with an example from [DJW17], with the same CES utility for all the agents, taking in (4.18) the preferences $\gamma_j = 1.0$ and the power $b = 0.5$. Endowments are also the same, $e_a = (1, \dots, 1)$ for all $a \in \mathscr{A}$.

In this setting, the equilibrium price is unique and known, $p^* = (\frac{1}{n}, \dots, \frac{1}{n})$, where $n$ is the number of goods. In other words, if all agents have the same buying power and the same preferences, all the goods have the same price at equilibrium.

We consider $|\mathscr{A}| = 2$ agents exchanging $n \in \{2, 10, 20, 30\}$ goods. Notice that the objective function in (4.16) measures the capacity of the economy in clearing the market. Hence, the columns "Initial/Clearing" (both for Alg. 4.3.9 and PATH-Clearing) correspond, respectively, to the initial and final objective function values (found with our approach and with PATH). For each column, the output in Table 4.1 reports the average and standard deviation, computed by repeating the experiments of each configuration four times. For this case, four repetitions is enough because the standard deviation is small, as can be seen in Table 4.1.

| $n$ | Initial Clearing | Alg. 4.3.9-Time (sec) | Alg. 4.3.9-Clearing | PATH-Time (sec) | PATH-Clearing |
|---|---|---|---|---|---|
| 2 | 0.03 / 0.04 | 0.03 / 0.01 | $10^{-6}$ / $10^{-6}$ | 0.01 / 0.00 | $10^{-33}$ / $10^{-33}$ |
| 10 | 2.67 / 1.96 | 0.13 / 0.01 | $10^{-5}$ / $10^{-5}$ | 0.57 / 1.09 | $10^{-33}$ / $10^{-33}$ |
| 20 | 3.43 / 1.79 | 0.16 / 0.02 | $10^{-4}$ / $10^{-4}$ | 0.07 / 0.00 | $10^{-32}$ / $10^{-32}$ |
| 30 | 4.19 / 0.83 | 0.21 / 0.02 | $10^{-4}$ / $10^{-4}$ | 0.13 / 0.00 | $10^{-32}$ / $10^{-32}$ |

Table 4.1: Comparison between Alg. 4.3.9 and PATH for deterministic WEP with symmetric agents.

Except for the experiments with $n = 10$, where PATH seems to have struggled for one run, the Newtonian updates in PATH make the output more precise and faster for this set of problems, as expected.

The second experiment, originally from [Sca73] and also reported in [DJW17], has $|\mathscr{A}| = 5$ agents and $n = 10$ goods. In (4.18), the CES utilities for the agents are defined with the following values for the power

$$b_1 = 2.0, \quad b_2 = 1.3, \quad b_3 = 3.0, \quad b_4 = 0.2, \quad b_5 = 0.6,$$

and the coefficients $\gamma$ reported in Table 4.2. The table also contains the initial endowments of the agents.

| $a$ | $\gamma_1$ | $\gamma_2$ | $\gamma_3$ | $\gamma_4$ | $\gamma_5$ | $\gamma_6$ | $\gamma_7$ | $\gamma_8$ | $\gamma_9$ | $\gamma_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.0 | 1.0 | 3.0 | 0.1 | 0.1 | 1.2 | 2.0 | 1.0 | 1.0 | 0.7 |
| 2 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 3 | 9.9 | 0.1 | 5.0 | 0.2 | 6.0 | 0.2 | 8.0 | 1.0 | 1.0 | 0.2 |
| 4 | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 | 9.0 | 10.0 |
| 5 | 1.0 | 13.0 | 11.0 | 9.0 | 4.0 | 0.9 | 8.0 | 1.0 | 2.0 | 10.0 |
| $a$ | $e_{a,1}$ | $e_{a,2}$ | $e_{a,3}$ | $e_{a,4}$ | $e_{a,5}$ | $e_{a,6}$ | $e_{a,7}$ | $e_{a,8}$ | $e_{a,9}$ | $e_{a,10}$ |
| 1 | 0.6 | 0.2 | 0.2 | 20.0 | 0.1 | 2.0 | 9.0 | 5.0 | 5.0 | 15.0 |
| 2 | 0.2 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 | 16.0 | 5.0 | 5.0 | 9.0 |
| 3 | 0.4 | 9.0 | 8.0 | 7.0 | 6.0 | 5.0 | 4.0 | 5.0 | 7.0 | 12.0 |
| 4 | 1.0 | 5.0 | 5.0 | 5.0 | 5.0 | 5.0 | 5.0 | 8.0 | 3.0 | 17.0 |
| 5 | 8.0 | 1.0 | 22.0 | 10.0 | 0.3 | 0.9 | 5.1 | 0.1 | 6.2 | 11.0 |

Table 4.2: CES utility coefficients and initial endowments for the agents.

We sampled ten prices in $\Delta$, and both algorithms were executed starting from these prices. The average and standard deviation for the quantities of interest are reported in Table 4.3. The final prices agree with those found in [DJW17]. Note that while PATH is still more precise, the smoothing approach is now faster (to termination).

| | Initial Clearing | Alg. 4.3.9-Time (sec) | Alg. 4.3.9-Clearing | PATH-Time (sec) | PATH-Clearing |
|---|---|---|---|---|---|
| Avg. | 7056.23 | 2.30 | $10^{-5}$ | 10.24 | $10^{-10}$ |
| Std. | 3401.70 | 1.76 | $10^{-2}$ | 5.45 | $10^{-8}$ |

Table 4.3: Comparison between Alg. 4.3.9 and PATH for [Sca73] problem.

### 4.4.3 Scaling capabilities of the Algorithm

In order to explore decomposition with respect to the number of agents, we extend the previous example to an economy with $n = 80$ goods, for $|\mathscr{A}|$ ranging from 2 to 640 agents.

| $\lvert\mathscr{A}\rvert$ | Initial Clearing | Alg. 4.3.9-Clearing | Alg. 4.3.9-Time (sec) |
|---|---|---|---|
| 2 | 15.56 / 5.11 | 0.66 / 0.21 | 10.26 / 2.63 |
| 4 | 10.00 / 10.59 | 0.12 / 0.08 | 8.74 / 4.45 |
| 8 | 5.17 / 1.28 | 0.02 / 0.01 | 22.77 / 13.47 |
| 10 | 5.43 / 0.61 | 0.02 / 0.00 | 18.20 / 9.98 |
| 20 | 6.79 / 2.15 | 0.01 / 0.02 | 17.61 / 9.32 |
| 40 | 5.79 / 2.05 | 0.01 / 0.01 | 45.70 / 38.24 |
| 80 | 4.45 / 1.79 | 0.00 / 0.00 | 37.21 / 26.06 |
| 160 | 3.91 / 0.92 | 0.01 / 0.01 | 53.06 / 15.26 |
| 320 | 4.90 / 0.58 | 0.04 / 0.01 | 246.49 / 168.07 |
| 640 | 4.36 / 0.50 | 0.06 / 0.02 | 654.76 / 387.20 |

Table 4.4: Illustration of the decomposition properties of the smoothing with respect to the number of agents.

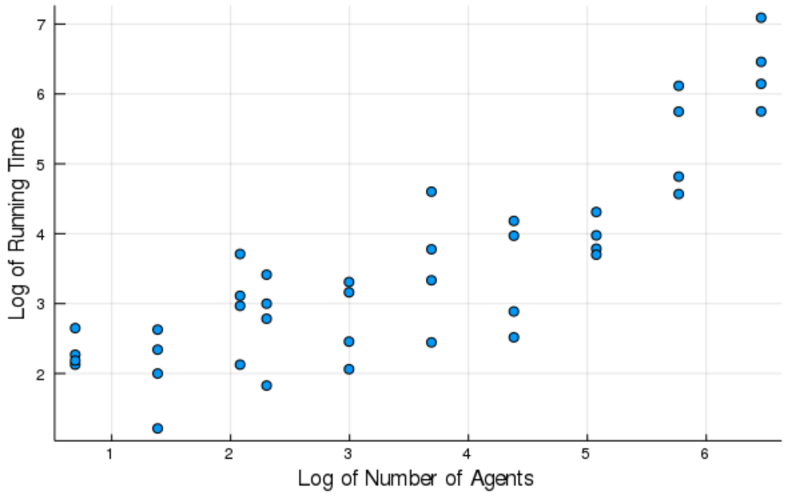

Figure 4.1: In log scale, the running times grow linearly with respect to the number of agents. Each dot is one experiment. (We use log scale because the total number of agents grows exponentially.)

We sample the $\gamma$ coefficients of the utilities in the box $[0.1,1]^n$, and sample $b$ in $[0.1,0.9]$ for all agents. For each configuration we start from four random initial prices. The results, reported as average / standard deviation, are shown in Table 4.4 and in Figure 4.1. We use eight threads.

## 4.5 Decomposition of stochastic hierarchical problems

Another model for the WEP represents an economy with production subject to uncertain delivery, that we handle in two stages. We give the corresponding formulation, then solve some instances using Algorithm 4.3.9, and finish with a decomposition method that exploits the two-stage structure of the agents' problems.

### 4.5.1 Stochastic Walrasian equilibrium

In this model, consumption decisions taken in the first stage are $x_a^0$ as well as another vector $z_a$ of production activity levels. The production effort takes place in the first stage and the resulting goods are delivered in the second stage in uncertain amounts, represented with a set $\mathscr{S}$ of equiprobable scenarios, of cardinality $|\mathscr{S}|$. The consumption decided in the second stage for the $s$-th scenario is $x_a^s$.

In the first stage, we need $B_a^0 z$ goods to start the production, so that the resulting amount of goods for each second-stage scenario is $B_a^s z$. The cost of the production effort in the first stage is $(p^0)^\top B_a^0 z$, and the profit in the

$s$-th scenario is $(p^s)^\top B_a^s z$. The resulting stochastic version of the agent's problem (taken from [DJW17]) with CES utilities and corresponding equality in the budget constraint is

$$\left(z_a(p), x_a^0(p), x_a^1(p), \dots, x_a^{|\mathscr{S}|}(p)\right) = \begin{cases} \underset{z,x^0,x^1,\dots,x^{|\mathscr{S}|}}{\arg\max} & \text{CES}_a^0(x^0) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} \text{CES}_a^s(x^s) \\ \text{s.t.} & (p^0)^\top x^0 = (p^0)^\top e_a^0 - (p^0)^\top B_a^0 z, \\ & (p^s)^\top x^s = (p^s)^\top e_a^s + (p^s)^\top B_a^s z \quad \text{for } s \in \mathscr{S}, \\ & z, x^0, x^1, \dots, x^{|\mathscr{S}|} \geq 0. \end{cases} \tag{4.19}$$

Again, we need to analyze whether the feasible set of problem (4.19) is bounded near a strictly positive price $p$. By examining the first constraint of (4.19), it suffices to take the matrices $B_a^s$ so that all production activities $z_a$ have a positive cost if the prices of all goods are positive. Under this condition, feasible sets are uniformly bounded and each agent problem (4.19) satisfies Assumption 4.3.1 because the equality constraints are indeed linearly independent.

In (4.19), uniqueness of the optimal consumption follows from strict concavity of the utility functions. Uniqueness of the production levels $z_a(p)$ depends on the existence of a unique solution in $z$ to the system

$$z \geq 0, \quad (p^0)^\top B_a^0 z = (p^0)^\top e_a^0 - (p^0)^\top x^0, \quad (p^s)^\top B_a^s z = -(p^s)^\top e_a^s + (p^s)^\top x^s \quad \text{for } s \in \mathscr{S},$$

for fixed values $x_a^0, \dots, x_a^{|\mathscr{S}|}$. This issue is related to market completeness. In an incomplete market, the total of activities (sometimes called financial instruments) is less than the amount of scenarios (or future states of the world). In this case, the system above is over-determined and uniqueness of the production levels depends on the scenario realizations. If the market is complete (there are at least as many activities as scenarios), $z_a(p)$ may be non-unique because the system above is under-determined. In any case, the smoothing of the solution mappings is always well-defined. The assumption on the single-valuedness of $x_a(p)$ is used only for the convergence analysis. In practice, independently of assumptions, the algorithm will minimize the objective.

The price that best balances demand and supply is determined similarly to (4.16), considering the expected value of the clearing for the second stage

$$p^* \in \arg\min_{p\in\Pi} \left\{ \|\bar{x}_{\mathscr{A}}^0(p) - \bar{e}_{\mathscr{A}}^0 + \overline{B}^0 z_{\mathscr{A}}(p)\|^2 + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} \|\bar{x}_{\mathscr{A}}^s(p) - \bar{e}_{\mathscr{A}}^s - \overline{B}^s z_{\mathscr{A}}(p)\|^2 \right\}. \tag{4.20}$$

Similarly to (4.17), in this problem we defined the mean consumption, endowment and production transfers, averaged for all the economy, for each scenario $s \in \{0\} \cup \mathscr{S}$:

$$\bar{e}_{\mathscr{A}}^s := \frac{1}{|\mathscr{A}|}\sum_{a\in\mathscr{A}} e_a^s, \qquad \bar{x}_{\mathscr{A}}^s(p) := \frac{1}{|\mathscr{A}|}\sum_{a\in\mathscr{A}} x_a^s(p), \quad \overline{B}^s z_{\mathscr{A}}(p) := \frac{1}{|\mathscr{A}|}\sum_{a\in\mathscr{A}} B_a^s z_a(p)$$

(for simplicity, the first-stage is referred to as the 0th scenario).

### 4.5.2 Numerical experiments

We report on performance of the proposed method for some examples found in the literature [Sca73; DJW17; Sch12], and analyze the impact of properly managing the parameter $\varepsilon_k$. Since the method described in [DJW17] is of derivative-free type, it needs to solve many subproblems to estimate the derivatives, and thus is not comparable with our approach. The stochastic setting leads to larger problems, and PATH starts to fail too often to collect meaningful information. Thus, PATH results are not reported. As for the method in [Sch12], it cannot be used to obtain decomposition.

The results for the deterministic Scarf's instance are shown in Figure 4.2. It reports the probability of each strategy delivering a reduction of x% of the initial clearing within the normalized budget time in $[0,1]$. Because the strategy $1/k$ does not drive $\varepsilon_k$ to zero fast enough, it lacks precision.

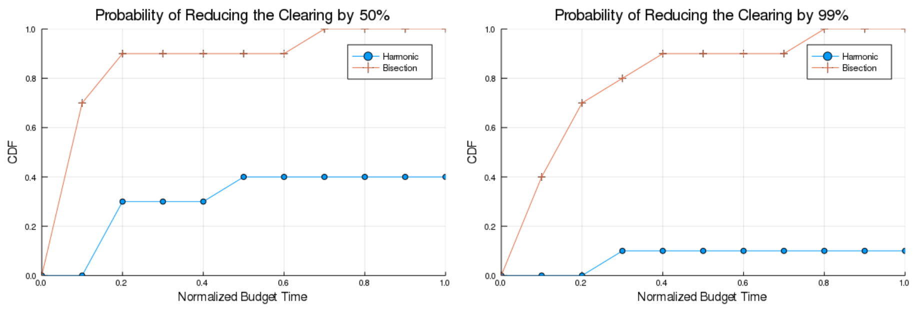

Figure 4.2: Data profile showing the impact on performance of the management of the parameter $\varepsilon_k$ for the Scarf's instance.

In the second test-case, our smoothing method is applied to the stochastic WEP described in [DJW17]. It has $|\mathscr{S}| = 9$ scenarios, $|\mathscr{A}| = 5$ agents, $n = 7$ goods, all with production. The problem data for the stochastic instance was obtained directly from the authors of [DJW17]. For this example we also use only one thread. The results are reported in Table 4.5. Note that the final clearing is not as close to zero as in the deterministic setting (for instance, see Table 4.4). Here, the algorithm converged to a price with positive clearing. This can happen, since the method is guaranteed to find only local solutions. Moreover, for the stochastic setting the equilibrium is not guaranteed to exist due to issues with complete and incomplete markets.

| | Initial Clearing | Alg. 4.3.9-Time (sec) | Alg. 4.3.9-Clearing |
|---|---|---|---|
| Avg. | 28512.10 | 81.27 | 5.41 |
| Std. | 60275.63 | 14.54 | 1.61 |

Table 4.5: Results for the stochastic WEP from [DJW17].

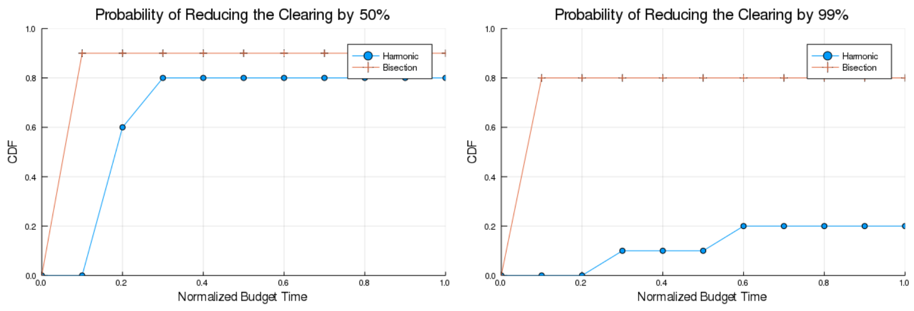

Figure 4.3: Data profile showing the impact on performance of the management of the parameter $\varepsilon_k$ for the stochastic instance.

We finish our computational analysis with showing that it is worth to invest some effort in the choice of the smoothing parameter $\varepsilon_k$ in Step 3 of Algorithm 4.3.9. The strategy described in Section 4.3.3 is inner/outer iteration scheme. In the inner step we minimize the smooth approximation and in the outer step we measure the quality of the approximation and calibrate $\varepsilon_k$. In this case $k$ is the number of outer iterations. For the strategy $\varepsilon_k = 1/k$ to make practical sense, the meaning of $k$ is different. In this case, it is the number of times the objective function oracle of the smooth approximation is called by Ipopt. Due to these differences, we compare both strategies using time budgets in a data profile.

More precisely, for a given instance of WEP, we run both algorithms recording the price iterates of the smooth approximations and the time when the iterate is generated. After the runs finish, we compute separately the true clearing for the price iterates. This procedure is repeated for ten random initial prices for each instance. We then make data profiles [MW09; DM02] reporting the probability of the best results until a certain time delivering a percentage decrease of the initial clearing. The instances are the ones from [Sca73] reported above, and the stochastic instance is from [DJW17], also used above. Those results are reported in Figure 4.3. Note that again the $1/k$ strategy is not as accurate, and also that the bisection strategy for $\varepsilon_k$ finds a good solution early in the process.

### 4.5.3 Inducing decomposition across scenarios

For simplicity, we consider the regularization parameter $\mu > 0$ fixed during this section, and all feasible sets in consideration uniformly bounded. The previous developments assume that the agents solve a parameterized convex problem with no special structure. In this section, we show how a two-stage stochastic structure can be used to obtain decomposition across scenarios. Computational implementation of this additional decomposition is a technically more involved topic, which we do not pursue here. But the proof of convergence can be done with the tools already developed in this chapter.

The agents' problems are given by

$$\begin{cases} \min\limits_{x^0,\dots,x^{|\mathscr{S}|}} & f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} f_a^s(x^s,p) \\ \text{s.t.} & B_a^i(p)x^i + C_a^i(p)x^0 = b_a^i(p) \qquad \text{for } i = 0,\dots,|\mathscr{S}|, \\ & g_a^i(x^i,p) + h_a^i(x^0,p) \le 0 \qquad \text{for } i = 0,\dots,|\mathscr{S}|. \end{cases} \tag{4.21}$$

In particular, if the problem above is the stochastic WEP, the inequalities are only $z \ge 0$ and $x^s \ge 0$ for $s \in \{0,\dots,|\mathscr{S}|\}$. In spite of some abuse of notation, the first stage variables of the current two-stage problem would be $x^0 := (z, x_w^0)$, where $x_w^0$ is the first stage consumption of the stochastic WEP. In other words, the activities are first stage decisions for the stochastic WEP. One has to map carefully the constraints of the general problem to that of the stochastic WEP. The objective function would be given by

$$f_a^s(x^s,p) = -\texttt{CES}_a^s(x^s) \quad \text{for all } s \in \{0,\dots,|\mathscr{S}|\}.$$

Now, there is one price for each scenario/stage configuration. This implies that $\Pi$ has the form

$$\Pi = \Delta^{1+|\mathscr{S}|}, \quad \text{where} \quad \Delta = \{\tau > 0 : \sum_j \tau_j = 1\}.$$

To write the optimal value reformulation it is useful to define the first stage feasible set

$$X_a^0(p) := \{x^0 \in \mathbb{R}^{n_a} : \quad B_a^0(p)x^0 = b_a^0(p), \quad g_a^0(x^0,p) \le 0\}.$$

The last problem, when written using the usual value function reformulation becomes

$$x_a^0(p) := \arg\min_{x^0}\{f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} Q_a^s(x^0,p) : x^0 \in X_a^0(p)\}, \tag{4.22}$$

where $x_a^s(x^0,p)$ and $Q_a^s(x^0,p)$ are the unique solution and value function of the scenario subproblem:

$$\min_{x^s}\{f_a^s(x^s,p) : \quad B_a^s(p) + C_a^s(p)x^0 = b_a^s(p), \quad g_a^s(x^s,p) + h_a^s(x^0,p) \le 0\}. \tag{4.23}$$

Note that $Q_a^s(x^0,p)$ is convex in $x^0$ for a fixed $p$, because: (i) $x^0$ influences the right-hand side of the equality constraints in a linear manner, and (ii) the function $g_a^s(x^s,p) + h_a^s(x^0,p)$ is jointly convex in $x^s$ and $x^0$ (since (4.21) is convex for all $p$).

The idea is to parameterize the second stage decisions of the agent's problem, denoted by $x_a^s(x^0,p)$, as functions of both the first stage decision $x^0$ and the price vector $p \in \Pi$. As usual, to get a smooth approximation to this function we define, for $s = 1,\dots,S$, the Tikhonov-regularized log-barrier penalty for the problem (4.23) by

$$\phi_a^{s,\mu}(x^s,p,x^0) := -\sum_{i=1}^{m_a^s} \log\{-g_{ai}^s(x^s,p) - h_{ai}^s(x^0,p)\} + \frac{\mu}{2}\|x^s\|^2.$$

If the scenario subproblems have only the inequalities $x^s \ge 0$, then

$$\phi_a^{s,\mu}(x^s,p,x^0) := -\sum_{i=1}^{m_a^s} \log\{x_i^s\} + \frac{\mu}{2}\|x^s\|^2.$$

Then, for any $\varepsilon > 0$, the smoothed approximation of $x_a^s(x^0,p)$ and the associated smoothing for the value function of the scenario subproblems are given by

$$x_a^s(\varepsilon,x^0,p) := \arg\min_{x^s}\{f_a^s(x^s,p) + \varepsilon\phi_a^{s,\mu}(x^s,p,x^0) : \quad B_a^s(p) + C_a^s(p)x^0 = b_a^s(p)\} \tag{4.24}$$

and

$$Q_a^s(\varepsilon,x^0,p) := f_a^s(x_a^s(\varepsilon,x^0,p),p).$$

Although $Q_a^s(x^0,p)$ is convex in $x^0$ for a fixed $p$, the smoothed value function $Q_a^s(\varepsilon,x^0,p)$ is only "approximately" convex [BSS20, Lemma 3]. The approximation of $Q_a^s(x^0,p)$ which is guaranteed to be convex in $x^0$ is given by

$$P_a^s(\varepsilon,x^0,p) := Q_a^s(\varepsilon,x^0,p) + \varepsilon\phi_a^{s,\mu}(x_a^s(\varepsilon,x^0,p),p,x^0).$$

By [BSS20, Lemma 3], we know that $\varepsilon\phi_a^{s,\mu}(x_a^s(\varepsilon,x^0,p),p,x^0) \to 0$ in a controlled manner when $\varepsilon \searrow 0$.

Because of the stated blanket assumptions, problems (4.24) satisfy the Slater condition for all $p$ and $x^0$, the rows of $B_a^s(p)$ are linearly independent, and problem (4.24) is convex. Therefore, if $\mu > 0$, it follows that $x_a^s(\varepsilon,x^0,p)$ is well-defined and smooth, even if $x_a^s(x^0,p)$ is not. The same holds for $P_a^s(\varepsilon,x^0,p)$. Note now that one smooth approximation (not necessarily from above) for the objective function of master problem (4.22) is given by

$$x_a^0(\varepsilon,p) := \arg\min_{x^0}\{f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} P_a^s(\varepsilon,x^0,p) : x^0 \in X_a^0(p)\}. \tag{4.25}$$

Note that in (4.25) we need to use $P_a^s(\varepsilon,x^0,p)$ instead of $Q_a^s(\varepsilon,x^0,p)$, so that problem (4.25) is smooth and convex for all $p$, our assumptions hold and the results apply.

The solution mapping $x_a^0(\varepsilon,p)$ is not guaranteed to be a smooth function of $p$. For this reason, we have to deal with the Tikhonov-regularized log-barrier for the first stage problem, given by

$$\phi_a^{0,\mu}(x^0,p) := -\sum_{i=1}^{m_a^0}\log\{-g_{ai}^0(x^0,p)\} + \frac{\mu}{2}\|x^0\|^2.$$

Again, when the first stage problem is linear and the only inequality constraints are $x^0 \geq 0$, we see that

$$\phi_a^{0,\mu}(x^0,p) := -\sum_{i=1}^{m_a^0}\log\{x_i^0\} + \frac{\mu}{2}\|x^0\|^2.$$

The smooth approximation $x_a^0(\delta,\varepsilon,p)$ for the first stage decision $x_a^0(\varepsilon,p)$ is given by

$$x_a^0(\delta,\varepsilon,p) := \arg\min_{x^0}\{f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} P_a^s(\varepsilon,x^0,p) + \delta\phi_a^{0,\mu}(x^0,p) : B_a^0(p)x^0 = b_a^0(p)\}. \tag{4.26}$$

For $s = 1,\dots,S$, the approximations of the second stage decisions are given by the composition

$$x_a^s(\varepsilon,p) := x_a^s(\varepsilon,x_a^0(\varepsilon,\varepsilon,p),p).$$

Note that we take $\delta = \varepsilon$ above for simplicity. The whole vector $x_a^\varepsilon(p)$ is thought of as the concatenation

$$x_a^\varepsilon(p) := (x_a^0(\varepsilon,p),x_a^1(\varepsilon,p),\dots,x_a^S(\varepsilon,p)).$$

If $\mu > 0$ and the functions defining the agents' problems are sufficiently smooth, the regularized solution mappings $x_a^s(\varepsilon,p)$ above are smooth. The numerical issue is that to compute their derivatives based on formulas (4.14), we need higher-order derivatives of the problem's data.

Convergence of the joint agent-wise and scenario-wise decomposition is still based on proving that property (4.3) holds. For this purpose, denote by $Q_a^0(p)$ the objective function of problem (4.22), by $Q_a^0(\varepsilon,p)$ the objective function of problem (4.25), and by $Q_a^0(\delta,\varepsilon,p)$ the objective function of problem (4.26). Inequality (4.9) applied to the scenario subproblems yields, for $s = 1,\dots,S$,

$$Q_a^s(p,x^0) \leq Q_a^s(\varepsilon,p,x^0) \leq Q_a^s(p,x^0) + \varepsilon m_a^s + \varepsilon\frac{\mu}{2}\|x_a^s(x^0,p)\|^2 \quad \text{for all}\, p\in p, x^0 \in X_a^0(p).$$

Then, because (4.11) is a blanket assumption if $\mu > 0$, we can use (4.10) to find $K > 0$ such that for all $p \in \Pi$ and $x^0 \in X_a^0(p)$ we have

$$f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} Q_a^s(x^0,p) \leq f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} Q_a^s(\varepsilon,x^0,p) \leq f_a^0(x^0,p) + \frac{1}{|\mathscr{S}|}\sum_{s\in\mathscr{S}} Q_a^s(x^0,p) + \varepsilon K. \tag{4.27}$$

Note that we used $Q_a^s(\varepsilon, x^0, p)$ in (4.27). To take into account $P_a^s(\varepsilon, x^0, p)$, we have (see [BSS20, Lemma 3]) to recall that for any $\kappa > 0$ it holds that $|Q_a^s(\varepsilon, x^0, p) - P_a^s(\varepsilon, x^0, p)| \leq \kappa$ for all $\varepsilon > 0$ small enough, uniformly on $x^0$ and $p$, since we are assuming uniformly bounded feasible sets. Then, for $\varepsilon = \varepsilon(\kappa) > 0$ small enough we have that

$$\begin{aligned} f_a^0(x^0, p) + \frac{1}{|\mathscr{S}|} \sum_{s \in \mathscr{S}} Q_a^s(x^0, p) - \kappa &\leq f_a^0(x^0, p) + \frac{1}{|\mathscr{S}|} \sum_{s \in \mathscr{S}} P_a^s(\varepsilon, x^0, p) \\ &\leq f_a^0(x^0, p) + \frac{1}{|\mathscr{S}|} \sum_{s \in \mathscr{S}} Q_a^s(x^0, p) + \varepsilon K + \kappa. \end{aligned}$$

Taking the infimum on the last inequality over $x^0 \in X_a^0(p)$, we obtain that

$$Q_a^0(p) - \kappa \leq Q_a^0(\varepsilon, p) \leq Q_a^0(p) + \varepsilon K + \kappa \quad \text{for all}\, p \in \Pi.$$

Applying inequality (4.9) and condition (4.11) to problem (4.26), and possibly modifying $K > 0$, we obtain that

$$Q_a^0(\varepsilon, p) \leq Q_a^0(\delta, \varepsilon, p) \leq Q_a^0(\varepsilon, p) + \delta K \quad \text{for all}\, p \in \Pi.$$

The last inequality combined with (4.5.3) gives

$$Q_a^0(p) - \kappa \leq Q_a^0(\delta, \varepsilon, p) \leq Q_a^0(p) + \varepsilon K + \delta K + \kappa \quad \text{for all}\, p \in \Pi.$$

We now take limits on the last inequality in $\varepsilon$ and $\delta$ for a fixed $\kappa$. Then, we take let $\kappa \searrow 0$. It follows that

$$\lim_{\varepsilon, \delta \searrow 0, p' \in p, p' \to p} x_a^0(\delta, \varepsilon, p') = x_a^0(p) \quad \text{for all}\, p \in \Pi.$$

Therefore, we also have that

$$\lim_{\varepsilon \searrow 0, p' \in p, p' \to p} x_a^s(\varepsilon, p') = x_a^s(p) \quad \text{for all}\, p \in \Pi.$$

We conclude that property (4.3) holds, which ensures convergence of the presented agent-wise and scenario-wise decomposition scheme.

## 4.6 Concluding remarks

We finish commenting on some issues related to gradient consistency, decomposition, and risk.

Gradient consistency for the smoothing technique in question was proven in [BSS20] in the case when the value function is convex. For nonconvex value functions, the issue is considerably more involved and still open. But under second-order sufficient conditions for the subproblems, classical statements can be made, because the nonregularized solution mappings are smooth themselves. However, this is not our focus indeed and we do not assume second-order conditions or strict complementarity.

Augmented Lagrangian methods for usual optimization problems with general lower-level constraints are considered in [And+08], and generalizations of such methods for GNEPs in [KS16]. The augmented Lagrangian method in [Sch12] solves essentially the same WEP as we deal with here, and solution times much better than in [DJW17] are reported. One issue with these methods though, is that decomposition is not possible because of the quadratic penalizations of the constraints. For instance, in [Sch12] experiments with up to thirty agents only are reported. With our method, subject to memory limitations and distributed computing capabilities, in principle we can solve problems with as many agents as required. The same holds for the number of scenarios. Decomposition techniques are important for this type of problems, because the number of variables for an equivalent formulation is the product of the number of agents, scenarios, and goods (thus, potentially huge).

Issues related to the existence of equilibrium prices for social welfare problems where agents solve risk-averse multistage stochastic optimization problems are considered in [PF18], and their relation to the classical first and second welfare theorems from economics are explored under some circumstances. This problem class is the same that we deal with here, the difference being that we aim to compute efficiently the best price possible according to the function $F$ above, independently whether this is an equilibrium price or not. Following [PF18], we could consider a risk measure for the stochastic WEP, proceeding as in in [BSS20], with the decisions of the agents that go in (4.2) being risk-averse. Therefore, conceptually, there is no difference and the algorithm we present here remains essentially the same, with the same decomposition properties. For instance, if $F$ stands for a measure of excess supply like in our numerical section, an equilibrium price $p^*$ could be characterized by $F(p^*) := F(x_{\mathscr{A}}(p^*)) = 0$. On the other hand, if with risk measures and incomplete risk-markets it becomes impossible to perfectly balance supply and demand, the best $p^*$ would be such that $F(p^*) > 0$. The non-existence of an equilibrium price is not an impediment to run a minimization procedure to find a price with a smaller associated objective function value (or with a smaller clearing) than what we currently have (the starting guess).

# Part II

# Free Floating Cuts: An Extension of Benders Cuts

# Chapter 5

# Profit Sharing Mechanisms in Multi-Owned Cascaded Hydro Systems

We consider the optimal management of hydropower generated by a cascade of three interconnected reservoirs owned by different agents. In this setting, water availability at the downhill reservoirs depends on decisions taken by the agents upstream. This creates an opportunity for the hydroplant at the top to withhold water and take advantage of situations with higher selling prices, which makes the overall decisions of the agents deviate from what can be considered best for the cascade as whole. In order to mitigate the market power of the hydroplant uphill, we propose a mechanism to enforce some collaborative behavior among the agents. This is achieved by agents transferring upstream fractions of their profit in exchange for water released from the top. The corresponding mathematical models are trilevel deterministic and trilevel stochastic linear programming problems, with uncertainty in prices and streamflows (exogenous inflows). For the stochastic variants, analyzed both in two- and multi-stage formulations, we propose new solution methods, extending to the trilevel nested setting the well-known L-Shaped and Stochastic Dual Dynamic Programming methods. A successful implementation of the later depends on certain floating cuts, that represent symbolically Benders-like linearizations. Convergence properties are discussed for some of the procedures. Numerical experiments confirm the interest of the approach, because with the proposed mechanism the top owner as well as the downhill hydroplants earn more money than when acting in an individualistic manner.

## 5.1 Introduction

Hydroelectricity is the most widely used renewable energy worldwide (60% in 2020, according to [IRE20]). In countries like Brazil, Canada, China and Norway hydrogeneration largely dominates the power mix. In France, hydropower is only second to nuclear energy, roughly representing 10% of the total generation. The value of hydropower for energy systems relies on several important features of the technology. To start with, hydroelectricity is renewable and storable, as potential power can be stored in the form of water in the reservoirs. Storage is one of the biggest challenges in the transition to sustainable electricity systems, and reservoirs offer volumes in storage capacity still unthinkable, at this time, in terms of batteries. According to the estimations [Int18], reservoirs represented 94% of the installed global energy storage capacity worldwide in 2018. In addition, triggering hydraulic generation is fast: it suffices to release water from the uphill reservoirs for the whole cascade to start generating.

All these properties clearly make hydroelectricity very attractive and, more importantly, extremely useful to counter some downsides of systems dealing with both flexible generation and flexible consumption, as customary nowadays. Hydropower is a crucial tool for network operators to succeed compensating uncertainty and intermittency of renewable energy sources with fluctuations of prices and demand, in a manner that guarantees the stability and safety of the electricity system.

The issue of optimal management of hydro-dominated power systems has been vastly studied, considering short, mid, and long planning times; see [GMH10]. How to properly manage a cascaded hydrosystem, however, still remains a challenge. A large portion of the literature focuses on a price-taker situation in which hydroplant owners cannot influence spot prices; see, for example, the extensive review [TD17]. For a cascade of hydroplants belonging to different owners, we also adopt a price-taking setting (prices are given by scenarios, see Figure 5.5). Our contribution is to propose to manage the cascade in a manner that mitigates market power due to water withheld upstream. For a cascade like the one with three hydroplants represented in Figure 5.1, water availability of the middle and bottom reservoirs highly depends on decisions taken at the levels upstream. If the top hydroplant plays opportunistically and withholds water, downstream owners will earn less profit. In our numerical experiments we observed that an individualistic management of the cascade can reach extremes in which, for the top hydroplant to increase profit in 1%, the other levels must lose up to 10% of their profit. The assessment and mitigation of market power in decentralized hydro-thermal systems is discussed in [KBP01]. A formulation solving an equilibrium problem with equilibrium constraints was proposed in [CA13]. Recently, a duopoly formulation of competing hydroplants in the same cascade is analyzed in [MM20]. While market power is clearly an issue for the deployment of decentralized hydro systems, proposals of mitigation strategies are rarely found in the literature, with the exception of the work on the Brazilian wholesale water market [Kel99], a market which, however, has not been implemented in practice.

The mechanism for market power mitigation we investigate in this work is informally described by the left-hand side arrows in Figure 5.1. Formally, we deal with a trilevel optimization problem with decisions intertwined in a complex pattern. For the optimization problem at a given level, variables of the uphill problem appear in right-hand side of the constraints, and downhill variables appear in the objective functions; see, for instance, the deterministic formulation in (5.2) below. While the former dependence is a natural consequence of the cascade, because water released upstream increases the reservoir volume downhill, the latter is due to the mitigation mechanism, with downhill levels paying to the upstream plants part of the profit they have that exceeds the individualistic profits.

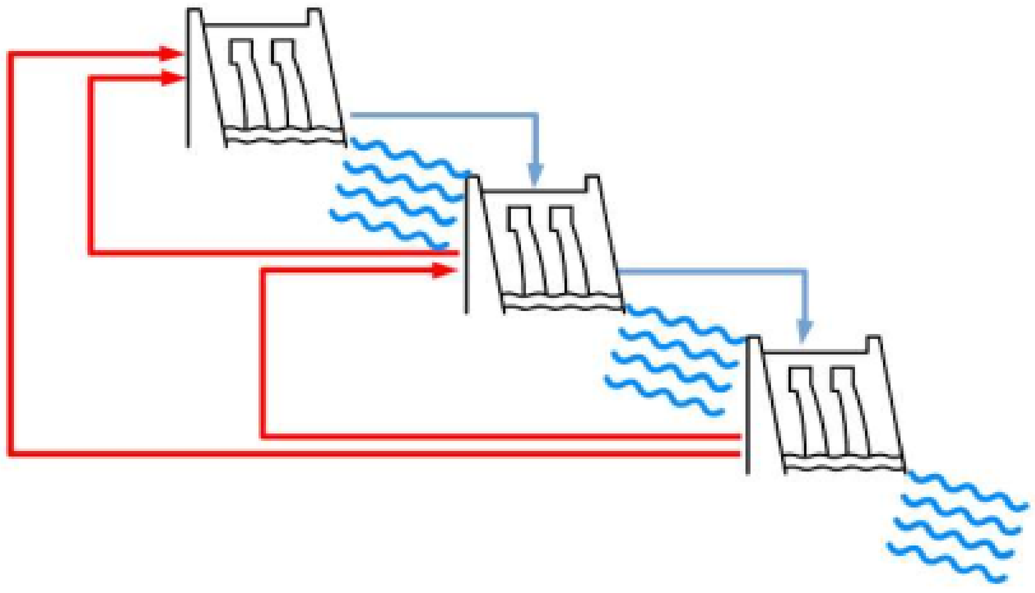

Figure 5.1: Water released uphill changes the right-hand side (RHS) in the water balance constraints of the power plant that is downhill. When water is withheld at some level, the profit of utilities that are downstream is diminished because they generate less power. To encourage water releases, downhill power plants transfer a fraction of their profit to utilities that are upstream. In the diagram, the utility in the top receives percentages $\tau_{2\to1}$ and $\tau_{3\to1}$ of the profit made by hydroplants in the middle and in the bottom, respectively. The latter also pays a fraction $\tau_{3\to2}$ of its profit to the middle power plant, that is immediately upstream. The arrows show how decisions are intertwined in the trilevel setting. Let $x_l$ denote the decision taken when optimizing the generation of utility $l$. On the left, transferring profit upwards (left red arrows) goes from level $l+1$ to level $l$: the $l$th objective function includes terms involving downhill decisions $x_{l+1}$ (for $l = 1$, the term includes both $x_2$ and $x_3$). On the right, the terms $\mathrm{RHS}(x_l)$ (right blue arrows) change the feasible set of the $(l+1)$th problem, and the dependence is reversed, going from level $l$ to level $l+1$.

Figure 5.1 considers three independent hydrogenerators (top, middle, bottom) optimizing their profit according to some mathematical model. Their decisions are summarized by the generated power, a function of the reservoir volume, which in turns depends on the water released upstream. In our model, all relations are assumed to be linear, including the so-called efficiency function transforming water into energy (see [Cat+09; CPM10] for nonlinear efficiency functions). Each generator aims at maximizing the profit resulting from selling the generated power at prices that can be random, while satisfying constraints, including bounds on the volumes. We consider a single objective function, for multi-objective formulations the reader is referred to [LQ15; Li+18]. Since streamflows to each reservoir are random, and prices increase if demand is high or if reservoir volumes are low, generators sometimes store water even in wet periods so that in dry periods, with higher prices, the released water generates power that yields more gains. In our proposal, agents in the middle and bottom levels transfer a fraction of their profit uphill, as an incentive to release water downstream. By this sharing mechanism, the profits of all the agents can only increase, when compared to the individualistic setting (the transfer is a fraction of what exceeds the individualistic profit). The numerical results reported in Figures 5.2 and 5.7 confirm that the top hydroplant as well as both downhill owners earn more money with our approach.

The mathematical optimization problem has three levels of nested stochastic linear programs, for which we discuss variants with uncertainty unveiling in two-stage and multi-stage manners. All variables are continuous, to ensure convergence of the solution procedure described in Algorithm 1 below, where certain bilevel subproblems are reformulated by means of primal-dual relations that are necessary and sufficient optimality conditions in the considered setting. The different models and solution methods are given in Sections 5.2, 5.3, and 5.4. Section 5.2 presents the deterministic formulation of the trilevel model and introduces the proposed mechanism of transfer of profits. The solution approach, which defines cuts for the value functions of the successive levels as in a Benders' decomposition, is assessed on a cascade with three reservoirs like in Figure 5.1. We employ a simple, yet realistic, system that confirms the interest of transferring profit as a device to increase the gains of the cascade as a whole. The stochastic case is addressed in the next two sections. We first study, in Section 5.3, the individualistic model, which for the trilevel optimization problem amounts to having dependencies only on the feasible sets (in Figure 5.1 there are no arrows on the left, going upwards). We extend the L-shaped method [SW69] to a "cascaded" variant that properly deals with a sequence of three nested two-stage stochastic linear programs. To solve the more complex multi-stage model (still the individualistic formulation) we introduce a nested variant of the Stochastic Dual Dynamic Programming (SDDP) method [PP91] that deals simultaneously with the three levels. Section 5.3 ends with numerical results comparing the output of the individualistic model with a joint management of the cascade, both modeled in a multi-stage setting. The final Section 5.4 deals with the stochastic trilevel model with profit sharing. The resulting "cascaded" SDDP method is rather involved, its successful implementation relies on a new concept, that we named floating cut, described in in Section 5.4.1. A floating cut is a symbolic representation of the Benders-like linearizations employed by the SDDP method. By this token, information is suitably transported between levels and the important properties of lower bounding and consistency are preserved throughout the iterative process. Regarding convergence, the methodology is shown in Theorem 5.2.1 to have finite termination in the deterministic case and also in the individualistic two-stage variant. For the remaining cases, our approach provides a reasonable heuristic whose good performance is confirmed by the numerical experiments carried on for this work. As an additional check of goodness, in the benchmark we compare in Subsection 5.4.3 the deterministic setting with the stochastic one, verifying that the output is consistent, with the respective results being similar to some extent.

## 5.2 Deterministic trilevel problem

It is convenient to cast the trilevel problem in an abstract format, that is particularized later on for the application. In order to gradually introduce the notation and setting, we consider first that there is no uncertainty and give the deterministic mathematical formulation, as well as a solution procedure for the resulting trilevel problem.

### 5.2.1 Problem formulation

When considered independently, the optimization problem at level $l$ is a linear programming problem of the form

$$\min_{x_l \geq 0} \quad f_l^\top x_l \quad \text{s.t.} \quad A_l x_l = a_l \,,$$

where the decision variable $x_l$ has components such as turbined outflows and spillage, the $l$th reservoir volume, and the power generation. Water balance and capacity constraints are abstractly described by the matrix $A_l$ and the vector $a_l$, of suitable dimensions. Finally, the negative of the vector $f_l$ in the objective function represents the unit profit made at level $l$.

To formalize the modifications due to the cascaded setting and formulate the trilevel problem, we start with the bottom level, $l = 3$. The effect that a release of water uphill (represented by $x_2$) has on the optimization problem downhill is measured by the following value function

$$v_3(x_2) := \min_{x_3 \geq 0} \quad f_3^\top x_3 \quad \text{s.t.} \quad A_3 x_3 = a_3 - B_3 x_2 \,.$$

As illustrated by the right arrows in Figure 5.1, changing $a_3$ to $a_3 - B_3 x_2$ relates levels $l-1$ and $l$ through the right-hand side term (RHS).

Let us now define the problem for $l = 2$. The mitigation strategy proposes to transfer fractions $(\tau_{3\to2}, \tau_{3\to1})$ of the profit of generator of level 3 upstreadm, to levels 2 and 1, respectively. Regarding our level of interest, $l = 2$, this establishes the link in the objective function, relating levels $l$ and $l+1$, represented by the left arrows in Figure 5.1. Specifically, because the term $-\tau_{3\to2}v_3(x_2)$ represents a gain, it enters in the optimization problem with a negative sign. Accordingly, the modified objective function for level 2 is the cost

$$f_2^\top x_2 + \tau_{3\to2}v_3(x_2)$$

and, hence, at level $l = 2$ the value function of interest is

$$v_2^{3\to2}(x_1) := \min_{x_2\geq0} \quad f_2^\top x_2 + \tau_{3\to2}v_3(x_2) \quad \text{s.t.} \quad A_2x_2 = a_2 - B_2x_1\,.$$

A similar reasoning yields the objective function at the top level ($l = 1$):

$$f_1^\top x_1 + \tau_{2\to1}v_2^{3\to2}(x_1) + \tau_{3\to1}v_3(x_2)\,, \tag{5.1}$$

where the decision variable $x_2$ is the decision that would be taken at $l = 2$ after $x_1$ is taken at $l = 1$. In this nested optimization problem, the downhill value function $v_3(\cdot)$ is polyhedral and convex on $x_2$ and the function $v_2^{3\to2}(\cdot)$ is convex on $x_1$ since $\tau_{3\to2} \geq 0$, see [HUL93, Cor.IV.2.4.3]. Then, the function (5.1) is convex as a sum of convex functions whenever $\tau_{2\to1}, \tau_{3\to1} \geq 0$. Non-convexity arises in the problem because of the nexted formulation: the value of $x_2$ that enters (5.1) has to solve the optimization problem at level 2, as made explicit in the formulation below by the notation $x_2^{\mathtt{C}}$.

Summing up, in an optimistic setting, the trilevel problem amounts to finding

$$x^{\mathtt{C}} = (x_1^{\mathtt{C}}, x_2^{\mathtt{C}}, x_3^{\mathtt{C}}) \text{ such that } \quad x_1^{\mathtt{C}} \text{ solves} \begin{cases} \min\limits_{x_1\geq0} & f_1^\top x_1 + \tau_{2\to1}v_2^{3\to2}(x_1) + \tau_{3\to1}v_3(x_2^{\mathtt{C}}) \\ \text{s.t.} & A_1x_1 = a_1 \end{cases}$$
$$x_2^{\mathtt{C}} \text{ solves} \begin{cases} \min\limits_{x_2\geq0} & f_2^\top x_2 + \tau_{3\to2}v_3(x_2) \\ \text{s.t.} & A_2x_2 = a_2 - B_2x_1^{\mathtt{C}} \end{cases} \tag{5.2}$$
$$x_3^{\mathtt{C}} \text{ solves} \begin{cases} \min\limits_{x_3\geq0} & f_3^\top x_3 \\ \text{s.t.} & A_3x_3 = a_3 - B_3x_2^{\mathtt{C}}\,. \end{cases}$$

In our numerical experiments, we compare the output $x^{\mathtt{C}}$ of (5.2) with two other policies. First, a socially optimal policy that minimizes the aggregate total cost for the society, while being feasible for all agents:

$$\text{find } x^{\mathtt{S}} = (x_1^{\mathtt{S}}, x_2^{\mathtt{S}}, x_3^{\mathtt{S}}) \text{ solving} \begin{cases} \min\limits_{(x_1,x_2,x_3)\geq0} & f_1^\top x_1 + f_2^\top x_2 + f_3^\top x_3 \\ \text{s.t.} & A_1x_1 = a_1 \\ & A_2x_2 = a_2 - B_2x_1 \\ & A_3x_3 = a_3 - B_3x_2\,. \end{cases} \tag{5.3}$$

Second, individualistic policies that are optimally taken in a sequential manner along the cascade:

$$\text{find } x^{\mathtt{I}} = (x_1^{\mathtt{I}}, x_2^{\mathtt{I}}, x_3^{\mathtt{I}}) \text{ solving}$$
$$\begin{cases} \min\limits_{x_1\geq0} & f_1^\top x_1 \\ \text{s.t.} & A_1x_1 = a_1 \end{cases} \quad \begin{cases} \min\limits_{x_2\geq0} & f_2^\top x_2 \\ \text{s.t.} & A_2x_2 = a_2 - B_2x_1^{\mathtt{I}} \end{cases} \quad \begin{cases} \min\limits_{x_3\geq0} & f_3^\top x_3 \\ \text{s.t.} & A_3x_3 = a_3 - B_3x_2^{\mathtt{I}}\,. \end{cases} \tag{5.4}$$

Since the RHS dependency may result in empty feasible sets, the policies might not be well defined in some configurations. We assume this is not the case, because feasibility can be ensured for our application by introducing slack variables representing energy deficits.

Notice that the individualistic point $(x_1^{\mathtt{I}}, x_2^{\mathtt{I}}, x_3^{\mathtt{I}})$ is feasible for the social problem (5.3). As a result,

$$v^{\mathtt{S}} := f_1^\top x_1^{\mathtt{S}} + f_2^\top x_2^{\mathtt{S}} + f_3^\top x_3^{\mathtt{S}} \leq v^{\mathtt{I}} := f_1^\top x_1^{\mathtt{I}} + f_2^\top x_2^{\mathtt{I}} + f_3^\top x_3^{\mathtt{I}}\,. \tag{5.5}$$

Policy makers are often concerned with the "social utility" of an energy market. The gap $v^{\mathtt{I}} - v^{\mathtt{S}} \geq 0$ gives a measure of the social dis-utility of the individualistic policy. We shall see that with our mechanism (5.2), of profit sharing, the social dis-utility of the market is decreased.

### 5.2.2 Solution procedure

To solve the trilevel problem (5.2) algorithmically we exploit the convexity of the value functions $v_2^{3\to 2}$ and $v_3$ in an iterative form. Letting $k$ represent the current iteration, the respective cutting-plane approximations, denoted by $v_2^k$ and $v_3^k$, are computed in a straightforward way. More precisely, for any given $x_2^k$, when solving the linear programming problem

$$v_3(x_2^k) = \min_{x_3 \geq 0} \quad f_3^\top x_3 \quad \text{s.t.} \quad A_3 x_3 = a_3 - B_3 x_2^k \tag{5.6}$$

the equality constraints vector of multipliers $\lambda_3^k$ is available. Convexity ensures the following inequality for the associated linearization:

$$\ell_3^k(x_2) := v_3(x_2^k) + (\lambda_3^k)^\top B_3(x_2 - x_2^k) \leq v_3(x_2)\,.$$

The cutting-plane approximation for $v_3$ is then

$$v_3^k(x_2) := \max_{j=1,\dots,k} \ell_3^j(x_2) \leq v_3(x_2)\,.$$

Having this piecewise affine function, an analogous mechanism can be put in place for defining a cutting-plane model function that bounds $v_2^{3\to 2}$ from below. Since at level $l=2$ the objective function involves the (unknown) value function $v_3$, linearizations are computed for an approximate function $v_2^k$, obtained when replacing $v_3$ by its cutting-plane model $v_3^k$:

$$v_2^k(x_1) := \min_{r_2, x_2 \geq 0} \quad f_2^\top x_2 + \tau_{3\to 2} r_2 \quad \text{s.t.} \quad A_2 x_2 = a_2 - B_2 x_1\,, \quad r_2 \geq \ell_3^j(x_2)\,, j = 1, \dots, k\,. \tag{5.7}$$

Solving this linear program with $x_1 = x_1^k$ gives a vector of multipliers $\lambda_2^k$ for the equality constraints and, therefore,

$$\ell_2^k(x_1) := v_2^k(x_1^k) + (\lambda_2^k)^\top B_2(x_1 - x_1^k) \leq v_2^k(x_1)\,.$$

The linearization also bounds the function $v_2^{3\to 2}$ from below, because, by construction, the additional scalar variable in (5.7) satisfies $r_2 = v_3^k(x_2) \leq v_3(x_2)$, which implies that $v_2^k(x_1) \leq v_2^{3\to 2}(x_1)$ for all $x_1$, as claimed. Notwithstanding, it is important to keep in mind that the objective function in (5.7) is different from the one written for $l=2$ in the trilevel problem (5.2). Specifically,

$$\text{Algorithm 1 replaces } f_2^\top x_2 + \tau_{3\to 2} v_3(x_2) \text{ from (5.2) by } f_2^\top x_2 + \tau_{3\to 2} v_3^k(x_2) \text{ using the function in (5.7)}. \tag{5.8}$$

Such a replacement, necessary for the numerical implementation, has an impact on the convergence properties of the procedure; see Theorem 5.2.1 and the nearby comments.

In our procedure, the cutting-plane estimations of the value functions approximate the trilevel setting (5.2) by a sequence of bilevel linear problems

$$\begin{aligned} &\text{Given } v_2^k, v_3^k, \text{find } (x_1^k, x_2^k) \text{ such that} \quad x_1^k \text{ solves } \left\{ \begin{array}{ll} \min\limits_{x_1 \geq 0} & f_1^\top x_1 + \tau_{2\to 1} v_2^k(x_1) + \tau_{3\to 1} v_3^k(x_2^k) \\ \text{s.t.} & A_1 x_1 = a_1 \end{array} \right. \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad x_2^k \text{ solves } \left\{ \begin{array}{ll} \min\limits_{x_2 \geq 0} & f_2^\top x_2 + \tau_{3\to 2} v_3^k(x_2) \\ \text{s.t.} & A_2 x_2 = a_2 - B_2 x_1^k\,. \end{array} \right. \end{aligned} \tag{5.9}$$

These bilevel linear problems are solved as follows. First, the lower level problem is replaced by its equivalent optimality conditions, involving primal and dual feasibility and strong duality. The latter equality constraint introduces bilinear terms that are reformulated along the lines of McCormick cuts, using binary variables and a "big M" approach, i.e., choosing large constants that can make the problem ill-conditioned, due to bad scaling [Dem02, Ch. 5]. The resulting reformulation is a mixed integer linear programming problem. It is well known that success in the reformulation is driven by a sound choice of the aforementioned large constants.

Algorithm 1 gives the resulting iterative process in full detail.

Since we are dealing with polyhedral value functions, they are defined by a finite number of cutting-planes; in particular, by a finite number of subgradient values. The latter are the optimal Lagrange multipliers associated to the equality constraints in the linear programs (5.6) and (5.7). While the primal-dual solutions of these problems may not be unique in general, many (if not most) LP solvers compute *specific* solutions, which makes their

**Algorithm 1** SOLUTION PROCEDURE FOR TRILEVEL PROBLEM WITH SHARING MECHANISM. DETERMINISTIC CASE

```
Initialization. Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$.
Set $k = 1$, $v_2^k(\cdot) \equiv -M$ and $v_3^k(\cdot) \equiv -M$.
Iterates for levels 1 and 2. Compute $(x_1^k, x_2^k)$ solving the bilevel linear problem (5.9).
Iterate and linearization for level 3. Compute the primal-dual solution $(x_3^k, \lambda_3^k)$ to the linear programming problem
(5.6) and the linearization
    $\ell_3^{k+1}(x_2) = v_3(x_2^k) + (\lambda_3^k)^\top B_3(x_2 - x_2^k)$.        (5.10)
Let $v_3^{k+1}(x_2) = \max\{v_3^k(x_2), \ell_3^{k+1}(x_2)\}$.
Linearization for level 2. Compute the primal-dual solution $(\hat{x}_2^k, \hat{r}_2^k, \lambda_2^k)$ to the linear programming problem
    $v_2^{k+1}(x_1^k) := \min_{r_2, x_2 \geq 0} f_2^\top x_2 + \tau_{3\to2} r_2$  s.t.  $A_2 x_2 = a_2 - B_2 x_1^k$,  $r_2 \geq \ell_3^j(x_2), j = 1, \ldots, k+1$.
and $\ell_2^{k+1}(x_1) = v_2^{k+1}(x_1^k) + (\lambda_2^k)^\top B_2(x_1 - x_1^k)$. Let $v_2^{k+1}(x_1) = \max\{v_2^k(x_1), \ell_2^{k+1}(x_1)\}$.
Stopping test and loop. Stop if both the gaps of levels 2 and 3 are small, e.g., if
    $v_3(x_2^k) - v_3^k(x_2^k) \leq \varepsilon$  and  $v_2^{k+1}(x_1^k) - v_2^k(x_1^k) \leq \varepsilon$.
Otherwise, set $k = k+1$ and go back to Iterates for levels 1 and 2.        □
```

selection finite. Moreover, these solutions correspond precisely to the cutting-planes whose maximum defines the polyhedral value functions exactly. For example, if the LP solver employed by Algorithm 1 computes basic optimal solutions (vertices), as the simplex method does, the number of optimal multipliers that the solver can return for (5.6) and (5.7) can only be finite; see, e.g., [OSS11, Prop. 4.1] (somewhat related considerations in a different context can also be found in [DSS09, p. 287]). Interior point methods which compute certain (unique) centered duals, see [ADCM91], can also return only a finite number of optimal solutions. The property of the LP solver producing a finite number of solutions, ensures finite termination of our algorithm; this can be seen, for example, from the argument in [Ber95, Prop. 6.3.2] for the cutting-plane method applied to a max-function.

Notwithstanding, a word of caution is in order, regarding the final iterate computed by Algorithm 1: it may not solve the trilevel problem (5.2), because the replacement (5.8) does not ensure the equality $v_3^k(x_2) = v_3(x_2)$ for all $x_2$. However, in practice, the output of Algorithm 1 provides at least a good estimate for practical purposes, as the inequality $v_3(x_2) \geq v_3^k(x_2)$ does hold everywhere. The next statement summarizes the convergence properties of Algorithm 1.

**Theorem 5.2.1** (Finite termination of Algorithm 1)**.** *Consider Algorithm 1 with $\varepsilon = 0$, and let the linear programming solver therein be such that applied to* (5.6) *and* (5.7)*, only a finite number of primal-dual solutions can be returned by the solver. Then the following holds.*

*(i) The algorithm terminates at some iteration k satisfying*

$$v_3(x_2^k) = v_3^k(x_2^k) \quad \text{and} \quad v_2^{k+1}(x_1^k) = v_2^k(x_1^k).$$

*(ii) If $\tau_{3\to1} = 0$, the iterate $x^k := (x_1^k, x_2^k, x_3^k)$ solves* (5.2).

*(iii) If $v_3^k(x_2) = v_3(x_2)$ for all $x_2 \geq 0$ such that $A_2x_2 = a_2 - B_2x_1$ for all $x_1$, then $x^k$ solves* (5.2).

*Proof.* (i) At some iteration $k$, the primal-dual points $(x_2^k, \lambda_2^k)$ and $(x_3^k, \lambda_3^k)$ eventually coincide with some other pairs computed previously, because the value functions are polyhedral and we assume that the linear solver produces only finitely many pairs of primal and dual solutions. By construction, the inequalities $v_3^{k+1}(x_2^k) \geq v_3^k(x_2^k)$ and $v_2^{k+1}(x_1^k) \geq v_2^k(x_1^k)$ hold. Since the point already defined linearizations at some past iteration, at the end of the $k$th iteration, by definition of $v_3^k$ and $v_2^k$, we would have that $v_3^k(x_2^k) \geq v_3^{k+1}(x_2^k)$ and, analogously, $v_2^k(x_1^k) \geq v_2^{k+1}(x_1^k)$. The conclusion follows.

(ii) If $\tau_{3\to1} = 0$, the problematic constraint on $x_2$ in problem (5.9) can be disregarded because we are assuming solvability. Then, convergence follows from standard properties of cutting-plane iterations.

(iii) When the algorithm stops, if $v_3^k(x_2) = v_3(x_2)$ for all $x_2$, then problem (5.9) represents perfectly problem (5.2). Then, $x^k$ solves (5.2). □

As a result, when Algorithm 1 stops, its output $x^k$ is feasible for the original problem (5.2), and the pair

$(x_1^k, x_2^k)$ solves globally the approximate problem

$$\text{Given } v_3^k, \text{find } (x_1^k, x_2^k) \text{ such that} \quad x_1^k \text{ solves} \begin{cases} \min\limits_{x_1 \geq 0} & f_1^\top x_1 + \tau_{2\to 1} v_2^{3\to 2}(x_1) + \tau_{3\to 1} v_3^k(x_2^k) \\ \text{s.t.} & A_1 x_1 = a_1 \end{cases}$$

$$x_2^k \text{ solves} \begin{cases} \min\limits_{x_2 \geq 0} & f_2^\top x_2 + \tau_{3\to 2} v_3^k(x_2) \\ \text{s.t.} & A_2 x_2 = a_2 - B_2 x_1^k. \end{cases}$$

In view of these properties, Algorithm 1 can be thought of as being a Phase I procedure. If desired, a subsequent Phase II mechanism can be put in place to generate cuts that may be missing, therefore guaranteeing that $v_3^k(x_2) = v_3(x_2)$ for all $x_2$. By Theorem 5.2.1(iii), this triggers optimality of $x^k$ for (5.2).

## 5.2.3 Numerical assessment

The experiments are performed for the cascade of three hydropower plants represented in Figure 5.1 in the introduction. Over an horizon with $t = 1, \ldots, \mathtt{T}$ time periods, and for each hydroplant $l$,

- the data is: the efficiency factor $\Phi_l$, the price of energy at time $t$ denoted by $\Pi^t$, the exogenous water inflow $A_l^t$, the maximum and minimum reservoir volumes $\overline{V}_l$ and $\underline{V}_l$, and the maximum turbine outflow $\overline{U}_l$.
- The variables are: the turbined outflow $u_l^t$, the reservoir volume $v_l^t$, and the spillage $w_l^t$.

The optimization problem solved by the hydropower plant $l$ is shown below, where decisions of other levels are shown in bold and not present if $l - 1 < 1$:

$$\begin{cases} \max\limits_{(u,v,w)\geq 0} & \Phi_l \sum_{t=1}^{\mathtt{T}} \Pi^t u_l^t \\ \text{s.t.} & v_l^{t+1} = v_l^t + \eta(-u_l^t - w_l^t + A_l^t + u_{l-1}^t + w_{l-1}^t), \quad t = 0, \ldots, \mathtt{T}-1 \\ & \eta(u_l^{\mathtt{T}} + w_l^{\mathtt{T}}) \leq v_l^{\mathtt{T}} + \eta A_l^{\mathtt{T}}, \quad t = 1, \ldots, \mathtt{T} \\ & \eta u_l^t \leq v_l^t - \underline{V}_l, \quad t = 1, \ldots, \mathtt{T} \\ & v_l^t \leq \overline{V}_l, \quad u_l^t \leq \overline{U}_l, \quad t = 1, \ldots, \mathtt{T}, \end{cases} \tag{5.11}$$

where $\eta$ is the amount of unit time per time period. The meaning of constraints in (5.11) is standard, starting with the water balance in the reservoir, and inequalities to keep the outflow (turbined and spilt) within the capacity of the reservoir. Nonnegativity for the turbined outflow $u_l^t$ rules out any pumping mechanism for simplicity (negative values could be handled as well).

As stated, problem (5.11) suffers from the end-of-period effect, that depletes reservoirs to maximize the profit of each agent. A final target volume could be incorporated to address this issue, but here we do not include that constraint, and focus on the stylized model (5.11). Additionally, to guarantee feasibility without resorting to deficit-related slack variables, neither the outflow nor the spillage are bounded above in (5.11). Finally, with respect to the abstract notation, we have the relations

$$x_l := \left((u_l^t, v_l^t, w_l^t, \text{slacks}_l^t)_{t=1}^{\mathtt{T}}\right), f_l := -\Phi_l\left((\Pi^t, 0, 0, 0)_{t=1}^{\mathtt{T}}\right),$$

where the slack variables are used to rewrite the feasible set in standard linear programming form, with equality constraints only.

The data defining our toy problem, with a simplified cascade configuration, is meant to illustrate the features of interest, but should not be considered a realistic system. Prices and inflows are the mean of those considered in the stochastic setting and shown in Figure 5.5. The McCormick-like reformulation of the bilevel problems (5.9) uses values for "big M" in $[10^4, 10^5]$, tuned numerically. The remaining parameters are given in Table 5.1.

Table 5.1: Data for deterministic runs

| $l$ | $\Phi_l$ | $V_0$ | $\underline{V}_l$ | $\overline{V}_l$ | $\overline{U}_l$ |
|---|---|---|---|---|---|
| 1 | 1 | 0.24 | 0.16 | 1.60 | 0.80 |
| 2 | 1 | 0.15 | 0.10 | 1.00 | 0.50 |
| 3 | 1 | 0.24 | 0.16 | 1.60 | 0.80 |

The benchmark compares the output $x^{\mathrm{C}}$ of the profit-sharing mechanism (5.2) with the social and individualistic policies, $x^{\mathrm{S}}$ and $x^{\mathrm{I}}$ solving (5.3) and (5.4), respectively. The considered percentages for (5.2) are

$$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} \in \{0.02, 0.05, 0.10, 0.20, 0.30, 0.40\},$$

and additionally $\tau_{3\to1} = 0.00$, as in Theorem 5.2.1(ii).

All experiments were run on an Intel i7 1.90GHz machine, with Ubuntu 18.04.3 LTS, Julia 1.1.1[Bez+17] and CPLEX 12.10. Solving each deterministic variant took less than 20 seconds for $\varepsilon = 10^{-4}$.

The profit of each variant is then compared to that obtained with the social policy. Since the latter is $-f_l^\top x_l^{\mathrm{S}}$, the difference in profit with the individualistic policy is $-f_l^\top(x_l^{\mathrm{S}} - x_l^{\mathrm{I}})$ for $l = 1,2,3$. With the profit sharing mechanism, the difference in profit is

$$\begin{cases} -f_3^\top x_3^{\mathrm{S}} + (1-\tau_{3\to1}-\tau_{3\to2}) f_3^\top x_3^{\mathrm{C}} & \text{if } l=3 \\ -f_2^\top x_2^{\mathrm{S}} + (1-\tau_{2\to1})(f_2^\top x_2^{\mathrm{C}} + \tau_{3\to2} f_3^\top x_3^{\mathrm{C}}) & \text{if } l=2 \\ -f_1^\top x_1^{\mathrm{S}} + f_1^\top x_1^{\mathrm{C}} + \tau_{2\to1}(f_2^\top x_2^{\mathrm{C}} + \tau_{3\to2} f_3^\top x_3^{\mathrm{C}}) + \tau_{3\to1} f_3^\top x_3^{\mathrm{C}} & \text{if } l=1\,. \end{cases}$$

The individualistic profits $v^{\mathrm{I}}$ defined by (5.5) are useful to improve the quality of the output of the profit-sharing model. This is made clear by realizing that the policy remains unchanged if, for instance,

$$\text{at level } l=2, \text{ the transfer is } \tau_{3\to2}(v_3^{\mathrm{C}}(x_2) - \phi_l) \text{ for any benchmark value } \phi_l. \tag{5.12}$$

We exploit this degree of freedom to improve the quality of the results in the profit-sharing model. Specifically in our numerical experiments, we take $\phi_l = v_{l+1}^{\mathrm{I}}$, so that rewards that go uphill are only a fraction of the effective improvement of profit downstream. This re-scaling, transferring net margins with respect to the individualistic policy instead of gross values, has a significant impact in the numerical solution, at least with our data.

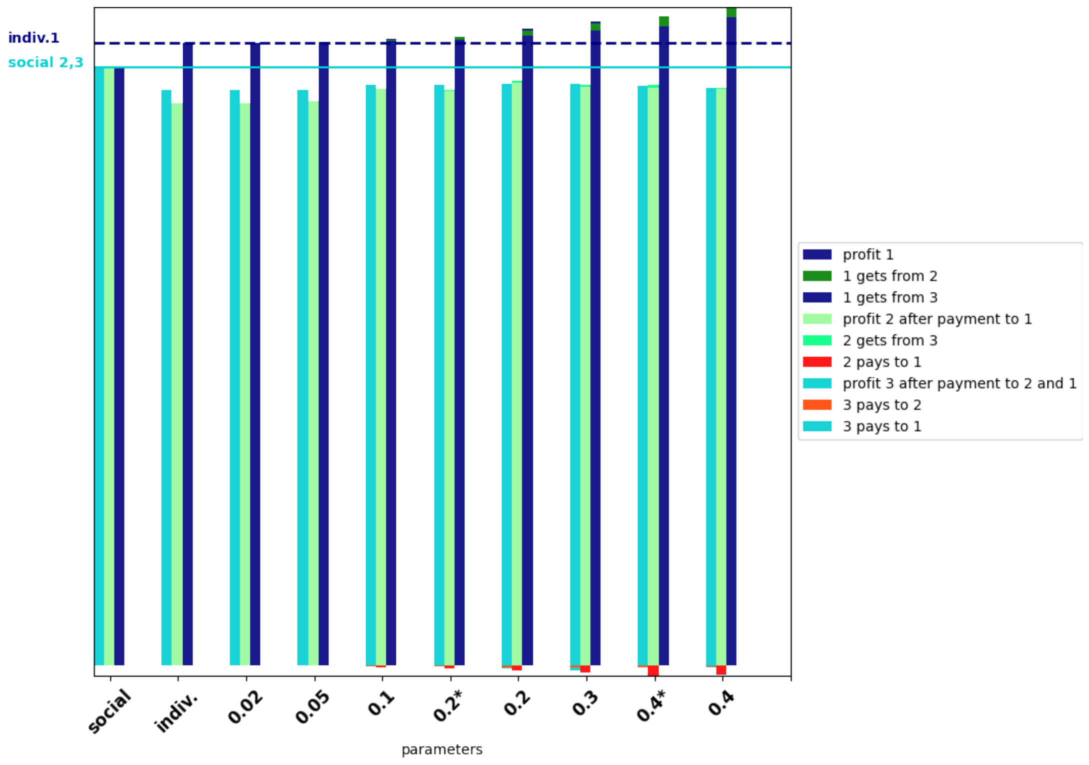

Figure 5.2: The profit-sharing mechanism is effective to recover more wealth from the cascade as a whole. With not too large transfers (the red bottom areas), when $\tau_{2\to1} = \tau_{3\to2} = 0.2$, level 1 makes more profit than in the individualistic setting, and levels 2 and 3 get the closest to the social policy. Larger transfers do not improve the situation for the downhill levels. When numbers are followed by a star, $\tau_{3\to1} = 0$, noting that tuning this parameter is more delicate, because there is an additional interplay within levels when $l = 3$ makes two transfers of profit.

Figure 5.2 reports the differences in profit, compared to the social one. For each level and run, profits are presented as bars proportional to the profit obtained with the social policy (in the first group of bars), assimilated to 100%.

The individualistic policy, reported in the second group of bars, results in a increase for level 1, but decreases the profit of both levels 2 and 3. A policy will be acceptable for level 1 only if the gain is at least the individualistic profit. This corresponds to the dashed horizontal line in the graph. By contrast, levels 2 and 3 prefer policies that drive their profit as close as possible to the social one, whose level is indicated by the solid horizontal line in the graph. The remaining groups of bars correspond to different configurations of the profit sharing mechanism. Numbers in the abscissa indicate the value that was taken for $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2}$, except when the number is followed by a star, in which case $\tau_{3\to1} = 0$, and $\tau_{2\to1} = \tau_{3\to2} =$ the displayed number. Red bars in the bottom represent the payments done from the downhill levels uphill (always as a percentage of the social profit). When there is more than one color in the top of a bar, the area illustrates a transfer of profit from downhill levels. This is perceptible for example in the column labeled 0.2, where the profit of level 2 is increased by a transfer from level 3, and level 1 profit gets transfers from both levels 2 and 3. For each level $l \in \{1,2,3\}$, the bars with profit represent the gain, net from payments uphill. Therefore, when stacking on top the transfer from levels below (in a different color in the figure), the top of the bar corresponds to the final profit of the considered policy. Numeric values for the transfers can be found in the Appendix A, Table A.1.

Figure 5.3 reports the water management for level $l$ in the $l$th row of plots. The left, middle, and right columns correspond, respectively, to the social, individualistic and profit-sharing policies, with $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.2$, the best parameters in Figure 5.2.

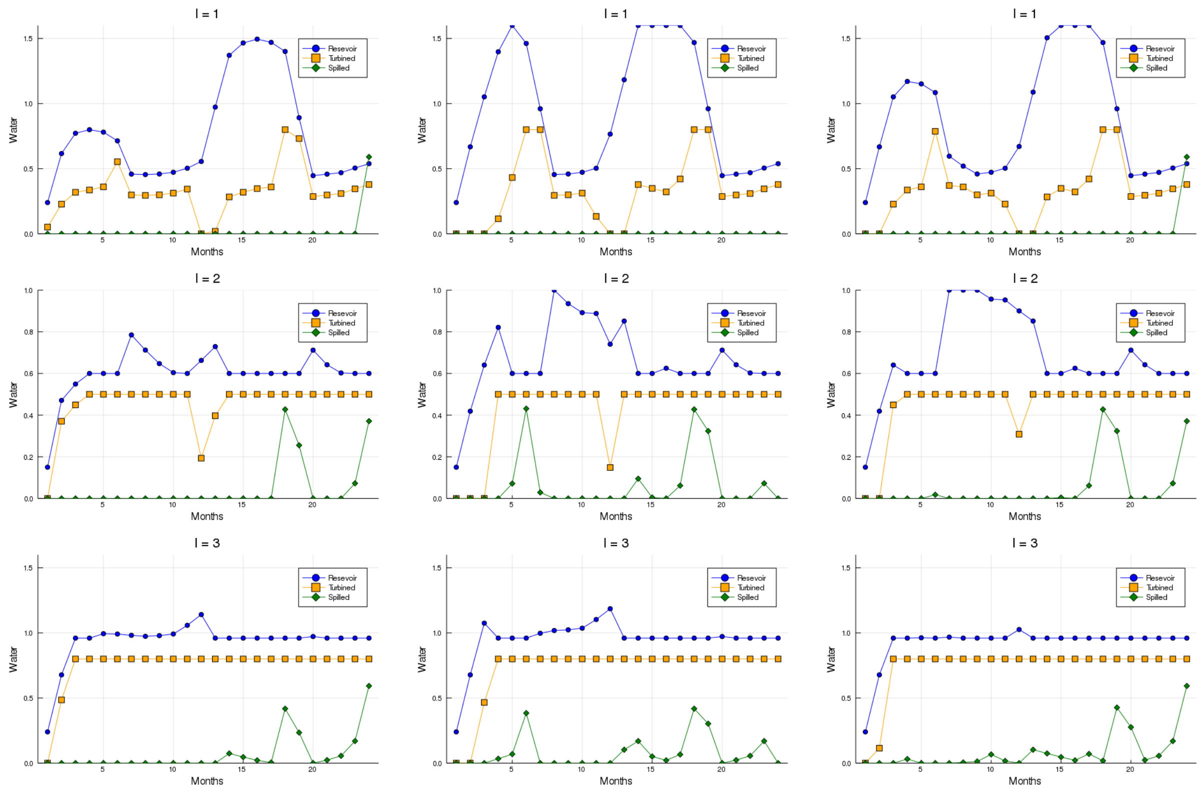

Figure 5.3: Water management of the cascade with social, individualistic and profit-sharing mechanisms (left, middle, right columns). With the individualistic approach (5.4), level 1 at the top withholds water until time $t = 5$, when the plant starts turbining to get high prices. There is more spillage with the individualistic approach, when compared to the other policies.

## 5.3 Nested stochastic optimization: individualistic approach

The trilevel formulations so far are deterministic. Problem (5.11) gives an idea on the changes induced by uncertainty. To start with, prices $\Pi_s^t$ in the objective function are uncertain and given by $S$ equiprobable scenarios, so we now have to deal with costs of the form $f_{ls}^t$, for $s \in \{1,\ldots,S\}$. Regarding the stochastic analogue of the equality $A_l x_l = a_l - B_{l-1} x_{l-1}$, it involves the water balance constraint

$$v_l^{t+1} = v_l^t + \eta(-u_l^t - w_l^t + A_l^t + u_{l-1}^t + w_{l-1}^t),$$

for inflows $A_l^t$ that were considered deterministic so far. The constraint assumes a water travel time equal to one time period: the water released $(u_{l-1}^t + w_{l-1}^t)$ arrives to the reservoir downhill at time $t+1$.

In a stochastic model, random inflows are specified by scenarios such as $A_{lr}^t$ and $A_{ls}^t$, for $r, s \in \{1, \ldots, S\}$ and all variables become indexed by scenarios. Suppose for a moment that the water travel time is equal to 1, then the water released by level $l-1$ at time $t$ reaches level $l$ at time $t+1$. In particular, the $r$th inflow scenario $A_{l-1,r}^t$ resulted in decisions $u_{l-1,r}^t$ and $w_{l-1,r}^t$ and at level $l$ this impacts the water balance constraints. If the $s$th inflow scenario $A_{ls}^{t+1}$ occurs, the corresponding constraint will be

$$v_{ls}^{t+1} = v_{lr}^t + \eta(-u_{ls}^{t+1} - w_{ls}^{t+1} + A_{ls}^{t+1} + u_{l-1,r}^t + w_{l-1,r}^t).$$

Note that scenarios $r$ and $s$ have no reason to be the same. In order to simplify the presentation, we consider below that the water travel time is zero (as opposed to the other modeling just explained), so that we deal with constraints of the form

$$v_{ls}^{t+1} = v_{ls}^t + \eta(-u_{ls}^{t+1} - w_{ls}^{t+1} + A_{ls}^{t+1} + u_{l-1,s}^{t+1} + w_{l-1,s}^{t+1}) \tag{5.13}$$

with the same scenario for all levels (the technique can still be applied for positive travel times, but the notation becomes too cumbersome).

Recall from (5.12) that individualistic profits are needed to chose $\phi$ therein and determine the actual transfer between levels. When dealing with scenarios the level interaction is quite intricate; in this section we explain the methodology for the uncertain version of (5.4), that is, when there is no transfer of profit between levels. The stochastic setting with profit being shared between consecutive levels is left for Section 5.4.

## 5.3.1 Computing individualistic two-stage policies

Since the first time step is considered deterministic in our model, both for the two-stage and the multi-stage cases, the specification of (5.13) for different time steps is

$$\begin{array}{rcll} v_l^1 &=& v_l^0 + \eta(-u_l^1 - w_l^1 + A_l^1 + u_{l-1}^1 + w_{l-1}^1) & \\ v_{ls}^{t+1} &=& v_{ls}^t + \eta(-u_{ls}^{t+1} - w_{ls}^{t+1} + A_{ls}^{t+1} + u_{l-1,s}^{t+1} + w_{l-1,s}^{t+1}) & t = 1, \ldots, \mathtt{T}-1, s = 1, \ldots, S. \end{array} \tag{5.14}$$

With the two-stage paradigm, data is considered deterministic until a time when uncertainty reveals, all at once, until the end of the horizon. For presentation purposes, it is convenient to assume that uncertainty is fully revealed after the first time step, and that the whole path of inflows $A_{ls}^{t+1}$ becomes known at once, for $t = 1, \ldots, \mathtt{T}-1$ and each $s = 1, \ldots, S$. Then in (5.14) we deal with scenarios for $t \geq 2$. Adopting a notation that can be extended for more than two stages, we let

$$\begin{array}{rcll} x_l^1 &:=& \left(u_l^t, v_l^t, w_l^t, \text{slacks}_l^t\right) & \text{for } t = 1, \\ x_{ls}^2 &:=& \left(u_{ls}^t, v_{ls}^t, w_{ls}^t, \text{slacks}_{ls}^t\right)_{t=2}^{\mathtt{T}} & \text{for each scenario } s = 1, \ldots, , \end{array}$$

respectively denote the first and second-stage variables. With this notation, choosing appropriate vectors and matrices, the relations in (5.14) can be represented in an abstract manner as

$$\begin{array}{rcll} A_l x_l^1 &=& a_l - B_l x_{l-1}^1 & \text{for constraints involving } t = 1, \\ A_{ls} x_{ls}^2 &=& a_{ls} - T_{ls} x_l^1 - B_{ls} x_{l-1,s}^2 & \text{for constraints involving } t = 2, \ldots, \mathtt{T}, s = 1, \ldots, S. \end{array} \tag{5.15}$$

For instance, taking $a_l = v_l^0 + \eta A_l^1$ and $A_l = B_l = [\eta \quad 0 \quad \eta \quad 0]$ gives the first equality in (5.14).

The starting point is once more the optimization problem at level $l$, written without any influence of the other levels. This corresponds to taking null matrices $B_{ls}$ above and, hence, we look for solutions to

$$\left\{ \begin{array}{ll} \min\limits_{x_l^1 \geq 0} & f_l^\top x_l^1 + \frac{1}{S} \sum\limits_{s=1}^{S} Q_{ls}(x_l^1) \\ \text{s.t.} & A_l x_l^1 = a_l \end{array} \right. \quad \text{for recourse functions } Q_{ls}(x_l^1) := \left\{ \begin{array}{ll} \min\limits_{x_{ls}^2 \geq 0} & f_{ls}^\top x_{ls}^2 \\ \text{s.t.} & A_{ls} x_{ls}^2 = a_{ls} - T_{ls} x_l^1 . \end{array} \right.$$

When levels are organized in a cascade, since the RHS terms are modified by decisions taken in level $l-1$, recourse functions depend not only on the first-stage variable, but also on the decision taken at level $l-1$ for the same scenario realization. As a result, instead of $Q_{ls}(x_l^1)$ as above, we now have the function $Q_{ls}(x_l^1, x_{l-1,s}^2)$, that remains polyhedral and convex.

In a two-stage formulation, the trilevel individualistic problem to be solved is

$$\begin{array}{llll}
\text{find} \quad (x_1^{1\mathrm{I}2}, x_2^{1\mathrm{I}2}, x_3^{1\mathrm{I}2}) \quad \text{such that} \quad x_l^{1\mathrm{I}2} & \text{solves} & & \begin{cases} \min\limits_{x_l^1 \geq 0} & f_l^\top x_l^1 + \frac{1}{S}\sum\limits_{s=1}^{S} Q_{ls}(x_l^1, x_{l-1,s}^{2\mathrm{I}2}) \\ \text{s.t.} & A_l x_l^1 = a_l - B_l x_{l-1}^{1\mathrm{I}2} \end{cases} \\
\text{where} \quad x_{l,s}^{2\mathrm{I}2} \quad \text{solves} & Q_{ls}(x_l^1, x_{l-1,s}^2) & := & \begin{cases} \min\limits_{x_{ls}^2 \geq 0} & f_{ls}^\top x_{ls}^2 \\ \text{s.t.} & A_{ls} x_{ls}^2 = a_{ls} - T_{ls} x_l^1 - B_{ls} x_{l-1,s}^2 \end{cases} \\
\text{for } l = 1,2,3\,. & & &
\end{array} \tag{5.16}$$

In these problems, all terms involving a subindex $l-1<1$ are void.

Algorithm 2 puts in place a decomposition approach that we refer to as a "cascaded" L-shaped method, as it extends to the trilevel setting the well-known algorithm of [SW69]. We denote by $Q_{ls}^k$ the current cutting-plane approximations for the recourse of each scenario and level, with the aggregated valued defined as

$$\mathfrak{Q}_l^k(x_l^1) := \frac{1}{S}\sum_{s=1}^{S} Q_{ls}^k(x_l^1, x_{l-1,s}^{2k}) \quad \text{at iteration } k. \tag{5.17}$$

**Algorithm 2** CASCADED L-SHAPED METHOD FOR TWO-STAGE STOCHASTIC TRILEVEL PROBLEMS. INDIVIDUALISTIC CASE

**Initialization.** Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$. Set $k = 1$ and $Q_{ls}^k(\cdot) \equiv -M$ for $l = 1,2,3$ and $s = 1,\ldots,S$. For $l = 1$, let $x_{l-1}^{1k} = x_0^{11}$ and $x_{l-1,s}^{2k} = x_{0s}^{21}$ for $s = 1,\ldots,S$ be void elements.

**REPEAT for** $l = 1,2,3$:

**First-stage master problem at level** $l$: Given $y_{l-1,s}^k$ for all $s = 1,\ldots,S$ and $\mathfrak{Q}_l^k$ from (5.17),

$$x_l^{1k} \text{ solves } \begin{cases} \min\limits_{y \geq 0} & f_l^\top y + \mathfrak{Q}_l^k(y) \\ \text{s.t.} & A_l y = a_l - B_l x_{l-1}^{1k}\,. \end{cases}$$

**Second-stage subproblems at level** $l$: for each scenario $s = 1,\ldots,S$,

$$x_{ls}^{2k} \text{ solves } Q_{ls}(x_l^{1k}, x_{l-1,s}^{2k}) := \begin{cases} \min\limits_{y \geq 0} & f_{ls}^\top y \\ \text{s.t.} & A_{ls} y = a_{ls} - T_{ls} x_l^{1k} - B_{ls} x_{l-1,s}^{2k}\,. \end{cases} \tag{5.18}$$

Let $\lambda_{ls}^k$ denote the optimal multiplier vector associated with the equality constraints.

**Model improvement for level** $l$: For $s = 1,\ldots,S$ the linearization

$$\ell_{ls}^k(x_l^1, x_{l-1,s}^2) := Q_{ls}(x_l^{1k}, x_{l-1,s}^{2k}) + (\lambda_{ls}^k)^\top T_{ls}(x_l^1 - x_l^{1k}) + (\lambda_{ls}^k)^\top B_{ls}(x_{l-1,s}^2 - x_{l-1,s}^{2k}) \tag{5.19}$$

improves the approximate recourse functions and its expected value,

$$Q_{ls}^{k+1}(x_l^1, x_{l-1,s}^2) := \max\left\{\ell_{ls}^k(x_l^1, x_{l-1,s}^2), Q_{ls}^k(x_l^1, x_{l-1,s}^2)\right\}, \quad \text{and } \mathfrak{Q}_l^{k+1}(x_l^1) \text{ defined like in (5.17).}$$

**STOPPING TEST.** Stop if for all the levels $\displaystyle\frac{1}{S}\sum_{s=1}^{S} Q_{ls}(x_l^{1k}, x_{l-1,s}^{2k}) - \mathfrak{Q}_l^{k+1}(x_l^{1k}) \leq \varepsilon\,.$

Otherwise, set $k = k+1$ and go to **REPEAT**. □

In (5.17), the left-hand side the dependency on $x_{l-1,1}^{2k}, \ldots, x_{l-1,S}^{2k}$ is dropped, for convenience.

In Algorithm 2, the loop parses $l \in \{1,2,3\}$, sequentially in the levels, and parallel with respect to scenarios (the second-stage subproblems can be solved independently). Replacing throughout $x_{l-1,s}^{2k}$ by the value computed at the previous iteration, $x_{l-1,s}^{2k-1}$ would yield a variant that is parallelizable also in the levels. Regarding convergence properties, by construction, the approximate recourse functions satisfy $Q^k_{ls}(x^1_l, x^2_{l-1,s}) \leq Q_{ls}(x^1_l, x^2_{l-1,s})$ for all scenarios $s$, levels $l$ and iterations $k$. Except for larger dimensionality, we are in a situation equivalent to the one for $v_2$ in item (ii) of Theorem 5.2.1. As a result, under the same assumptions on the linear programming solver, if $\varepsilon = 0$, after a finite number of iterations Algorithm 2 finds a solution to (5.16), the two-stage stochastic formulation of the individualistic trilevel problem.

### 5.3.2 Computing individualistic multi-stage policies

In the multi-stage paradigm, uncertainty is revealed gradually; the $t$th realization becomes known at the $t$th time stage. Accordingly, we shall deal with variables of the form

$$x^t_{ls} \quad := \quad \left(u^t_{ls}, v^t_{ls}, w^t_{ls}, \text{slacks}^t_l\right) \qquad \text{for } t = 1, \dots, \mathtt{T} \text{ and for each scenario } s = 1, \dots, S,$$

and constraints will be like the right-most equality in (5.15), noting that all scenarios have the same realization at the first time stage, so $x^1_{ls} = x^1_l$ is deterministic.

Parsing all the branches of the scenario tree in a multi-stage setting, as it was done in (5.18) in Algorithm 2, is clearly impractical. Sampling algorithms like SDDP [PP91] are the methods of choice in multistage programming. However, for a cascade of nested optimization problems, with decisions from level $l-1$ impacting the RHS of the problem at level $l$, a straightforward application of the SDDP approach is also impractical. To be more precise, recall that each basic SDDP iteration consists in certain forward and backward passes (the details are given below). Suppose we start at level 1 with a standard SDDP iteration. Once the forward-backward iterates at level 1 are available, they become RHS scenario information for level 2, therefore modifying the scenarios seen by $l = 2$. A brute-force approach in this setting would run a standard SDDP method for $l = 1$ until it converges, and only afterwards move to level 2, running a separate SDDP for each RHS defined by the forward-backward iterates at level 1 and averaging the results. After satisfying a convergence criterion for each one of those runs at level 2, the brute-force method would move to level 3. Since such approach is too cumbersome, in this section we show how to perform calculations in a manner that is computationally efficient.

Instead of sequentially applying SDDP for level $l$ and, when a convergence criterion is reached, moving to $l+1$, we perform one SDDP forward and backward passes for all the levels, and then iterate across levels. Recall that decisions taken upstream modify the downhill ones only via the RHS equality constraints, as in (5.15). In Algorithm 3 the key is in suitably transferring to level $l$ information obtained with the SDDP iteration done at level $l-1$. This is done by transporting cuts along levels.

The multi-stage formulation inevitably requires rather involved notation, that we gradually introduce for clarity. There are time stages $t \in \{1, \dots, \mathtt{T}\}$ and $S$ stage-wise independent scenarios referred to by a subindex $s$ or $r \in \{1, \dots, S\}$, having $\mathtt{T}$ components. Below, the notation with subindex $s$ refers to what is called a forward SDDP scenario, while $r$ refers to scenarios in the backward SDDP pass.

With respect to the notation in the previous section, given a scenario $s$,

$$\text{at stage } t \left\{ \begin{array}{llll} \text{the former here-and-now variable} & x^1_l & \text{corresponds to} & x^{t-1}_l \\ \text{the former recourse variables} & x^2_{ls} & \text{correspond to} & x^t_{ls} \end{array} \right. \quad \text{in the multi-stage setting.}$$

Consider first the simple setting of disconnected levels and the following shorter notation for the feasible sets

$$F^t_{ls}(x^{t-1}_l) := \{y \geq 0 : \ A^t_{ls} y = a^t_{ls} - T^t_{ls} x^{t-1}_l \ \},$$

for $l = 1,2,3$, $t = 1, \dots, \mathtt{T}$ and $s = 1, \dots, S$, and where $x^0_l$ is a given data, sometimes called the tendency. Note that the input of $F^t_{ls}(\cdot)$ does not depend on the scenario, since it can be evaluated for any iterate from state $t-1$ from any scenario, but the set-valued function $F^t_{ls}$ does depend on $s$. Letting $\mathbb{E}$ stand for the expectation operator, a nested representation for this multi-stage problem is

$$\min_{x^1_l \in F^1_l(x^0_l)} {f^1_l}^\top x^1_l + \mathbb{E}\left[\min_{x^2_l \in F^2_l(x^1_l)} {f^2_l}^\top x^2_l + \mathbb{E}\left[\dots + \mathbb{E}\left[\min_{x^{\mathtt{T}}_l \in F^{\mathtt{T}}_l(x^{\mathtt{T}-1}_l)} {f^{\mathtt{T}}_l}^\top x^{\mathtt{T}}_l\right]\dots\right]\right]. \tag{5.20}$$

In a Dynamic Programming formulation, each bracket above represents the recourse, often called cost-to-go, or future cost function, shortened to FCF from now on. In particular, the FCF at time $t$ represents the costs of all the decisions taken between $t+1$ and $\mathtt{T}$.

The basis of the SDDP approach is to define approximations at iteration $k$ moving first forward in (5.20), from $t$ to $t+1$, to find feasible points $\hat{x}^{tk}_{ls}$ for the sampled scenario $s = s^k = (s^k_1, \dots, s^k_{\mathtt{T}})$. Then (5.20) is parsed from right to left, moving backwards from $t+1$ to $t$, generating linearizations along all the branches of the given scenario, and the process is repeated with $k$ replaced by $k+1$. The points $\hat{x}_{ls}^{tk}$ computed in the forward pass are available for all $t = 1,\dots,\mathtt{T}$. At $t = \mathtt{T}$, given the forward vector $\hat{x}_{ls}^{\mathtt{T}-1,k}$, the backward pass computes

$$x_{lr}^{\mathtt{T}k} \text{ solving } Q_{lr}^{\mathtt{T}}(\hat{x}_{ls}^{\mathtt{T}-1,k}) := \begin{cases} \min & (f_{lr}^{\mathtt{T}})^{\top} y \\ \text{s.t.} & y \in F_{lr}^{\mathtt{T}}(\hat{x}_{ls}^{\mathtt{T}-1,k}), \end{cases}$$

for all $r = 1,\dots,S$. Similarly to (5.19), the solution process provides a linearization that improves the current piecewise affine function $Q_{ls}^{\mathtt{T}k}$. An average of those cutting-plane models gives the expected value $\mathfrak{Q}_l^{\mathtt{T},k}$, the FCF for the backward problem at $t = \mathtt{T}-1$. Proceeding further for any stage $t = \mathtt{T}-1, \mathtt{T}-2, \dots$, for all $r = 1,\dots,S$ the backward iterate

$$x_{lr}^{tk} \text{ solves } Q_{lr}^{tk}(\hat{x}_{ls}^{t-1,k}) := \begin{cases} \min & f_{lr}^{t\ \top} y + \mathfrak{Q}_l^{t+1,k}(y) \\ \text{s.t.} & y \in F_{lr}^{t}(\hat{x}_{ls}^{t-1,k}). \end{cases}$$

Letting $Q_{lr}^{\mathtt{T}+1,k} \equiv 0$, the formulation above is also valid for $t = \mathtt{T}$. The backward pass generates linearizations that improve the cutting-plane models defining the FCF to be used in the next forward pass, $\mathfrak{Q}_l^{t+1,k+1}$. Convergence of such procedure to a solution of the multi-stage problem, with probability one, can be found in [Sha11], as well as the required assumptions for the result to hold (on the sampling and on conditions on the linear programming solver similar to those in Theorem 5.2.1).

When feasible sets are connected stagewise but not between levels, the FCF depends only on the decision taken for the considered scenario and stage, at the current level $l$. When levels are connected in a cascade, the recourse functions of level $l$ depend on decisions taken at level $l-1$. As explained for the two-stage case, using the multi-stage notation and for a scenario $r = 1,\dots,S$ and $l = 1,2,3$,

$$\left(x_l^2, x_{l-1,r}^2\right) \text{ from Algorithm 2 becomes } \left(x_l^{t-1}, x_{l-1,r}^t\right),$$

and, similarly to (5.15), the RHS dependencies change the feasible sets to

$$F_{lr}^t(x_l^{t-1}, x_{l-1,r}^t) := \left\{ y \geq 0 : \ A_{lr}^t y = a_{lr}^t - T_{lr}^t x_l^{t-1} - B_{lr}^t x_{l-1,r}^t \ \right\}.$$

Hence, with connected levels, $Q_{lr}^{tk}$ is a function of $\left(x_l^{t-1,k}, x_{l-1,r}^{tk}\right)$. Note that that we use $x_l^{t-1,k}$ instead of $x_{lr}^{t-1,k}$, because the value of $x_l^{t-1,k}$ can come from any scenario at stage $t-1$. The two-stage case is similar to the last bracket in (5.20). Carrying on the parametric dependencies backwards by reasoning recursively based on the two-stage case, we see that the recourse function at stage $t$ and level $l$ depends

$$\begin{array}{lll} \text{on} & (x_l^{t-1}, x_{l-1,r}^t), & \text{through the feasible set } F_{lr}^t(x_l^{t-1}, x_{l-1,r}^t), \\ \text{on} & Z_{l-1}^{t+1} := \left(x_{l-1,s}^{t'}, s = 1,\dots,S, t' = t+1,\dots,\mathtt{T}\right), & \text{through the future cost function}. \end{array} \tag{5.21}$$

With respect to the two-stage setting, the main difference is in dimensionality increase of the arguments of the recourse function. As such, the trilevel individualistic problem is an extension of (5.16). to the multi-stage setting, with the solutions $x_{ls}^{t^{\mathtt{IT}}}$ solving nested problems of the form

$$Q_{lr}^t(x_{ls}^{t-1^{\mathtt{IT}}}, Z_{l-1}^{t^{\mathtt{IT}}}) = \begin{cases} \min & f_{lr}^{t\ \top} y + \frac{1}{S}\sum_{s=1}^{S} Q_{ls}^{t+1}(y, Z_{l-1}^{t+1^{\mathtt{IT}}}) \\ \text{s.t.} & y \in F_{lr}^t(x_{ls}^{t-1^{\mathtt{IT}}}, x_{l-1,r}^{t^{\mathtt{IT}}}), \end{cases}$$

for each scenario $r,s = 1,\dots,S$, stage $t = 1,\dots,\mathtt{T}$ and level $l = 1,2,3$.

At iteration $k$ of the cascaded SDDP method, by definition of $Z$ in (5.21), the identity $Z_{l-1}^{t+1,k} = (x_l^{t-1,k}, Z_{l-1}^{t,k})$ holds. So at stage $t$ the recourse function is computed at

$$(x_l^{t-1,k}, x_{l-1,1}^{tk}, \dots, x_{l-1,S}^{tk}, Z_{l-1}^{t+1,k}) = (x_l^{t-1,k}, Z_{l-1}^{t,k}),$$

and backward problems yield

$$x_{lr}^{tk} \text{ solving } Q_{lr}^{tk}(\hat{x}_{ls}^{t-1,k}, Z_{l-1}^{t,k}) := \begin{cases} \min & f_{lr}^{t\ \top} y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{lr}^t(\hat{x}_{ls}^{t-1,k}, x_{l-1,r}^{tk}), \end{cases}$$

where $\mathfrak{Q}_l^{t+1,k}$ is the expected future cost function.

The solution algorithm is given in Algorithm 3.

**Algorithm 3** CASCADED SDDP METHOD FOR MULTI-STAGE STOCHASTIC TRILEVEL PROBLEMS. INDIVIDUALISTIC CASE

```
Initialization. Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$. Set $k = 1$. For $l = 1,2,3$, $s = 1,\dots,S$ and $t \geq 2$, let $Q_{ls}^{t,k} \equiv -M$. For $t = 1$, all levels and scenarios, the initial tendency $\hat{x}_{ls}^{t-1,k}$ is given. For $l = 1$, the vectors $x_{l-1,s}^{tk}$ are void.

Sampling. Sample a scenario $s_t^k \in \{1,\dots,S\}$ for each $t = 1,\dots,\mathtt{T}$. To simplify notation recall that all scenarios at $t = 1$ are assumed to be the same.

REPEAT For $l = 1,2,3$:

    Forward pass. For each $t = 1,\dots,\mathtt{T}$ and $s = (s_1^k,\dots,s_\mathtt{T}^k)$, compute
        $\hat{x}_{ls}^{tk}$ solving $\begin{cases} \min & {f_{ls}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}) \\ s.t. & y \in F_{ls}^t(\hat{x}_{ls}^{t-1,k}, x_{l-1,s}^{tk}). \end{cases}$

        Init. Take $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.
        Iterate Across Stages. For $t = \mathtt{T},\dots,2$.
            Cut computation. For $r = 1,\dots,S$, solve
                $x_{lr}^{tk}$ solving $\begin{cases} \min & {f_{lr}^t}^\top y + \mathfrak{Q}_l^{t+1,k+1}(y, Z_{l-1}^{t+1,k}) \\ s.t. & y \in F_{lr}^t(\hat{x}_{ls}^{t-1,k}, x_{l-1,r}^{tk}). \end{cases}$
            Obtain subgradients such that for all $x_l^{t-1}, x_{l-1,r}^t$ and $Z_{l-1}^{t+1}$ the value function $Q_{lr}^t(x_l^{t-1}, x_{l-1,r}^t, Z_{l-1}^{t+1})$ lies below
                $Q_{lr}^t(x_{ls}^{t-1,k}, x_{l-1,r}^{t,k}, Z_{l-1}^{t+1,k}) + (\lambda_l^{t-1,k})^\top(x_l^{t-1} - \hat{x}_{ls}^{t-1,k}) +$
                $(\mu_{l-1,r}^{t,k})^\top(x_{l-1,r}^t - x_{l-1,r}^{t,k}) + (\nu_{l-1}^{t+1,k})^\top(Z_{l-1}^{t+1} - Z_{l-1}^{t+1,k}).$

            Cut Aggregation. Average the cuts in (3) to obtain a cut such that $\mathfrak{Q}_{lr}^t(x_l^{t-1,k}, Z_{l-1}^t)$ lies below
                $\mathfrak{Q}_{lr}^t(x_{l\omega}^{t-1,k}, Z_{l-1}^{tk}) + (\phi_l^{t-1,k})^\top(x_l^{t-1} - x_{l\omega}^{t-1,k}) + (\rho_{l-1}^{tk})^\top(Z_{l-1}^t - Z_{l-1}^{tk}).$   (5.22)
            Define $\mathfrak{Q}_{lr}^{t,k+1}$ as a maximum between $\mathfrak{Q}_{lr}^{t,k}$ and (5.22).

    Upper bound. Set $v_l^k = \sum_{t=1}^{\mathtt{T}} {f_{ls}^t}^\top \hat{x}_{ls}^{tk}$ for the estimation of the upper bounds via average, where $s = (s_1^k,\dots,s_\mathtt{T}^k)$.
    Lower bound. Set $u_l^k$ as the optimal value of the subproblem at $t = 1$ solved on the backward pass for estimation of the lower bounds via average.

STOPPING TEST. Stop if the average lower bound and average upper bound for all levels are close enough or the lower bounds stabilized. Otherwise, set $k = k+1$ and go to the Sampling step again. □
```

It is worth noting that the forward samples are common for all levels, which is quite natural if one has the brute-force solution procedure in mind. Also, the cut calculation performed at the backward pass is entirely based on the two-stage case. Because of the dependence on the iterates computed at level $l-1$, cuts have (much) larger dimension in the cascaded setting.

Since at the top level the past vectors $x_{l-1}^{tk}$ are void, when $l = 1$ Algorithm 3 boils down to a standard SDDP iteration. Under the same assumptions as in [Sha11, Proposition 3.1], with probability one, the forward step at $l = 1$ produces an optimal policy after finitely many iterations. As a result, with probability one and for sufficiently large $k$, the cutting-plane models in our cascaded SDDP represent well the relevant parts of the future costs at level 1. This property does not suffice to ensure convergence in the trilevel setting, however. The reason bears some resemblance with risk-averse forms of SDDP, to solve problems as in (5.20), with the expectation operator replaced by a risk measure, see [KM14]. As noted in [Sha11, Remark 5], sampling makes the upper bound $v_1^k$ random. When passing to level $l = 2$, the lower bound $u_2^k$ is also random, because it depends on cuts involving forward-backward iterates from level 1. The lower bound at level 3 is also random and, therefore, no convergence result is available in this setting. However, in practice we observe small gaps, after averaging.

Algorithm 3 is essentially an efficient implementation of the brute-force algorithm described at the beginning of this section (which itself would not be computationally practical). The difference is that, when a new sequence of forward-backward iterates is obtained at level 1, the cuts at level 2 already provide a valid lower bound for the

SDDP problem at level 2 associated with the new sequence from level 1. Thanks to this feature, we do not need to solve the SDDP problem at level 2 from scratch. Analogously, the cuts at level 3 provide a valid lower bound given the new forward-backward iterates at level 2.

For a problem with four stages and three scenarios ($\mathtt{T} = 4$ and $S = 3$), the diagram in Figure 5.4 illustrates with solid lines how information flows from level $l$ (up) to level $l+1$ (bottom) in Algorithm 3. For each level, dotted boxes indicate the path of scenarios sampled in the forward pass, given by $s = (0,3,2,2)$ in the figure. Since the same path of scenarios is used for all levels, dotted boxes have the same position in the top and bottom rows. The dotted lines connecting boxes horizontally represent how information is transmitted in the same level, between time stages. Those lines have no arrows because information goes both ways, forward and backwards, as in the SDDP passes. Notice also that all boxes are connected from stage to stage. When moving forward in time, decisions from box $l, t = 2, s = 3$ go to all scenarios at $l, t = 3$, while when moving backwards, cuts from all boxes at $l, t = 3$ go back to $l, t = 2, s = 3$.

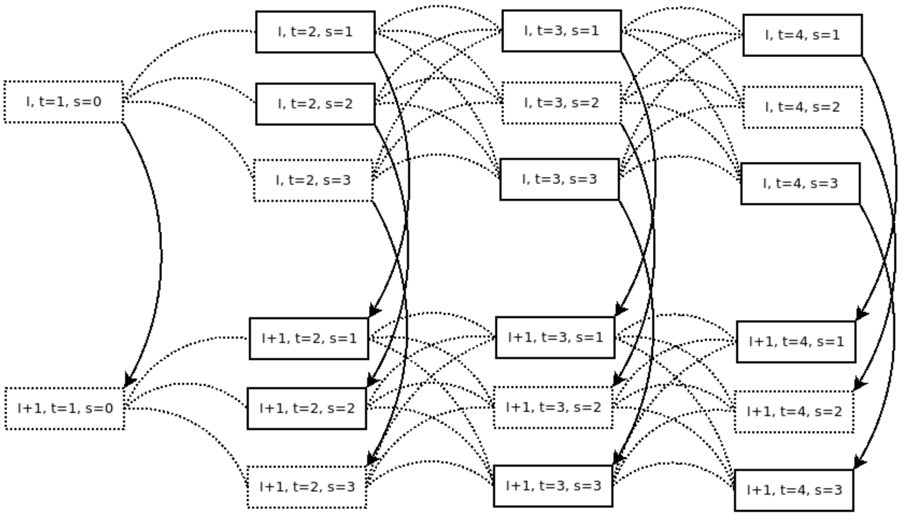

Figure 5.4: Illustration of the flow of information in Algorithm 3.

### 5.3.3 Numerical assessment

We designed a stochastic variant of our toy problem (5.11), with uncertainty in the prices $\Pi^t$ and inflows $A_l^t$. The respective considered scenarios are shown in Figure 5.5.

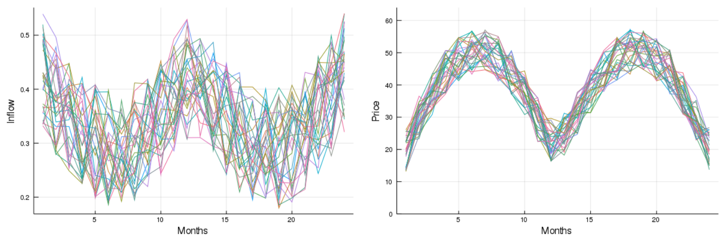

Figure 5.5: Scenarios for the water inflow and price. Inflows are shown as percentages of reservoir's volume. The inflow around December is a fraction of the maximum volume followed by a dry period around the middle of the year. Prices follow an inverse pattern. Recall that the deterministic values taken for price and inflow in (5.11) are stagewise averages of these scenarios.

The dimensions of the cascade are the same of the deterministic case, as well as the initial volumes. The individualistic policies obtained by the cascaded SDDP method presented in Algorithm 3 are compared to the social policies obtained with the traditional SDDP method for managing the cascade jointly. Professional SDDP software typically exploits parallelization. This is not the case with our implementation, which took up to 8 hours to produce the output reported below. We run 100 forward-backward iterations on each level and initialize the cuts at $l = 1$ to make the process faster. All the obtained gaps are smaller than 4%, relative to the lower bound.

We simulate the cascade operation with each policy, for new out-of-sample 100 scenarios, the corresponding mean profit for each level is reported in Table 5.2.

| Policy Type | Profit $l = 1$ | Profit $l = 2$ | Profit $l = 3$ | Tot. Profit |
|---|---|---|---|---|
| Social (Standard SDDP) | 336.89 | 439.16 | 710.05 | 1486.12 |
| Individualistic (Cascaded SDDP) | +4.20% | -6.13% | -2.86% | -2.22% |

Table 5.2: Comparison of expected profits for each level relative to the social policy. The qualitative behavior of the profits is like in the deterministic case: the hydroplant at $l = 1$ earns more and the hydroplants downhill earn less. Also, the total wealth obtained from the cascade decreases.

The mean reservoir operation and its standard deviation are reported for each level in Figure 5.6, respectively in solid and dashed lines. With the social policy, the spillage observed at the end of the simulation is just an evidence of the end-of-horizon effect, that does not affect the profits. With the individualistic policy, similarly to the deterministic model, the top hydroplant does not deplete its reservoir at initial times, saving water for it to be released when prices are high. The pattern of each policy is similar to the one observed in the deterministic case reported in Figure 5.3.

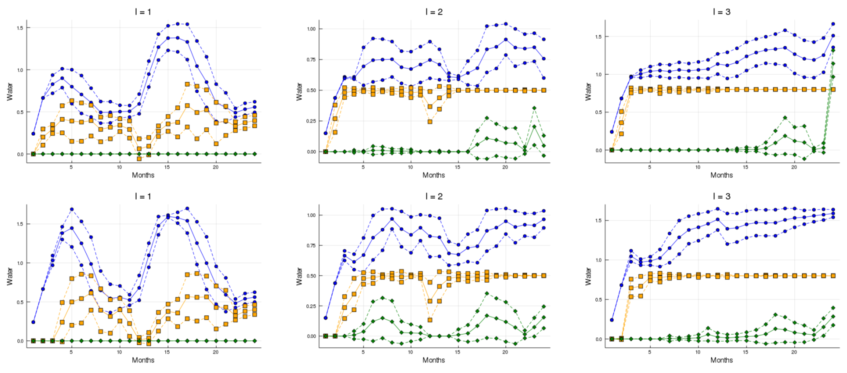

Figure 5.6: Water management of the cascade with social and individualistic policies (top and bottom rows) in the stochastic setting, computed with three different approaches. Circles represent volume, squares turbined water and "diamond" spillage. Given the lines of the same color, the dashed lines correspond to the larger and smaller variation in the solutions of the three methods.

## 5.4 Sharing mechanism between neighbors only

We now extend the multi-stage procedure to deal with the mechanism of profit sharing. With respect to the deterministic case, the setting is slightly less general, as here we assume the transfer is done only between consecutive levels (this amounts to taking $\tau_{3\to1} = 0$ in the deterministic formulation). For the cascade in Figure 5.1, we now consider dependencies represented by both the left and right arrows, only that level 3 shares profit with level 2, but not with level 1 (in the figure, the left arrow with the label $\tau_{3\to1}$ is not present).

The iterative procedure follows the rationale in the previous sections, defining cuts based on extended variables, as in (5.21), only that now we adopt a handy symbolic representation. The mechanism is described below in an informal style to avoid heavy and cumbersome notation. A more precise statement of the algorithm is given in Appendix B.

### 5.4.1 The concept of floating cut

The multi-stage individualistic model gives a hint on the difficulties that need to be tackled when there is a hierarchy of three nested multi-stage programs with transfer of profit to the next level upstream. The challenge in the more general setting considered here starts with defining an extended variable, from which the parametric dependencies in the FCF can be written down, to define the linearizations, or "cuts". Along the lines in (5.21), we need to detect connections between the current decision variable (say $x_l^{t-1}$) and decision variables of other levels (say $Z_{l-1}^t$).

The difference here is that, instead of trying to figure out all the (nontrivial) dependencies that can happen by ourselves, we put in place a symbolic code that detects automatically those relations and defines a "floating cut". The procedure starts representing the forward and backward linear programs that are solved for each level in a manner similar to a modeling language. For a given optimization problem, the representation stores, in a human-readable format, three data structures with relevant information. A first structure deals with variables, distinguishing decisions from parameters (a certain flag is set to 0 or 1), specifying attributes such as type (continuous, binary), bounds, name, and storing the actual value of the variable in question. A second data structure represents the linear expressions appearing in the optimization problem (a real number and a list of pairs of real numbers and instances of the data structure variable). The third structure contains a linear expression for the objective function and a list of pairs of linear expressions and integers to represent the constraints in the optimization problem under consideration.

With this symbolic representation at hand, and its distinction between parameters and decisions, it is possible to compute values for the latter, given the values of the former. The same solution process yields optimal dual variables for the constraints, and these values are stored in the first data structure, with the problem variables. A floating cut is the symbolic expression of a linearization like (5.10) in Algorithm 1. Namely, an affine relation given by the Benders cut that results from fixing the parameters at their current values in the data structure. Since the floating cut is a linear expression of all the right-hand side parameters involved in the optimization problem, it can be symbolically represented by means of the second data structure. At every iteration, knowing the parameters of the optimization problem, a specific instance is obtained, and its value is inserted as a new constraint in the third data structure of other optimization problems (those for which the current decision variable appears as a parameter).

The sophisticated construction of floating cuts is fundamental to manipulate efficiently the huge amount of optimization problems and linearizations involved in the multi-stage three-level setting. The naming is justified by the following observation. When a specific subproblem is to be solved, the portions of the linear expressions associated with parameters are reduced to a number, called the parametric value of the linear expression, based on the values of those parameters. When the values of the parameters change, the parametric value of the linear expressions change too. In turn, this change is interpreted as the free term of the linear constraints having "floated" to another level or stage.

As we explain now with an example, before floating a cut we might also need to update its value. Consider an SDDP problem with $\mathtt{T} = 3$ and $S = 2$ (only one level) and suppose iterations start from $\mathtt{T}$, backwards to $\mathtt{T}-1$. The decision variables at $t = 3$ and $s = 1, 2$ are $x_s^t$ and the parameter at $t = 3$ is the forward decision at $t = 2$, denoted by $\hat{x}^t$ (for this variable we drop the scenario subindices for convenience). The cuts computed on any scenario at $t = 3$ are functions of the parameter, the forward decision $\hat{x}^t$. By this token, for a subproblem at $t = 2$, the forward decision $\hat{x}^3$ will be considered as a parameter. But when $t = 2$, the parameter is $x_1^2$ if $s = 1$, and $x_2^2$ if $s = 2$. For this reason, the value of the floating cut computed at $t = 3$ needs to be updated, replacing the parameter $\hat{x}^3$ by the variable $x_s^2$ for each subproblem scenario. A similar translation step needs to be performed to account for the fact that the forward decision taken at $t = 1$ is a parameter for all subproblems at $t = 2$.

Suppose we represent symbolically the subproblems of a multi-stage stochastic problem and that we declare symbolically all the external parameters of each problem at each level, stage, and scenario. When making this symbolic declaration we do not know the dependencies of the FCF of that specific subproblem. However, we can always start with a large negative number as a valid approximation. Given the values of all the RHS parameters on the symbolic representation, a symbolic expression for the cut can be derived, which depends on all RHS parameters found on the symbolic representation. Whenever the values of the RHS parameters are updated, the symbolic linearization, that we named the "floating" cut, is updated too.

This symbolic representation of the cuts, instead of dealing with the usual matrix- and index-based representations, suffices to detect non trivial parametric dependencies for the multi-stage case, as long as calculations are carried out in the correct order. To understand this issue, let us explore the two-stage individualistic setting as an example. Assume we have forward-backward iterates at $l = 1$. The first cuts computed at $l = 2$ are those corresponding to the final time index. Each scenario subproblem at the final time $\mathtt{T}$ depends on $x_l$ and $x_{l-1,s}^2$. The symbolic cuts computed at each scenario would be a linear function of both $x_l$ and $x_{l-1,s}^2$. After averaging the symbolic expressions for all scenarios, we would recognize that the FCF at the first stage is a function of $x_l$, $x_{l-1,1}^2$, $\dots$, $x_{l-1,S}^2$. The same reasoning applies to the multi-stage case, except that at stage $\mathtt{T}-1$ we would have to search for new parameters on the symbolic representation of the subproblems at $\mathtt{T}-1$ that come from subproblems at stage $\mathtt{T}$.

We emphasize that to compute the general affine expression of the floating cut, we need the values of all parameters and we need to possibly update the list of RHS parameters as dependencies appear along the process. In this sense, the individualistic cascaded SDDP Algorithm 3 first gets values of external parameters making a forward-backward pass at $l = 1$. When making the forward-backward pass at $l = 2$, all the correct parametric dependencies show up naturally, which also gives the values of the external parameters at $l = 3$ and so on. Therefore, although hard to express precisely, the mechanism is not difficult to implement.

The procedure can fail if cuts are computed in the wrong order. More specifically, if $\mathtt{T} = 3$ and the first cut is computed at $t=2$, calculations would be made without the parametric dependencies at the final time $t=3$ and all the FCF estimations arriving at $t=1$ would be wrong. For the multi-stage problems we solve, it is enough to just follow the standard iterations, but always start computing cuts at the last stage. This is not an issue for the traditional SDDP method: in the parametric symbolic view, it amounts to fixing the values of all the RHS parameters. The corresponding contribution of the fixed parameters on the floating cuts is zero and, hence, our approach is a generalization of the well-known SDDP algorithm.

These explanations should make it clear that rather than struggling to express the correct RHS parametric dependencies, the real issue is to organize the symbolic calculations in the correct order, starting from the final time $\mathtt{T}$. In our trilevel problem, the order is clear since decisions are sequential in nature and, under reasonable assumptions, the initial lower bounds for value functions remain valid for bounded values of the external RHS parameters.

### 5.4.2 Computing policies with profit sharing

Having outlined the general procedure, we now focus on the sharing mechanism when dealing with nested SDDP problems. To fix ideas, suppose that SDDP problem 1 influences SDDP problem 2 and vice-versa (in the individualistic setting, SDDP problem 1 affects SDDP problem 2, which affects SDDP problem 3, but not the other way round). Dropping unnecessary indices, we let the corresponding value functions be defined as

$$v_1(x_2) = \min_{x_1} f_1(x_1, x_2) \text{ s.t. } x_1 \in X_1(x_2),$$

and

$$v_2(x_1) = \min_{x_2} f_2(x_2, x_1) \text{ s.t. } x_2 \in X_2(x_1)\,.$$

Typically, the solution to such a pair of problems is addressed by computing a generalized equilibrium, for example by iterating over the best-response of one player, given the other players' strategies are fixed. See [SMK18]. When the pair of problems at hand is simple, the best-response iteration is easy to implement and might converge to an equilibrium. When dealing with a pair of multi-stage stochastic problems, the situation is much less straightforward. Even the computation of the best response given the strategies of the other players is a hard task. Floating cuts are very useful in this setting, because linearizations computed for a given $x_2^k$ can be carried over to another iterate $x_2^{k+1}$. Thanks to the floating cuts, a best-response iterative procedure is possible in the context of multi-stage stochastic equilibrium problems as we further explain now.

Similarly to (5.21), the tuple $Z_l = \{x_{ls}^t : t = 1, \ldots, \mathtt{T}, \quad s = 1, \ldots, S\}$ represents feasible forward-backward iterates for each level $l$. Additionally, $Z_{-l}$ is the vector of tuples referring to levels other than $l$; in particular, $Z_{-1} = (Z_2, Z_3)$. For the trilevel problem feasible tuples are obtained by sequentially making a forward-backward pass at $l = 1, 2, 3$, in order. The tuple $Z_l$ is random because it depends on the scenarios sampled to perform the forward passes.

Recall that the parameter $\tau_{\to l}$ is the fraction of cost being transferred to level $l \in \{1, 2\}$ from level $l+1 \in \{2, 3\}$. Accordingly, the symbolic declaration of the $r$th scenario subproblem at stage $t$ and level $l = 1, 2$ is given by the expression below:

$$Q_{lr}^t(x_l^{t-1}, Z_{-l}) := \begin{cases} \min & f_{lr}^{t\,\top} y + \mathfrak{Q}_l^{t+1}(y, Z_{-l}) + \tau_{\to l}\mathfrak{U}_{lr}^t(y, Z_{-l}) \\ \text{s.t.} & y \in F_{lr}^t(x_l^{t-1}, x_{l-1,r}^t). \end{cases}$$

Notice that we now have two functions in the objective function. The first one, $\mathfrak{Q}$, deals with the usual nested Dynamic Programming scheme similar to (5.20). The second function, $\mathfrak{U}$, is specific to the sharing mechanism in our proposal. A closer inspection of the symbolic representation above reveals some additional features. To begin with, the first argument in the recourse function, $x_l^{t-1}$, has no scenario subindex because its value can come from any scenario at stage $t-1$ (as in the dotted lines in Figure 5.4). By contrast, the RHS value $x_{l-1,r}^t$, defining the feasible set, comes necessarily from the $r$th scenario at level $l-1$ (as in the full lines in Figure 5.4). In addition, the cut computed from this representation is an affine function of $Z_{-l}$ as well as of $x_l^{t-1}$ and $x_{l-1,r}^t$.

Along iterations, the only information that changes in the symbolic representation is the piecewise affine approximations $\mathfrak{Q}_l^{t+1}(\cdot,\cdot)$ and $\mathfrak{U}_{lr}^t(\cdot,\cdot)$, which initially are set to a fixed negative number, sufficiently large. Being the well-known FCF within the SDDP problem at level $l$, $\mathfrak{Q}_l^{t+1}(\cdot,\cdot)$ is the usual function updated in a forward-backward pass. The new function $\mathfrak{U}_{lr}^t(\cdot,\cdot)$, represents the instantaneous cost realized at stage $t$ and scenario $r$

from level $l+1$. As such, for $l=2$, the new function is a piecewise affine approximation of

$$U^t_{l+1,r}(x^{t-1}_{l+1},x^t_{lr}) := \begin{cases} \min & {f^t_{l+1,r}}^\top y \\ \text{s.t.} & y \in F^t_{l+1,r}(x^{t-1}_{l+1},x^t_{l,r}) \,, \end{cases}$$

and, for $l=1$, it is an approximation of

$$U^t_{l+1,r}(x^{t-1}_{l+1},x^{t-1}_{l+2},x^t_{lr}) := \begin{cases} \min & {f^t_{l+1,r}}^\top y + \tau_{\to(l+1)} U^t_{l+2,r}(x^{t-1}_{l+2},x^t_{lr}) \\ \text{s.t.} & y \in F^t_{l+1,r}(x^{t-1}_{l+1},x^t_{l,r}) \,. \end{cases}$$

We are now dealing with two different polyhedral approximations, whose update needs to be done in a manner slightly different from the usual SDDP. The following list explains the procedure step by step. Algorithms 4 to 9, given in the Appendix B, respectively correspond to pseudo-code for items 1-3, 4, 5, 6, 7, and 8 below.

1. First, a sequence of forward samples for each level is generated and the tuples $(Z_1,Z_2,Z_3)$ are obtained by making a forward-backward pass at each level *without adding any cuts*. At this initialization phase, this is crucial to detect the correct parametric dependencies in the symbolic representation. The goal is to obtain points at which cuts can be computed in a second phase, which needs to parse the levels and stages at the right order.

2. Since $l=3$ does not receive any transfer from lower levels ($\tau_{\to 3}=0$), the last level triggers the updating procedure. The symbolic mechanism applied to $l=3$ will detect parametric dependencies in the floating cuts that are similar to those arising in the individualistic setting for $l=3$. The functions $\mathfrak{Q}^t_{lr}(\cdot,\cdot)$ with $l=3$ are updated making a forward-backward pass at $l=3$, that fills the values for $Z_3$.

3. The next step is to update the functions $\mathfrak{U}^t_{lr}(\cdot,\cdot)$ for $l=2$. This is quite natural, since the iterates at $l=2$ would be more informed if they could take into account the implied costs of $l=3$. Accordingly, for all $t=1,\ldots,\mathtt{T}$ and $r=1,\ldots,S$, we compute a cut for the value function $U^t_{l+1,r}(x^{t-1}_{l+1},x^t_{lr})$, where $x^{t-1}_{l+1}$ and $x^t_{lr}$ are taken from the tuples $(Z_1,Z_2,Z_3)$. Precisely, $x^{t-1}_{l+1}$ is the forward iterate associated with the scenario sampled at stage $t-1$ and $x^t_{lr}$ is the corresponding value at $Z_2$ associated with scenario $r$. Note that there is a difference relative to the scenario used for $x^{t-1}_{l+1}$ and $x^t_{lr}$, as already announced. On the initialization step, we sampled a sequence of scenarios $s=(s_1,\ldots,s_{\mathtt{T}})$. The value used for $x^{t-1}_{l+1}$ is the one associated with $s_{t-1}$ and not with the $r$ index. After such step, all functions $\mathfrak{U}^t_{lr}(\cdot,\cdot)$ for $l=2$ have been updated.

4. The next step is to run a forward-backward pass at $l=2$, but this time updating the future costs $\mathfrak{Q}^{t+1}_l(\cdot,\cdot)$, to detect the parametric dependencies from $l=3$ that impact on $l=2$. We start computing the cuts at the last stage, $\mathtt{T}$.

5. Since now $\mathfrak{U}^t_{lr}(\cdot,\cdot)$ for $l=2$ depends on $x^{t-1}_{l+1}$, after aggregating the symbolic cuts, we realize that the SDDP problem at level $l=2$ depends not only $Z_1$, but also on the forward decisions at $l=3$ represented by $Z_3$. However, for the algorithm this dependence is dealt with extremely easily. A particularity is that the instantaneous cost $\mathfrak{U}^t_{lr}(\cdot,\cdot)$ for $l=2$ influences the future cost $\mathfrak{Q}^{t+1}_l(\cdot,\cdot)$ for $l=2$, which is again extremely natural.

6. The algorithm continues in the same fashion at $l=1$. We start updating $\mathfrak{U}^t_{lr}(\cdot,\cdot)$ for $l=1$ and then perform a forward-backward pass at $l=1$ updating the future costs and performing parameter detection. We again observe that SDDP problem at $l=1$ depends on some components of $Z_2$ and $Z_3$.

7. When the forward-backward pass at $l=1$ is finished, we sample a new sequence of scenarios $s=(s_1,\ldots,s_{\mathtt{T}})$ to make the initialization step again, without cut computation in the backward pass, and go back to the forward-backward at $l=3$ again. In other words, we sample $s=(s_1,\ldots,s_{\mathtt{T}})$ and compute a new tuple $(Z_1,Z_2,Z_3)$ without adding cuts. Then, we go back to step 2 and start all over with $l=3$.

For each sample $s=(s_1,\ldots,s_{\mathtt{T}})$ obtained, we also obtain for each level a realization of an estimate of a lower bound and another one for the upper bound. After averaging these realizations of the upper and lower bounds, we obtain an expected gap, which is used to stop the algorithm.

Back to the best response setting, it is important to understand that every time some component of $Z_2$ and $Z_3$ change, the cuts available at $l=1$ need to be transported ("floated"), so that they are still valid for the new values of $Z_2$ and $Z_3$. The symbolic cuts enable the application of a best-response iteration to our trilevel nested multi-stage stochastic setting.

## 5.4.3 Numerical assessment

Our last set of experiments illustrates the nested multi-stage stochastic setting with the profit sharing mechanism, solved by the technique based on symbolic dependencies and floating cuts described in the previous sections. Since the procedure is quite involved, we use for comparison in the benchmark Algorithm 1 with $\tau_{3\to1} = 0$. This is the deterministic model with profit sharing; our rationale is that results should be similar if scenarios do not vary too much. In our runs, such variability depends on the standard deviation of considered scenarios shown in Figure 5.5.

The range chosen for the profit-sharing percentages is

$$\tau_{2\to1} = \tau_{3\to2} \in [0.05, 0.9].$$

Our prototype code is not efficient and multi-stage stochastic trilevel problems are computationally heavy. To keep running times manageable for our code, the configuration from Section 5.3.3 is run with shorter time horizon and less scenarios, taking $\mathtt{T} := 8$ and $S := 5$. Each run takes 2 hours, ending with gaps smaller than 3% after 100 forward-backward iterations.

The simulation phase, with the system operating with the obtained policies uses 100 out-of-sample scenarios with the profile reported in Figure 5.5, truncated at $\mathtt{T} = 8$.

Figure 5.7 follows the premises in Figure 5.2, with vertical bars representing the total gain for each level. The results with the deterministic configurations is shown on the left, and the stochastic one on the right, taking the expected value of the cost of the 100 simulations. Numeric values for the transfers can be found in the Appendix A, Tables A.2 and A.3.

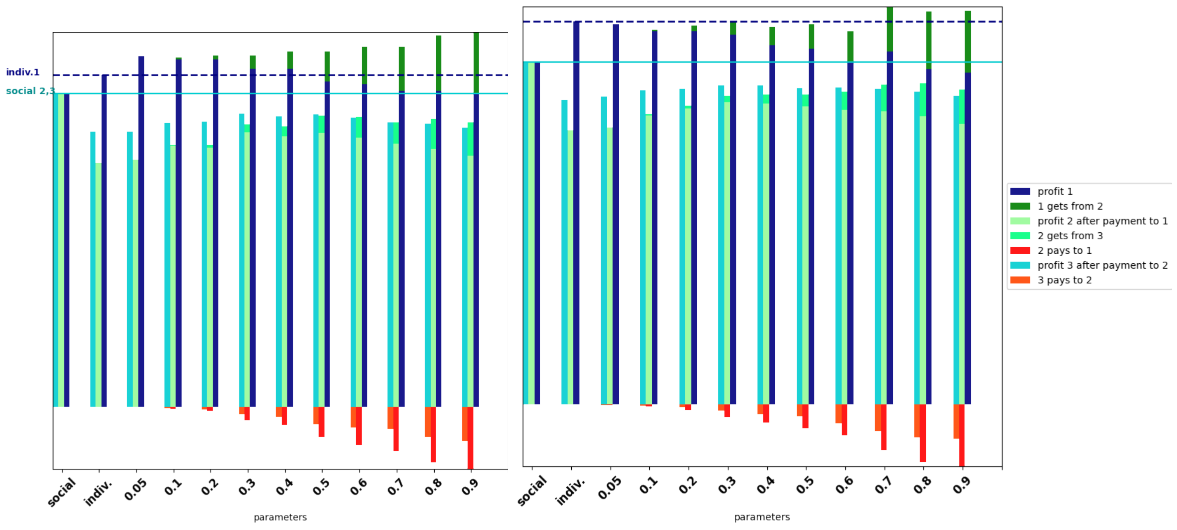

Figure 5.7: Wealth with deterministic and stochastic sharing policies (left and right). Since the latter considers few scenarios with little dispersion, the pattern of the bars is similar to the deterministic case. For level 1, any configuration of parameters above the dashed horizontal line is acceptable. For levels 2 and 3, the best is to get the closest to the solid horizontal line. In the stochastic model, the rightmost configurations that are satisfactory for the three levels are $\tau_{2\to1} = \tau_{3\to2} \geq 0.7$, whereas $\tau_{2\to1} = \tau_{3\to2} \geq 0.3$ suffices in the deterministic case. With the stochastic model and for $\tau_{2\to1}$ and $\tau_{3\to2}$ closer to 1, level 1 profit (the blue section of the bar) is close to the one obtained with the social profit (indicated by the solid horizontal line). With those configurations level 1 achieves a gain comparable to the individualistic policy (the dashed line) only after receiving a payment from level 2. For those configurations, the uphill level $l = 1$ behaves similarly to a confiscatory agent.

Near the time horizon $\mathtt{T} = 8$ prices in Figure 5.5 are high. This feature, combined with the short time that water has to travel downhill, increases the market power of the hydroplant at level 1. This phenomenon is perceptible when comparing the output on the left and right plots in Figure 5.7. In the deterministic case on the left, transferring 30% of the net margin is acceptable to the level in the top. By contrast, for the stochastic model, the fraction jumps $\tau_{2\to1} = \tau_{3\to2} \geq 0.7$. The owner in the top will accept not to play opportunistically and will stop withholding water only if the payment received from the lower level is at least 70% of its net margin. In a somewhat indirect manner, such significant difference gives a quantitative perception of the value of the stochastic solution in our nested trilevel setting, see [Bir82] and [Esc+07].

## Concluding remarks

The issue of market power mitigation in multi-owned hydro cascades is among the main causes that hydro systems did not undergo the same privatization process that thermally dominated systems experienced worldwide. While being a topic of high applied value for countries like Brazil and Canada, it also involves the practical solution of advanced game-theoretic models, some of which do not have effective solution strategies yet, see [FP03]. Our proposal provides a response in that direction, as it guarantees a more efficient management of the overall system. The methodology is shown to terminate finitely for the deterministic and two-stage settings. The approach can be applied to nested multi-stage stochastic programs thanks to the innovative concept of floating cuts defined symbolically in an SDDP framework.

# Chapter 6

# Cut Sharing Across Trees and Efficient Sequential Sampling for SDDP with Uncertainty in the Right-Hand Side

Multistage stochastic optimization problems (MSOP) are a commonly used paradigm to model many decision processes in energy and finance. Usually, a set of scenarios (the so-called tree) describing the stochasticity of the problem is obtained and the Stochastic Dual Dynamic Programming (SDDP) algorithm is often used to compute policies. Quite often, the uncertainty affects only the right-hand side (RHS) of the optimization problems in consideration. After solving an MSOP, one naturally wants to know if the solution obtained depends on the scenarios and by how much. In this chapter we show that when an MSOP with stage-wise independent realizations has only RHS uncertainties, solving one tree using SDDP provides a valid lower bound for all trees with the same number of scenarios per stage without any additional computational effort. The only change to the traditional SDDP is the way cuts are computed. Once the first tree is solved approximately, a computational assessment of the statistical significance of the current number of scenarios per stage is performed, solving for each new sampled tree, an easy LP to get a valid lower bound for the new tree. The objective of the chapter is to estimate how much the lower bound of the first tree depends on randomness. The approach provides fast estimates of the mean, variance and max variation of lower bounds across many trees. If the variance of the computed lower bounds is small, we conclude that the cutting-planes model has a small sensitivity to the trees sampled. Otherwise, we increase the number of scenarios per stage and repeat. We do not make assumptions on the distributions of the random variables. The results are not asymptotic. Our method has applications to the determination of the correct number of scenarios per stage. The stage-wise independence assumption can be dropped as well as the constraint of having only RHS uncertainties. However, the sensitivity of the lower bound with respect to the tree is only for the RHS vectors. Extensions for uncertainties in the objective only are possible via the dual SDDP [Lec+20]. We test our method numerically and verify the correctness of the cut-sharing technique.

## 6.1 Introduction

Multistage stochastic optimization problems (MSOP) have been actively used in energy planning and finance to make decisions in a dynamic and stochastic setting. For big MSOP, one has to choose a compromise between (i) solving the problem in reasonable time and financial costs and (ii) representing uncertainty in detail. Therefore, one naturally asks how to quantify the effect of the scenarios on an MSOP so that it is possible to decide whether or not to improve uncertainty representation given that the solution of one instance of the MSOP in consideration is extremely costly. This chapter helps to deal with this question.

While the general MSOP is computationally intractable, the assumption that the stochastic process has stage-wise independent realizations is reasonable in some settings and induces great simplifications to solution methods via the technique known as cut-sharing. Algorithms leveraging on this technique are the SDDP [PP91] and the CUPPS [CP99].

Cut-sharing is based on realizing that value functions representing future costs in MSOP are the same when the realizations are stage-wise independent. Therefore, there is no need to visit all branches of the tree to improve the current cutting-planes representation of the value functions. This is the source of the enormous reduction on the computational cost per iteration made possible by SDDP. CUPPS went further with the cut-sharing principle by sharing cuts among realizations in the same stage when there is only right-hand side (RHS) uncertainty. Cut-sharing for some forms of inter-stage dependence on the realizations is shown in [IM96].

The most used algorithms for multistage stochastic optimization [PP91; CP99; LW03] consider given a fixed tree of scenarios. However, specially for high dimensional stochastic and volatile processes, the issue of the statistical significance of the solution found using one tree is important [BM06; SM98]. To approximate and evaluate the quality of candidate solutions of problems with continuous distributions many techniques based on solving separately many sampled trees (sequential sampling) have been introduced [MMF11; MMF16; LSW06]. As far as we know, none of these techniques leverage on cut-sharing across trees to make faster inferences. Instead, they develop statistical theory to make inference about solutions found separately by traditional algorithms [PP91; LW03]. In contrast, we get more lower bounding information from the solution of a single tree.

In general sequential sampling, solving many trees from scratch might be unavoidable as a conservative decision to check if the solution obtained is good enough or not. This is so because assumptions of theorems for asymptotic statistics or convergence of optimization methods might not hold. This comes sometimes with heavy costs because SDDP simulations can be quite time consuming and expensive when clusters are used to solve big problems. In this sense, being able to evaluate the sensitivity of general trees with respect at least to RHS vectors might be helpful because it would concentrate computationally hard sampling on other components of the uncertainty. Moreover, the need to solve bigger problems is always standing and therefore efficiency on the solution of important special cases is always welcome.

Consider an SDDP problem that depends on sampled data. We want to stress that sometimes it is *not* practical to solve entirely new SDDP problems for each sample drawn as required by some sequential sampling methods. On the other hand, some estimation of the statistical significance or sensitivity of the lower bound for a given sampled tree is required. Our method can provide this sensitivity at a minor cost. This is clearly desirable information in practice, since the alternative would be either to solve more SDDP problems or trust the result obtained with the first tree. In summary, some sensitivity information might be better than more precise information at very high computational costs or no sensitivity information at all. For instance, in [LSW06; LW03] the authors employ a *computational cluster* to solve many two-stage stochastic problems with RHS uncertainty. Using our technique, we can share cuts between instances and perform a statistical inference cheaper than other proposals.

In some sense, our computational approach is based on trading the costly exact evaluations of optimal values used in sequential sampling with fast estimates of the lower bounds of these optimal values over more trees. The lower bounds computed might be very loose if the total number of scenarios per stage is not enough to represent the continuous problem. However, if we conclude that these lower bounds do not change much across many other trees, we can also conclude that solving more trees from scratch is useless as a strategy to improve current confidence intervals because the first optimal value computed has essentially no sensitivity to the tree being used. In this sense, our contribution is more a tool to state that the true model is reasonably approximated. Nonetheless, the lower bounds can be used as a proxy to build confidence intervals for the true optimal value.

Recently, the topic of sensitivity and duality of multistage stochastic optimization gained some attention [Lec+20; GSC20; TW20] with the development of the dual SDDP algorithm (see also [HS06; Roc99]). The motivation of these new developments was to improve the computation of the upper bound on the traditional SDDP method as also considered in [PMF13]. The method of [Lec+20] focuses only on RHS uncertainty as we do, while [GSC20] is for general stage-wise independent uncertainty. In principle, [Lec+20] should have applications in efficient sequential sampling for MSO. Using the dual SDDP one can apply our techniques to get fast upper bound estimates on the optimal value of a tree under perturbations of the cost vectors as was initially pursued using other strategies in [BCC12] for the sensitivity of energy contracts to some prices.

The approach used in [BCC12] and also in [GSC20] is based on using Danskin's theorem [BS00] to estimate directional derivatives of the optimal value. However, they obtain a lower estimate on the directional derivative using the primal optimal solution computed, because Danskin's theorem requires the knowledge of *all* primal optimal solutions. Precisely, given a nonempty compact set $Z$ and a directionally differentiable function $\phi(x,z)$ that is convex in $x$ for all $z$, we can define the value function

$$f(x) = \max\{\phi(x,z) : z \in Z\}$$

and the associated set of optimal solutions

$$Z^*(x) = \{z \in Z : \phi(x,z) = f(x)\}.$$

Danskin's theorem states that the directional derivative $f'(x;d)$ of $f$ in the direction $d$ is given by

$$f'(x;d) = \max\{\phi'(x,z;d) : z \in Z^*(x)\}.$$

This approach is related to ours as we further explain now. Instead of trying to estimate the directional derivative of the optimal value with respect to problem data, we estimate a *global* lower bound on the optimal value with a max-type function (the cutting-planes model). The cutting-planes model is convex because it is given by a maximum of finitely many affine functions as in

$$m(x) = \max_i \{a_i + b_i^\intercal x : i = 1, \dots, I\}.$$

As is well known, the directional derivatives of max-type functions can be computed easily identifying the so-called active set (explained in Section 6.2). Therefore, we can also get an estimate on the directional derivative of the true value function given a specific direction similarly to [BCC12]. Instead of fixing a direction, we are actually interested in evaluating the global lower bounds given by the cutting-planes model as we expect these lower bounds to show small variance as a sign of good approximation of the true problem.

Moreover, it seems to us that using the individual estimates of the directional derivatives at all directions as in [BCC12] could be suboptimal when there are lots of possible directions because (i) analyzing more numbers is harder and (ii) the joint effect of the directional derivatives is better understood when they are plugged together in a cutting-planes model that gives one single estimate for the optimal value as opposed to Tables 6 and 7 of [BCC12].

In summary, our method has some similarities with [BCC12], but with more focus on cutting-plane models and calculations of guaranteed lower bounds. The cutting-planes are used to summarize nonsmooth derivative information, jointly with a strategy to replicate RHS parameters backwards explained opportunely. This strategy makes the lower bound calculations we advertise possible at the expense of an increase in memory usage proportional to the number of stages and scenarios of the discretized SDDP problem. The same idea of free-floating cuts that we employ here to build the algorithm is also used to solve some multilevel and multistage stochastic equilibrium problems [Bor+21].

In Section 6.2 we make some preliminary considerations. In Section 6.3 we show our approach for the two-stage case. In Section 6.4 we show the multistage case. In Section 6.5 we report the numerical experiments supporting our methods. In Section 6.6 we show extensions based on the dual SDDP.

## 6.2 Preliminaries

The subdifferential [RW09] of a finite-valued convex function $P$ at a point $z$ is defined as

$$\partial P(z) = \{\gamma : P(z') \geq P(z) + \gamma^\intercal (z' - z) \quad \forall z'\}.$$

The directional derivative $P'(z;d)$ of $P$ at $z$ and a direction $d$ always exists and is such that

$$P'(z;d) = \max\{\gamma^\intercal d : \gamma \in \partial P(z)\}.$$

We denote by conv $D$ the smallest convex set containing $D$. The connection of the directional derivative of $P$ can be made with the cutting-planes model as follows. If we consider a max-type function

$$P(z) = \max\{f_j(z) : j = 1, \dots, J\}$$

where $f_j$ are smooth convex functions, it is well known that

$$\partial P(z) = \operatorname{conv}\{\nabla f_j(z) : f_j(z) = P(z)\}.$$

In this section we consider the widely used convex value function given by

$$Q(x,h) := \min_{y \geq 0} \{c^\intercal y : Wy = Rh - Bx\}. \tag{6.1}$$

Since $Q$ is finite-valued (by assumption) and the underlying optimization problem is linear, we know there are optimal primal and dual solutions. For a fixed pair $x = \hat{x}$ and $h = \hat{h}$ we can compute an optimal Lagrange multiplier $\hat{\lambda}$ of the equality constraints associated with the optimization problem defining $Q(\hat{x}, \hat{h})$. Doing so for a fixed $h = \hat{h}$ yields the widely used formula

$$Q(x, \hat{h}) \geq Q(\hat{x}, \hat{h}) - \hat{\lambda}^\intercal B(x - \hat{x}) \quad \forall x. \tag{6.2}$$

Nonetheless, with the same multiplier $\hat{\lambda}$ already obtained we can compute the more general inequality

$$Q(x,h) \geq Q(\hat{x}, \hat{h}) - \hat{\lambda}^\intercal B(x - \hat{x}) + \hat{\lambda}^\intercal R(h - \hat{h}) \quad \forall x, \quad \forall h. \tag{6.3}$$

It is worth noting that inequalities (6.2) and (6.3) are equivalent. First, fixing $h = \hat{h}$ on (6.3) yields (6.2), which proves that (6.3) implies (6.2). Conversely, since (6.2) holds for any linear problem with the same structure, we can change the problem data in (6.1) so that we obtain (6.3) as an application of (6.2). Let us take $z = (x,h)$, $\tilde{B} = (B,-R)$ and $\tilde{R} = 0$. Then, applying (6.2) to

$$\tilde{Q}(w,g) := \min_{y\geq 0}\{c^\mathsf{T} y : Wy = \tilde{R}g - \tilde{B}z\}$$

at the point $\hat{z} = (\hat{x},\hat{h})$ and $\hat{g} = 0$ gives a multiplier $\hat{\lambda}$ such that

$$\tilde{Q}(z,\hat{g}) \geq \tilde{Q}(\hat{z},\hat{g}) - \hat{\lambda}^\mathsf{T}\tilde{B}(z-\hat{z}) \quad \forall z. \tag{6.4}$$

Substituting $z = (x,h)$ and using $\tilde{B} = (B,-R)$, we obtain from (6.4) inequality (6.3).

Therefore, defining $Q_{\hat{h}}(x) = Q(x,\hat{h})$, it follows respectively, from (6.2) and (6.3), that

$$-B^\mathsf{T}\hat{\lambda} \in \partial Q_{\hat{h}}(\hat{x}) \quad \text{and} \quad (-B^\mathsf{T}\hat{\lambda}, R^\mathsf{T}\hat{\lambda}) \in \partial Q(\hat{x},\hat{h}).$$

Let us assume now that $Q(x,h)$ represents a cost-to-go function [PP91; CP99] used in a cutting-planes method (CPM). In this case, $x$ is a decision from the master problem of the CPM and $h$ is a scenario (problem data). The value of $h$ is not decided inside the CPM, only the value of $x$. Schematically, we consider in this chapter a family of nonsmooth convex problems indexed by $h$ and given by

$$\text{CPM}(h): \quad \min_x \quad Q(x,h) \quad \text{s.t.} \quad x \in D. \tag{6.5}$$

The algorithms proposed consist of solving a base problem CPM($\hat{h}$), and then inferring the optimal values of the other problems CPM($h$) with a minor computational cost. For any $h = \tilde{h}$, the cuts actually used for solving or estimating the optimal value of (6.5) are

$$Q(x,\tilde{h}) \geq U(\hat{x},\hat{h},\tilde{h}) - \hat{\lambda}^\mathsf{T}B(x-\hat{x}) \quad \forall x, \quad \text{where} \quad U(\hat{x},\hat{h},\tilde{h}) = Q(\hat{x},\hat{h}) + \hat{\lambda}^\mathsf{T}R(\tilde{h}-\hat{h}). \tag{6.6}$$

Although we do not present the statistical theory for our approach, it can be easily explained by analogy with the mean and variance of the empirical mean that are related via Chebyshev's Theorem. Let $(\Omega,\mathscr{A},\mathbb{P})$ be a probability space and $\{X_s : s \in \mathbb{N}\}$ be a sequence of independent and identically distributed random variables with finite mean $\mathbb{E}X_s = \mathscr{Q}$ and finite variance $\operatorname{Var} X_s = \sigma^2$. The empirical mean with $S$ samples is given by

$$\overline{X} = \frac{X_1 + \cdots + X_S}{S}. \tag{6.7}$$

It is well known that

$$\mathbb{E}\overline{X} = \mathscr{Q}, \quad \operatorname{Var}\overline{X} = \frac{\sigma^2}{S} \quad \text{and} \quad \mathbb{P}(|X_s - \mathscr{Q}| \geq k\sigma) \leq \frac{1}{k^2}. \tag{6.8}$$

Considering $h$ as a random vector, we can consider its replication as a sequence $\{h_s : s \in \mathbb{N}\}$ so that $h_s$ form an independent and identically distributed sequence such that $h_s$ has the same distribution as $h$. For the sake of connecting (6.7) and (6.8) with the work developed in this chapter, we take $X_s = Q(x,h_s)$, where $x$ is deterministic. It turns out that (6.7) is the expected recourse function minimized in Section 6.3 after drawing a sample of size $S$ denoted by $(\hat{h}_1,\ldots,\hat{h}_S)$. Using this information reveals that $\mathbb{E}\overline{X} = \mathscr{Q}(x)$. The sample average approximation used on this work consists in changing problem

$$\min_x \quad \mathscr{Q}(x) \quad \text{s.t.} \quad x \in D \tag{6.9}$$

by problem

$$\min_x \quad S^{-1}\sum_s Q(x,\hat{h}_s) \quad \text{s.t.} \quad x \in D. \tag{6.10}$$

The basis for replacing (6.9) by (6.10) is the assumption that $\min_x \mathbb{E}\overline{X} = \mathbb{E}(\min_x \overline{X})$. In other words, if the minimization and expectation operators can be interchanged, then the expected optimal value of (6.10) is the optimal value of (6.9). This chapter provides a technique to compute a lower bound for the true optimal value $\min_x \mathscr{Q}(x)$ if these operators can be exchanged.

While (6.9) cannot be solved in general, problem (6.10) can in most cases. However, we are left with the dependency on the sample $(\hat{h}_1,\dots,\hat{h}_S)$, which does not appear on (6.9). Therefore, we would expect that if indeed (6.10) approximates (6.9), then we would observe that the sensitivity of (6.10) with respect to $(\hat{h}_1,\dots,\hat{h}_S)$ becomes smaller when $S$ grows big enough. This is indeed verified experimentally on Section 6.5.

Note that (6.10) resembles (6.5), which means that with the developments of this chapter we can analyze empirically how well (6.10) approximates (6.9) by understanding how the optimal value of (6.10) changes when the sample $(\hat{h}_1,\dots,\hat{h}_S)$ changes. The later entails estimating the optimal values of a family of problems indexed by $h=(h_1,\dots,h_S)$ as described on (6.5). This estimation has to be performed with reasonable computational costs and time.

## 6.3 Two-stage stochastic problems

The traditional two-stage stochastic problem (2TSP) consists of $S$ scenarios in the second stage represented by a sequence $(h_{21},\dots,h_{2S})$ of RHS vectors. An actual sample of this RHS sequence is denoted by $(\hat{h}_{21},\dots,\hat{h}_{2S})$. It consists in solving

$$\min_x \quad c_1^\intercal x + S^{-1}\sum_{s=1}^{S} Q_s(x,\hat{h}_{2s}) \quad \text{s.t.} \quad x\in D_1 := \{x\geq 0 : W_1x = h_1\} \tag{6.11}$$

where for $s=1,\dots,S$ we take

$$Q_s(x,h_{2s}) := \min_{y_s\geq 0}\{c_2^\intercal y_s : W_2y_s = h_{2s} - B_2x\}.$$

For convenience, it is useful to define the full cost-to-go function given by

$$Q(x,h_{21},\dots,h_{2S}) := S^{-1}\sum_{s=1}^{S} Q_s(x,h_{2s}). \tag{6.12}$$

The 2TSP with continuous expectation in the objective is considered in [SM98; BM06]. To solve the continuous problem, often one tries to analyze (6.11) for different samples $(\hat{h}_{21},\dots,\hat{h}_{2S})$ and values $S$. Our procedure consists in analyzing fast and empirically if the optimal value of (6.11) depends on the realization of the sample $(\hat{h}_{21},\dots,\hat{h}_{2S})$ for a fixed $S$ because the expectation of the true problem, as shown in (6.9), is not sample-dependent.

Note that although we do not consider $(c,W,B)$ dependent on $s$, we could have done so. However, our technique would apply only for the sensitivity of the RHS vectors. In practice, this possibility is already accounted for on the notation $Q_s$ that expresses the value function of the scenario problems separately. This is performed by taking scenarios $(c_{2s},W_{2s},B_{2s})$ and setting

$$Q_s(x,h_{2s}) := \min_{y_s\geq 0}\{c_{2s}^\intercal y_s : W_{2s}y_s = h_{2s} - B_{2s}x\}.$$

The resulting value function $Q_s(x,h_{2s})$ is still jointly convex with respect to $x$ and $h_{2s}$. The cut calculation is then performed as usual. The values of $(c_{2s},W_{2s},B_{2s})$ can change across $s$, since the cuts computed for each $s$ are valid approximations for $Q_s$ and do not interfere with each other. On what follows we consider $(c,W,B)$ independent of $s$.

The algorithm for the sensitivity of 2TSPs with respect to the sample $(\hat{h}_{21},\dots,\hat{h}_{2S})$ as explained by (6.9) and (6.10) and nearby text is shown below. It takes as input the matrices defining the value functions $Q_s(x,h_{2s})$ and the output is a vector of estimates of the optimal value of problem (6.10) for other samples.

**Algorithm 6.3.1** (Sensitivity of 2TSPs with respect to RHS vectors)**.**

```
**Initialization.** Set $k = 1$ and take $\varepsilon > 0$. Take $S$ and a base sample $(\hat{h}_{21}, \dots, \hat{h}_{2S})$.
**Step 1: Solve Sample Problem using Modified Cuts.**
    **Initialization.** Take $x^k \in D$.
    **Step 1.1: Get Subgradient.** For $s = 1, \dots, S$ compute $(\alpha_s^k, \beta_s^k) \in \partial Q_s(x^k, \hat{h}_{2s})$ based on (6.3).
    **Step 1.2: Define the Scenario Model.** Take
        $Q_s^k(x, h_{2s}) := \max_{i=1,\dots,k} \{Q_s(x^i, \hat{h}_{2s}) + (\alpha_s^i)^\intercal (x - x^i)\} + (\beta_s^i)^\intercal (h_{2s} - \hat{h}_{2s})\}.$   (6.13)
    **Step 1.3: Define the Aggregate Model.** Take
        $Q^k(x, h_{21}, \dots, h_{2S}) := S^{-1} \sum_{s=1}^{S} Q_s^k(x, h_{2s}).$
    **Step 1.4: Next Iterate.** Compute $x^{k+1} \in \arg\min_x \quad c_1^\intercal x + Q^k(x, \hat{h}_{21}, \dots, \hat{h}_{2S}) \quad \text{s.t.} \quad x \in D_1.$
    **Step 1.5: CPM Stopping Test.** Go to Step 2 if
        $Q(x^{k+1}, \hat{h}_{21}, \dots, \hat{h}_{2S}) - Q^k(x^{k+1}, \hat{h}_{21}, \dots, \hat{h}_{2S}) \leq \varepsilon.$
    **Step 1.6: Loop.** Set $k = k + 1$ and go back to Step 1.1.
**Step 2: Evaluate Fast Lower Bounds.** For $l = 1, \dots, L$ sample $(\hat{h}_{21}^l, \dots, \hat{h}_{2S}^l)$ and compute
    $\hat{v}^l := \min_x \quad c_1^\intercal x + Q^k(x, \hat{h}_{21}^l, \dots, \hat{h}_{2S}^l) \quad \text{s.t.} \quad x \in D_1.$   (6.14)
**Step 3: Output.** Compute statistics of interest over $\{\hat{v}^l : l = 1, \dots, L\}$.
```

Step 2 of Algorithm 6.3.1 consists in solving many LPs. For the current standards, this is not something to be concerned about. Nonetheless, step 2 can be further simplified computing one lower bounding cut for the value function of the first stage problem as a function of the scenarios $(h_{21}, \dots, h_{2S})$ so that $\hat{v}^l$ would be estimated evaluating an affine expression, which is absolutely inexpensive.

Precisely, the value function of problem (6.14) is convex with respect to $(h_{21}, \dots, h_{2S})$ and has the same structure of problem (6.1). Denoting it by $U^k(h_{21}, \dots, h_{2S})$, it follows that we can calculate subgradients

$$(\sigma_1^k, \dots, \sigma_S^k) \in \partial U^k(\hat{h}_{21}, \dots, \hat{h}_{2S})$$

such that

$$U^k(h_{21}, \dots, h_{2S}) \geq U^k(\hat{h}_{21}, \dots, \hat{h}_{2S}) + \sum_{s=1}^{s} (\sigma_s^k)^\intercal (h_{2s} - \hat{h}_{2s}) \quad \forall h_{21}, \dots, h_{2S}. \tag{6.15}$$

Therefore, instead of estimating $\hat{v}^l$ as $U^k(\hat{h}_{21}^l, \dots, \hat{h}_{2S}^l)$, we estimate it as the right-hand side of (6.15), which has the cost of evaluating a linear expression. The dependency of $U^k(\hat{h}_{21}^l, \dots, \hat{h}_{2S}^l)$ on $h_1$ can also enter on (6.15).

Note that the actual cuts appearing in step 1.4 are equal to the traditional ones based on formula (6.2) because $Q^k(x, h_{21}, \dots, h_{2S})$ is evaluated at $(\hat{h}_{21}, \dots, \hat{h}_{2S})$ and this makes the free-floating part mentioned in (6.3) vanish, as we further explain now. Because the sample $(\hat{h}_{21}, \dots, \hat{h}_{2S})$ is fixed for all $k$ we have

$$Q^k(x, \hat{h}_{21}, \dots, \hat{h}_{2S}) = S^{-1} \sum_{s=1}^{S} Q_s^k(x, \hat{h}_{2s})$$

and by (6.13) it follows that

$$\begin{aligned} Q_s^k(x, \hat{h}_{2s}) &:= \max_{i=1,\dots,k} \{Q_s(x^i, \hat{h}_{2s}) + (\alpha_s^i)^\intercal (x - x^i)\} + (\beta_s^i)^\intercal (\hat{h}_{2s} - \hat{h}_{2s})\} \\ &= \max_{i=1,\dots,k} \{Q_s(x^i, \hat{h}_{2s}) + (\alpha_s^i)^\intercal (x - x^i)\}\}. \end{aligned}$$

Therefore, the inner CPM inside step 1 finishes with finitely many iterations under mild conditions [Sha11; Kel60]. There is at least one modification of Algorithm 6.3.1 that one is tempted to consider that is to improve the representation of $Q(x,h_{21},\dots,h_{2S})$ using more base scenarios $(\hat{h}_{21},\dots,\hat{h}_{2S})$. This would translate into repeating step 1 some times. However, at least for our experience, this does not help much because the statistical properties of $\{\hat{v}^l : l=1,\dots,L\}$ hardly change. The free-floating cuts we propose also allow to re-sample the base scenario inside step 1 and make one forward-backward (Step 1.1 to Step 1.6) for each new base scenario. While all these modifications are possible, we find that the algorithm as presented provides better results.

As commented briefly in the introduction, the free-floating cut requires storing much more coefficients. For instance, in the two-stage case we would have to store $(\alpha_s^i,\beta_s^i)$ for all $s=1,\dots,S$ and $i=1,\dots,k$, which uses much more memory. This might indeed be a problem if only primary memory (RAM) is used. However, as we explain now, it is possible to break free of this limitation storing cuts on secondary memory (files) because it is much cheaper and exists in greater quantity.

First, recall that LPs are solved on primary memory. For the sake of simplicity assume that Algorithm 6.3.1 is implemented without any parallelization. In this case, the minimum memory required to run Algorithm 6.3.1 is the memory to solve the LPs. Now, notice that this amount of memory is not affected by the additional coefficients $\beta_s^i$ because those influence only the free term of the cut, as can be noted by analogy examining (6.6). In other words, for any $s=1,\dots,S$ and $i=1,\dots,k$, the cuts actually used on (6.14) are given by

$$Q(x,\hat{h}_{21}^l,\dots,\hat{h}_{2S}^l) \geq U(x^i,\hat{h},\hat{h}^l) + \frac{1}{S}\sum_{s=1}^{S}(\alpha_s^i)^\intercal(x-x^i) \quad \forall x \tag{6.16}$$

where $U(x^i,\hat{h},\hat{h}^l)$ is a number calculated as

$$U(x^i,\hat{h},\hat{h}^l) = Q(x^i,\hat{h}_{21},\dots,\hat{h}_{2S}) + \frac{1}{S}\sum_{s=1}^{S}(\beta_s^i)^\intercal(\hat{h}_{2s}^l-\hat{h}_{2s}). \tag{6.17}$$

Therefore, the coefficients $\beta_s^i$ do not need to be stored on the main memory, because they are not directly used during the solution of the LPs. It is enough to store them on a file as they are obtained during the iterations of Step 1 of Algorithm 6.3.1 and read the respective file to calculate the intercepts of the cuts as needed along the process. Although this is possible, we use only main memory during our experiments, as they are intended only for the illustration of the methods.

To finish this section, we provide an example that can be solved by hand and shows the same features that we observe on the experiments of Section 6.5.

**Example 6.3.2.** Take a real random variable $h$ following a uniform distribution on $[0,1]$ and define

$$Q(x,h) = \min_{y\geq 0}\{y : y \geq x-h, y \geq h-x\} = |x-h|.$$

By definition, we have $\mathscr{Q}(x) = \mathbb{E}_h|x-h|$. For any $x\in[0,1]$, it follows that

$$\mathscr{Q}(x) = \int_0^x (x-h)dh + \int_x^1 (h-x)dh = x^2 - x + \frac{1}{2}.$$

The solution to the problem

$$\min_{x\in[0,1]} \quad \mathscr{Q}(x)$$

is $x=0.5$ and optimal value is 0.25. Moreover, such optimal value does not depend on $h$. Now, let us consider a replication of $h$ as an independent and identically distributed sequence $\{h_s : s\in\mathbb{N}\}$ such that $h_s \sim h$. The sample average approximation is given by

$$\min_{x\in[0,1]} \quad \frac{1}{S}\sum_{s=1}^{S} Q(x,\hat{h}_s) \quad = \quad \frac{1}{S}\sum_{s=1}^{S}|x-\hat{h}_s|.$$

Without loss of generality, let us assume that $\hat{h}_s \leq \hat{h}_{s+1}$ for all $s=1,\dots,S-1$. If $S$ is odd, it follows that the median value of the sequence $\{\hat{h}_s\}$ solves the sample average problem. If $S$ is even, the situation is a bit more tricky. For instance, if $S=2$, $\hat{h}_1=0$ and $\hat{h}_2=1$, then any $x\in[0,1]$ solves the problem. In general, we take the only $s$ such that the median of $\{\hat{h}_s\}$ lies inside $[\hat{h}_s,\hat{h}_{s+1}]$. Then, any $x\in[\hat{h}_s,\hat{h}_{s+1}]$ solves the sample average problem. Therefore, the optimal value of $x$ tends to be close to 0.5 when $S$ is big.

Step 1 of Algorithm 6.3.1 starts with $k = 1$ and $x^k = 0$. The cut obtained by applying (6.3) for each scenario $s$ at the first iteration is

$$h_s - x = \hat{h}_s + 1^\intercal(h_s - \hat{h}_s) - 1^\intercal(x - 0) = Q(0, \hat{h}_s) - 1^\intercal(h_s - \hat{h}_s) - 1^\intercal(x - 0)$$

because the active constraint is $y \geq \hat{h}_s - x$ with associated multiplier 1. After averaging across $s$ we obtain $\frac{1}{S}\sum_{s=1}^{S} h_s - x$. At the second iteration ($k = 2$) we obtain $x^k = 1$ and for each $s$ the cut is

$$x - h_s = 1 - \hat{h}_s - 1^\intercal(h_s - \hat{h}_s) + 1^\intercal(x - 1) = Q(1, \hat{h}_s) - 1^\intercal(h_s - \hat{h}_s) + 1^\intercal(x - 1)$$

because the active constraint is $y \geq x - \hat{h}_s$ with multiplier 1. After averaging across $s$ we obtain the cut $x - \frac{1}{S}\sum_{s=1}^{S} h_s$. After applying the values of $h_s = \hat{h}_s$, the master problem (Step 1.4) at the third iteration is

$$\min_{x,r} \quad r \quad : \quad x \in [0,1], \quad r \geq \frac{1}{S}\sum_{s=1}^{S} \hat{h}_s - x, \quad r \geq x - \frac{1}{S}\sum_{s=1}^{S} \hat{h}_s.$$

Therefore, the third iterate of $x$ is the average of $\hat{h}_s$, which is already a good estimate of the optimal solution. However, the estimate of the optimal value is still bad. The next cuts added already make the gap (Step 1.5) small, however they start showing complicated cancellation effects because, contrary to the first two iterations, for some scenarios the constraint $y \geq x - \hat{h}_s$ is active and for other scenarios the other constraint $y \geq \hat{h}_s - x$ is active.

Finally, computing the histogram of the realizations of the optimal value with Step 2 of Algorithm 6.3.1 with $L = 2000$ we obtain Figure 6.1. We perform 20 iterations on Step 1. Note that even for $S = 10$ (left-most figure), the average of the realizations is close to the true optimal value 0.25. While we do not explore mathematically this behavior, we also observe on Section 6.5 that the center of the histograms on Figure 6.4 is stable across $S$. We also observe that the histograms are collapsing to a distribution concentrated at the point 0.25 when $S$ grows, which is the distribution of the true optimal value (a deterministic value).

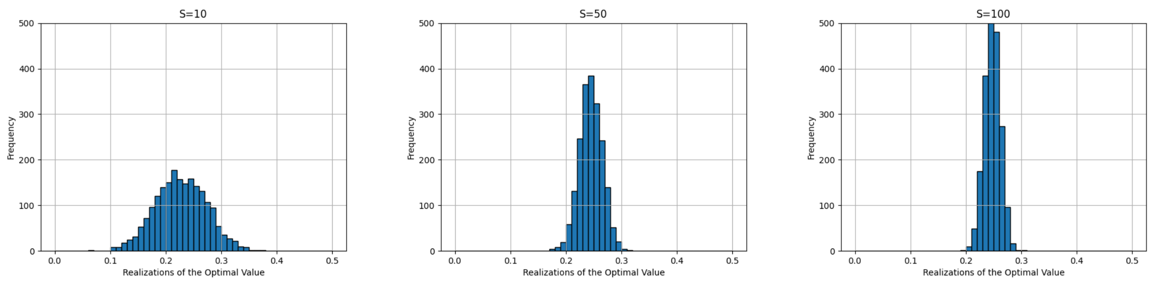

Figure 6.1: Histogram of the realizations of the optimal values of Step 2 of Algorithm 6.3.1 with $L = 2000$.

## 6.4 Multistage stochastic problems

In this section we show how to perform the sensitivity of the results of the SDDP algorithm with respect to the RHS vectors. Naturally, the method is the extension of the one for the two-stage case applied recursively and backwards. In the last section, the cost-to-go function at $t = 1$ also depends on $(h_{21}, \ldots, h_{2S})$. Analogously, for the SDDP, the cost-to-go function at stage $t$ also depends on all the RHS vectors of the forward problems. With $S$ stage-wise independent scenarios per stage and a total of $T$ stages, these vectors can be represented by a sequence (or tuple) denoted by

$$h[t] := (h_{\tau s} \quad : \quad \tau = t+1, \ldots, T, \quad s = 1, \ldots, S). \tag{6.18}$$

We understand $h[t]$ as an empty tuple if $t \geq T$. Again, the actual sampling of these vectors is denoted by $\hat{h}[t]$. The first stage feasible set is the same as in the two-stage case and the feasible sets for $t = 2, \ldots, T$ and $s = 1, \ldots, S$ are given by

$$D_{ts}(x_{t-1}, h_{ts}) := \{x_{ts} \geq 0 \quad : \quad W_t x_{ts} = h_{ts} - B_t x_{t-1}\}.$$

Note that the feasible set $D_{ts}$ depends also on $h_{ts}$. The first stage problem is given by

$$\min_{x_1} \quad c_1^\intercal x_1 + Q_2(x_1, h[1]) \quad \text{s.t.} \quad x_1 \in D_1$$

where for $t = 2,\dots,T$ and $s = 1,\dots,S$ we have

$$\mathcal{Q}_t(x_{t-1},h[t-1]) := S^{-1}\sum_{r=1}^{S} Q_{tr}(x_{t-1},h_{ts},h[t]) \tag{6.19}$$

and

$$Q_{ts}(x_{t-1},h_{ts},h[t]) := \min_{x_{ts}} \quad c_t^\mathsf{T} x_{ts} + \mathcal{Q}_{t+1}(x_{ts},h[t]) \quad \text{s.t.} \quad x_{ts}\in D_{ts}(x_{t-1},h_{ts})$$

with

$$\mathcal{Q}_{T+1}(\cdot,\cdot)\equiv 0.$$

For instance, note that (6.19) generalizes (6.12). Again, we do not take $(c_t,W_t,B_t)$ depending on the scenario. However, as already explained we could have taken, in which case our method would be applied to the RHS sensitivity only. For both cases, the notation $Q_{ts}(x_{t-1},h_{ts},h[t])$ is general enough. As usual, our method is also based on building polyhedral approximations of $Q_{ts}$ denoted $Q_{ts}^k$ with the upper index $k$. The aggregate value function used on the algorithm below is defined for all $k$ by

$$\mathcal{Q}_t^k(x_{t-1},h[t-1]) := S^{-1}\sum_{r=1}^{S} Q_{tr}^k(x_{t-1},h_{ts},h[t]). \tag{6.20}$$

**Algorithm 6.4.1** (Sensitivity of SDDP problems with respect to RHS vectors)**.**

**Initialization.** Take a tolerance $\varepsilon \geq 0$. Set $k = 1$. Take $S$ and a base sample $\hat{h}[1]$. Take $M > 0$ large. Set $Q_{ts}^k(\cdot,\cdot,\cdot)\equiv -M$ for all $t = T,\dots,2$ and all $s = 1,\dots,S$. For all $t = T,\dots,2$ the functions $\mathcal{Q}_t^k(\cdot,\cdot)$ are given by formula (6.20).

**Step 1: Solve Sample SDDP Problem using Modified Cuts.**

**Step 1.1: Sample Scenario.** Sample a scenario $\omega_t^k \in \{1,\dots,S\} \quad \forall t = 2,\dots,T$.

**Step 1.2: Modified Forward.** Compute

$$\hat{x}_1^k \in \arg\min_{x_1} \quad c_1^\mathsf{T} x_1 + \mathcal{Q}_{12}^k(x_1,\hat{h}[1]) \quad \text{s.t.} \quad x_1\in D_1$$

and for $t = 2,\dots,T$ and $s = \omega_t^k$ compute

$$\hat{x}_t^k \in \arg\min_{x_{ts}} \quad c_t^\mathsf{T} x_{ts} + \mathcal{Q}_{t+1}^k(x_{ts},\hat{h}[t]) \quad \text{s.t.} \quad x_{ts}\in D_{ts}(\hat{x}_{t-1}^k,\hat{h}_{ts}).$$

**Step 1.3: Modified Backward.** For all $t = T,\dots,2$ and all $s = 1,\dots,S$ compute

$$x_{ts}^k \in \arg\min_{x_{ts}} \quad c_t^\mathsf{T} x_{ts} + \mathcal{Q}_{t+1}^{k+1}(x_{ts},\hat{h}[t]) \quad \text{s.t.} \quad x_{ts}\in D_{ts}(\hat{x}_{t-1}^k,\hat{h}_{ts})$$

where $\mathcal{Q}_{T+1}^{k+1}(\cdot,\cdot)\equiv 0$ and for $t = T-1,\dots,1$ take $\mathcal{Q}_{t+1}^{k+1}(\cdot,\cdot)$ given by formula (6.20). For $t = T-1,\dots,1$ and $s = 1,\dots,S$ take $Q_{t+1,s}^{k+1}(\cdot,\cdot,\cdot)$ as a maximum between $Q_{t+1,s}^{k}(\cdot,\cdot,\cdot)$ and a free-floating cut of $Q_{t+1,s}(\cdot,\cdot,\cdot)$ at the point $(\hat{x}_t^k,\hat{h}_{t+1,s},\hat{h}[t+1])$ as explained in (6.3). Precisely, for each $t = T-1,\dots,1$ and $s = 1,\dots,S$ calculate subgradients

$$(\alpha_{tsk},\beta_{t+1,sk},\gamma_{t+1,sk}) \in \partial Q_{t+1,s}(\hat{x}_t^k,\hat{h}_{t+1,s},\hat{h}[t+1])$$

and take $Q_{t+1,s}^{k+1}(\cdot,\cdot,\cdot)$ as a maximum between $Q_{t+1,s}^{k}(\cdot,\cdot,\cdot)$ and the affine function

$$Q_{t+1,s}(\hat{x}_t^k,\hat{h}_{t+1},\hat{h}[t+1]) + \alpha_{tsk}^\mathsf{T}(x_t-\hat{x}_t^k) + \beta_{t+1,sk}^\mathsf{T}(h_{t+1}-\hat{h}_{h+1,s}) + \gamma_{t+1,sk}^\mathsf{T}(h[t+1]-\hat{h}[t+1]).$$

**Step 1.4: Lower Bound.** Compute

$$x_1^k \in \arg\min_{x_1} \quad c_1^\mathsf{T} x_1 + \mathcal{Q}_2^{k+1}(x_1,\hat{h}[1]) \quad \text{s.t.} \quad x_1\in D_1. \tag{6.21}$$

Set $\underline{v}_k$ as the optimal value of (6.21) and $\overline{v}_k = \sum_{t=1}^T c_t^\mathsf{T}\hat{x}_t^k$.

**Step 1.5: Stopping Test.** Go to Step 2 if the lower bound $\underline{v}_k$ stabilized across $k$ or if the average forward cost $\overline{v}_i$ for $i = 1, \dots, k$ is close enough to the lower bound $\overline{v}_k$ as expressed by

$$\frac{1}{k}\sum_{i=1}^{k}\overline{v}^i \leq \varepsilon + \underline{v}_k.$$

**Step 1.6: Loop.** Set $k = k+1$ and go back to Step 1.1.

**Step 2: Evaluate Fast Lower Bounds.** For $l = 1, \dots, L$ sample a new sequence $\hat{h}^l[1]$ and compute

$$\hat{v}_1^l := \min_{x_1} \quad c_1^\mathsf{T} x_1 + Q_1^k(x_1, \hat{h}^l[1]) \quad \text{s.t.} \quad x_1 \in D_1.$$

**Step 3: Output.** Compute statistics of interest over $\{\hat{v}_1^l : l = 1, \dots, L\}$.

As explained previously for the two-stage case, some variations of Algorithm 6.4.1 could have been considered. Among the most obvious possibilities, we could run step 1 more times for more base samples $\hat{h}[1]$. We could also have proposed to sample trees inside step 1 and run one forward-backward pass (Step 1.1 to Step 1.6) for each new tree. Note that this is all possible within the free-floating cuts approach. Because the number of possible discretizations of the true SDDP is far greater than our ability to solve at least a minor portion of them, running step 1 more times re-using cuts already computed tends not to change the results in step 2. On the other hand, the sampling in step 2 can be quite exhaustive because $L$ can be quite big. The problem with sampling trees inside Step 1 is that the stopping test inside Step 1.5 would be arbitrary, as opposed to the current of having solved the base tree sampled in the start and apply commonly used stopping tests for the traditional SDDP.

The convergence proof for the SDDP method inside step 1 is the same of the traditional case [Sha11] because the polyhedral approximations $Q_{t+1}^k(x_{ts}, \hat{h}[t])$ are the same. This is so because the free-floating parts of the cuts are evaluated exactly at the base point $h = \hat{h}$, using formula (6.3) for the analogy. The contribution of the free-floating part of the cuts is only at step 2. More precisely, for any $t \in \{1, \dots, T-1\}$, $s \in \{1, \dots, S\}$ and $i \in \{1, \dots, k\}$ there are subgradients

$$(\alpha_{tsi}, \beta_{t+1,si}, \gamma_{t+1,si}) \in \partial Q_{t+1,s}(\hat{x}_t^i, \hat{h}_{t+1,s}, \hat{h}[t+1])$$

such that $Q_{t+1,s}^k(x_t, h_{t+1}, h[t+1])$ is a maximum for $i = 1, \dots, k$ of the affine functions

$$Q_{t+1,s}(\hat{x}_t^i, \hat{h}_{t+1}, \hat{h}[t+1]) + \alpha_{tsi}^\mathsf{T}(x_t - \hat{x}_t^i) + \beta_{t+1,si}^\mathsf{T}(h_{t+1} - \hat{h}_{h+1,s}) + \gamma_{t+1,si}^\mathsf{T}(h[t+1] - \hat{h}[t+1]). \tag{6.22}$$

Since Step 1 of Algorithm 6.4.1 evaluates (6.22) at $h_{t+1} = \hat{h}_{t+1,s}$ and $h[t+1] = \hat{h}[t+1]$, we conclude that Step 1 is exactly as the traditional SDDP method. Therefore, the convergence properties indeed follow from [Sha11].

In the multi-stage case, the issue with memory usage explained on Section 6.3 becomes critical because memory complexity is quadratic (in the computer science sense). Note that the dimension of the cuts at stage $t$ is about the same of $h[t]$. Because the dimensions of $h[t]$ increase when $t$ runs backwards, we get an additive effect as in the arithmetic progression that leads to quadratic memory complexity. Precisely, as can be seen in (6.18), the dimension of $h[t]$ is about $S(T-t)H$, where $H$ is the dimension of the samples $\hat{h}_{ts}$. Therefore, the memory complexity for storing the cuts on main memory from time $T$ to time $t$ for one iteration is $\mathcal{O}(S(T-t)^2)$. Then, the total memory complexity for one iteration is $\mathcal{O}(ST^2)$. These estimates assume that the cuts are aggregated across scenarios, otherwise, if they are stored separately for each scenario, we get instead $\mathcal{O}(S^2T^2)$ per iteration. This is the reason we use aggregated cuts, instead of the multicut formulation.

In spite of such quadratic complexities, the algorithm is still useful. First, the other possibilities based on fully solving an SDDP problem for each new tree are in practice much more expensive when $T$ or $S$ are big. Second, a strategy based on using secondary memory as in Section 6.3 can also be put in place for multistage problems for exactly the same reasons. Namely, the additional coefficients multiplying $h[t]$ do not need to be stored on main memory because their influence on any given LP to be solved by Algorithm 6.4.1 reduces to the free term of any cut. The mathematical description is similar to equations (6.16) and (6.17) so that we leave the details for the reader.

As commented briefly in the introduction, fast upper bound estimates can also be obtained without having to run the backward pass for the new trees, which is the costly part of Algorithm 6.4.1. First, recall that

$$Q_{t+1}^k(x_{ts}, h[t]) \leq Q_{t+1}(x_{ts}, h[t]) \quad \forall x, \quad \forall h[t], \quad \forall k, \quad \forall t = 1, \dots, T-1. \tag{6.23}$$

Therefore, the estimates for the value functions obtained by Algorithm 6.4.1 are valid estimates of the cost-to-go function for all stages, trees and iterates. The key for computing fast upper bounds is to reuse the value functions already obtained as described in Algorithm 6.4.2, which is presented separately from Algorithm 6.4.1 because the computation of upper bounds is not the focus of the chapter. We just want to point out the possibility. Nonetheless, the trees sampled on Step 1.1 of Algorithm 6.4.2 should be the same as in Step 2 of Algorithm 6.4.1 so that the fast upper bound can be compared with the fast lower bound for a given $\hat{h}^l[1]$.

**Algorithm 6.4.2** (Sensitivity of upper bounds of SDDP problems with respect to RHS vectors)**.**

```
**Initialization.** Take integers $L > 0$ and $K > 0$.
**Step 1: Iterate New Trees.** For $l = 1,\dots,L$
    **Step 1.1: Tree Sample.** Sample a new sequence $\hat{h}^l[1]$.
    **Step 1.2: Iterate Forward Paths.** For $k = 1,\dots,K$
        **Step 1.2.1: Forward Sampling.** Sample a scenario $\omega_t^k \in \{1,\dots,S\} \quad \forall t = 2,\dots,T$.
        **Step 1.2.2: Modified Forward.** Compute
            $\hat{x}_1^k \in \arg\min_{x_1} \quad c_1^\top x_1 + Q_{12}^k(x_1,\hat{h}^l[1]) \quad \text{s.t.} \quad x_1 \in D_1$
        and for $t = 2,\dots,T$ and $s = \omega_t^k$ compute
            $\hat{x}_t^k \in \arg\min_{x_{ts}} \quad c_t^\top x_{ts} + Q_{t+1}^k(x_{ts},\hat{h}^l[t]) \quad \text{s.t.} \quad x_{ts} \in D_{ts}(\hat{x}_{t-1}^k,\hat{h}_{ts}^l)$.
        **Step 1.2.3: Forward Path Cost.** Set $\overline{v}_k^l = \sum_{t=1}^T c_t^\top \hat{x}_t^k$.
    **Step 1.3: Upper Bound Estimate.** Set $\overline{v}^l = \frac{1}{K}\sum_{k=1}^K \overline{v}_k^l$.
**Step 2: Output.** Compute statistics of interest over $\{\overline{v}^l : l = 1,\dots,L\}$.
```

## 6.5 Experiments

All experiments are run on an Intel i7 1.90GHz machine, with 15Gb of RAM, Ubuntu 18.04.3 LTS, CPLEX 12.10 and C++. For ease of explanation we consider the configuration of the experiments, given next, fixed throughout the section.

The experiments consider the integrated management of a cascade with a total of $G = 3$ hydroplants for $T = 12$ months (see Figure 6.2). A hydroplant is a reservoir (a triangle) equipped with a turbine downwards (rounded rectangle). The main idea is that such hydroplants are connected via tubulations so that the water released from a hydroplant to produce energy, ends up stored at the reservoirs of other hydroplants. The natural objective for an agent seeking profit is then to produce energy at months with high prices. However the production cannot be concentrated only at the month with the highest price because there is the risk of having to waste water because reservoirs become full or because there are lower and upper limits to the volume of water released per day due to environmental reasons. The cascade is illustrated in Figure 6.2.

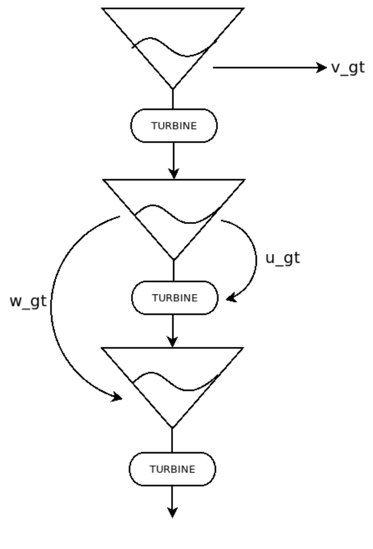

Figure 6.2: The water flows downwards. The volume of each hydroplant $g$ at time $t$ is $v_{gt}$. The water that is used to produce energy is $u_{gt}$ and the water that is released without producing energy is $w_{gt}$.

The water flows from the top of the cascade to the bottom. The hydroplant at the top is identified with $g = 1$, the next with $g = 2$ and so on. There is at most one hydroplant immediately next any other. The $g$-th hydroplant receives water from the rain and possibly also the water released from the only upwards hydroplant. The hydroplants are used to produce energy that is sold at deterministic prices for simplicity and because price does not enter on the RHS. The inflows of water from the rain are stochastic. The objective is to maximize the expected profits of selling energy subject to constraints regarding the amount of water released and withheld. First we show the management problem in the deterministic setting and then explain the differences for the stochastic setting. The variables of the optimization problem are

- the turbined water $u_{gt}$ at hydroplant $g$ and time $t$ (water that passes through the turbine to produce energy),
- the reservoir volume $v_{gt}$ at hydroplant $g$ and time $t$ (raw volume of the reservoir),
- the spilled water $w_{gt}$ at hydroplant $g$ and time $t$ (water released without producing energy in case the reservoir is full).

The constants for each hydroplant are

- the productivity $\rho_g$ of hydroplant $g$ (the fraction of water turbined that is transformed into energy),
- the price $\pi_t$ of energy at time $t$ (price for selling energy),
- the inflow of water from rain $a_{gt}$ at hydroplant $g$ and time $t$,
- the max volume $\overline{v}_g$ of hydroplant $g$,
- the dead volume $\underline{v}_g$ of hydroplant $g$ (volume below which it is not possible to generate energy),
- the maximum spillage capacity $\overline{w}_g$ of hydroplant $g$ (capacity for releasing water per month without producing energy in case the reservoir is full),
- the maximum and minimum outflows for hydroplant $g$, respectively denoted by $\overline{e}_g$ and $\underline{e}_g$ (outflow is the sum of water turbined and spilled).

The units of $u_{gt}, w_{gt}, a_{gt}, \overline{w}_g, \overline{e}_g, \underline{e}_g$ are cubic meters per month, while $v_{gt}$ is cubic meters only. The deterministic analogous of the problem solved is shown below, where decisions are not present if $g-1<1$.

$$\begin{aligned} \max \quad & \sum_{t,g} \rho_g \pi_t u_{gt} \\ \text{subject to} \quad & v_{g,t+1} = v_{gt} - u_{gt} - w_{gt} + u_{g-1,t} + w_{g-1,t} + a_{lt}, \\ & u_{gT} + w_{gT} \le v_{gT} + a_{gT}, \\ & u_{gt} \le v_{gt} - \underline{v}_g, \\ & v_{gt} \in [0, \overline{v}_g], \quad w_{gt} \in [0, \overline{w}_g], \quad u_{gt} \in [0, \overline{u}_g], \quad u_{gt} + w_{gt} \in [\underline{e}_g, \overline{e}_g]. \end{aligned} \tag{6.24}$$

Naturally, problem (6.24) suffers from the end-of-period effect. However, we use it during the experiments for illustrative purposes. The constraints $u_{gt} + w_{gt} \in [\underline{e}_g, \overline{e}_g]$ are written due to environmental reasons. Nonetheless, $u_{gt} + w_{gt} \in [\underline{e}_g, \overline{e}_g]$ and $w_{gt} \in [0, \overline{w}_g]$ are neglected on the experiments below. Therefore, there is no need to report the quantities $\underline{e}_g, \overline{e}_g$ and $\overline{w}_g$. The constraint $u_{gt} \le v_{gt} - \underline{v}_g$ says that water turbined is only of what exceeds the dead volume. The constraint $u_{gT} + w_{gT} \le v_{gT} + a_{gT}$ just says that at the end the reservoir level has to be non-negative and disregards the decisions of the other hydroplants.

We take the efficiencies $\rho_g = 1$ for $g = 1, 2, 3$. The dead volumes are 10% of the maximum volumes and the initial volumes are 15% of the maximum volumes. The maximum turbining capacity per month is 50% of the maximum volume. The maximum volumes are $\overline{v}_1 = 1.6, \overline{v}_2 = 1.0$ and $\overline{v}_3 = 1.6$. The limits for spillage, max outflow and min outflow are not used. The deterministic profiles for prices and inflows are shown below.

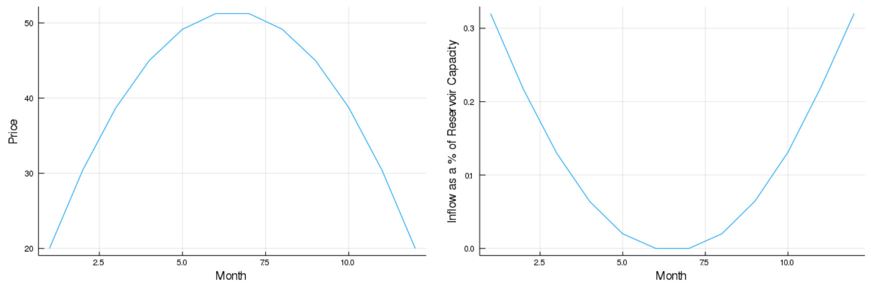

Figure 6.3: Note that the prices and inflows are "out-of-phase". The dry months happen around $t = 6$ as well as higher prices of energy. After adding noises to these profiles, we truncate negative results to zero.

Let us now explain what are the differences in the stochastic setting. The inflows $a_{gt}$ are changed to $a_{gts}$ for all $s = 1, \dots, S$ because they are stochastic and assumed to be stage-wise independent. Still inside the SDDP framework, the inflows can also be modeled as an auto-regressive process (such as ARIMA) where the noise is stage-wise independent. The stage-wise and scenario-wise decomposition of problem (6.24) is given below.

$$Q_{ts}(v_{g,t-1}, a_{gts}, a[t]) = \begin{cases} \max & \displaystyle\sum_{g=1}^{G} \rho_g \pi_t u_{gt} + \frac{1}{S}\sum_{r=1}^{S} Q_{t+1,r}(v_{gt}, a_{g,t+1,r}, a[t+1]) \\ \text{subject to} & v_{g,t} = v_{g,t-1} - u_{gt} - w_{gt} + u_{g-1,t} + w_{g-1,t} + a_{gts}, \\ & u_{gT} + w_{gT} \leq v_{gT} + a_{gT}, \\ & u_{gt} \leq v_{gt} - \underline{v}_g, \\ & v_{gt} \in [0, \overline{v}_g], \quad w_{gt} \in [0, \overline{w}_g], \quad u_{gt} \in [0, \overline{u}_g], \quad u_{gt} + w_{gt} \in [\underline{e}_g, \overline{e}_g]. \end{cases}$$

Indeed, such ARIMA model mentioned above is one of the motivations of this work because the noise follows a continuous distribution in which case the sensitivities are important. For instance, the tendency is that we would need more scenarios to get a good representation of the uncertainty per stage if $G$ is big. Nonetheless, the resulting problem could be computationally too hard and then we would like to be at least aware of sensitivities of the cutting-planes model to the sample of the inflows. Instead of fitting an ARIMA, we make the experiments taking a deterministic profile and then adding different types of noises around the profile. We expect that distributions with heavier tails would need more scenarios to be represented properly.

Now, all the information about the problem is given, except for the type of noise around the profiles in Figure 6.3 and the total scenarios used on the discretization. These last two items are the only that vary on the experiments that follow, which are organized into some parts, each one illustrating an important aspect or application of the right-hand side sensitivity of multistage stochastic optimization problems. In particular, we want to

- illustrate numerically the lower bounding property for the approximations of value functions, and
- measure how the discretized SDDP problem relates to the true problem.

The motivation for the first item is that it is the fundamental building block for the entire chapter. Therefore, it is a nice validation to report. The second item includes two applications as we describe now. The probability distribution of scenarios of the true SDDP problem can be continuous or discrete. For the continuous case, the scenarios can follow a multivariate normal distribution, for instance. In the discrete case, the scenarios can be finitely many, but way too many to put into a computer. For both cases, the discretization of the true uncertainty comes into play strongly.

It turns out that the discretization of an already discrete distribution is known as scenario reduction. We further split the scenario reduction problem into (i) the selection of the discretization itself and (ii) the evaluation of the discretization. Scenario reduction algorithms usually address item (i), which is *not* our objective. We rather make a proposal to efficiently address item (ii), given that the discretization has been made. To achieve such an objective, we can take any discretization, since it does not matter which one. Our point is to show that the computation of sensitivities is possible and effective.

The purpose of the experiments can be further detailed as:

- Validate numerically the lower bounding property of the approximations of the value functions when iterates, trees or stages are changed, as claimed in Eq. (6.23). This assessment is reported on Section 6.5.1.

Column LBSTEP2 reports lower bounds computed as in Step 2 of Algorithm 6.4.1 and compared against an independent lower bound using the traditional SDDP method (column LBSDDP). Moreover, the lower bounds from reusing all previous value function approximations obtained during the experiments of Section 6.5.1 serves to illustrate Eq. (6.23) and are reported on the columns LBSDDP* and UBSDDP*. The objective of this subsection is not to prove that LBSTEP2 bounds are weaker than those obtained independently (LBSDDP, UBSDDP), all of which are weaker than those reusing cuts (LBSDDP*, UBSDDP*). Instead, we use there only $S = 10$ or $S = 20$ scenarios (a small number compared to the other sections) because each independent SDDP run to validate the LBSTEP2 already takes around 10 minutes, which validates that a fast sensitivity technique is helpful. Moreover, as it is clear from the other subsections, $S = 20$ is not enough to make the sample average problem come close to the true problem, which implies that a bigger sensitivity is justified and therefore a weaker fast lower bound is expected.

- Validate numerically that when we discretize an already discrete distribution by selecting a subset of scenarios to use, the sensitivities of the optimal values converge to zero as the discretized distribution converges to the original distribution. This test is shown in Section 6.5.2 on Table 6.3 and Table 6.4 as $S$ approaches $\overline{S}$. It validates that the sensitivities reproduce our statistical intuition. This test is related to scenario reduction techniques, however our objective is only to *evaluate* a possible scenario reduction method. When the original distribution is discrete, convergence of the discretized distribution to the original one can be achieved exactly. This is in contrast to the tests performed on Section 6.5.3, where the original distribution is continuous and can only be approximated.

- Validate numerically that even for a continuous distribution with an unbounded support, the sensitivities obtained by Algorithm 6.4.1 get smaller and smaller as $S$ grows bigger. These tests are shown in Section 6.5.3. Comparing Tables 6.5 and 6.6 to the results of Section 6.5.2 suggests that sensitivities for continuous distributions still get small, but "convergence" is harder to achieve. We note that the histograms shown in Figure 6.4 are qualitatively similar to the ones in Figure 6.1.

### 6.5.1 Validity of lower bounds across trees

In this subsection we illustrate that the lower bounds we obtain are indeed valid. This is done by comparing the lower bounds of Step 2 of Algorithm 6.4.1 with the lower and upper bounds obtained solving the new trees *separately* with the usual SDDP. For this purpose, the distributions of the noises applied to Figure 6.3 do not matter, as is clear from algorithms 6.3.1 and 6.4.1 because the lower bound is valid for all trees. The distribution just influences the rate of convergence of the sample average to the continuous problem. We also report on the SDDP* columns the solution of the new trees reusing previous free-floating cuts. The results are shown in Table 6.1 and Table 6.2.

Notice that the value functions on Algorithm 6.4.1 are initialized with a negative constant. This is the initialization of value functions for the base tree. The index $l \geq 1$ below, means that the value functions for the columns LBSDDP* and UBSDDP* are initialized with the value functions from experiment $l - 1$, instead of the negative constant mentioned. On the other hand, the values of columns LBSDDP and UBSDDP are based on solving the problem from scratch, that is, initializing value functions with the negative constant.

For Table 6.1, we use $S = 10$ scenarios per stage. The scenarios are generated adding a uniform noise around the inflow profile with range $[0, \eta]$, where $\eta$ is the average inflow for the respective hydroplant. For each tree we perform 70 forward-backward iterations (Step 1 of Algorithm 6.4.1). First we solve the base tree. We find a lower bound of -724.37 and an upper bound of -706.58. We can see that the SDDP* columns show lower gaps at the expense of bigger running times because the subproblems are bigger due to more cuts coming from lower $l$. The cuts from the solution of the base tree are passed to the $l = 1$ tree and the resulting cuts to $l = 2$ and so on. At $l = 5$ all the previous cuts are being reused.

Generally speaking, we observe for both Table 6.1 and Table 6.2 that LBSTEP2 is the weakest bound, followed respectively by LBSDDP and LBSDDP*. However, in spite of LBSTEP2 being weaker in absolute terms, the difference to the other bounds is less than 2% for all tests. On the other hand, the gap obtained by the independent SDDP run (UBSDDP - LBSDDP) is greater than 2%. This is a satisfatory result, since the fast lower bounds obtained have much lower computational costs.

| New Tree | LBSTEP2 | LBSDDP | UBSDDP | LBSDDP* | UBSDDP* |
|---|---|---|---|---|---|
| $l = 1$ | -736.49 | -726.82 | -710.18 | -726.63 | -726.01 |
| $l = 2$ | -735.22 | -727.14 | -707.41 | -727.00 | -725.97 |
| $l = 3$ | -734.44 | -723.79 | -702.66 | -723.67 | -720.59 |
| $l = 4$ | -729.25 | -726.33 | -708.59 | -713.95 | -713.40 |
| $l = 5$ | -728.22 | -722.77 | -703.15 | -717.29 | -712.17 |

Table 6.1: Here we use $S = 10$. The base tree LB is -724.37 and an upper bound of -706.58.

| New Tree | LBSTEP2 | LBSDDP | UBSDDP | LBSDDP* | UBSDDP* |
|---|---|---|---|---|---|
| $l = 1$ | -743.16 | -733.40 | -716.59 | -733.28 | -731.43 |
| $l = 2$ | -734.63 | -725.30 | -707.34 | -725.20 | -717.59 |
| $l = 3$ | -735.90 | -727.28 | -712.83 | -726.70 | -722.25 |
| $l = 4$ | -736.23 | -726.10 | -708.37 | -725.71 | -725.05 |
| $l = 5$ | -734.38 | -724.60 | -703.41 | -724.40 | -718.35 |

Table 6.2: Here we use $S = 20$ and the same setting of Table 6.1. The base tree LB is -728.93 and UB is -712.04.

### 6.5.2 Evaluation of scenario reduction techniques

In the context of scenario reduction [Oli+10; DGKR03] we are given a big number of scenarios $\overline{S}$ and want to find a subset of size $S$ of these scenarios that represents well the problem with $\overline{S}$ scenarios. The motivating assumption is that solving the problem with $\overline{S}$ scenarios is too hard or impossible. The techniques proposed [Oli+10; DGKR03] are based on heuristics because selecting the subset of size $S$ cannot take longer than solving the problem with size $\overline{S}$. Our technique of cut-sharing across trees can be used to *evaluate* a subset of size $S$ once it is selected and solved. We are not proposing a scenario reduction technique. Instead, we propose a fast way to *evaluate* a scenario reduction algorithm.

Precisely, we can (i) use a technique of scenario reduction to select a subset of size $S$ out of the $\overline{S}$ scenarios and (ii) sample, without replacement, other sets of $S$ scenarios from the $\overline{S}$ and evaluate a fast lower bound. It has to be without replacement because if we are allowed to sample $S = \overline{S}$ scenarios, we need to obtain exactly the original set of scenarios. If the set of $S$ scenarios is well selected by the scenario reduction method the cutting-planes model will not show significant sensitivity to other samples with the same size. This subsection tests this idea, but making the scenario reduction by selecting a subset of size $S$ randomly. The results are shown in Table 6.3 and Table 6.4.

For Table 6.3, the scenarios are generated adding a uniform noise around the inflow profile with range $[0, \eta]$, where $\eta$ is the average inflow for the respective hydroplant. We take $\overline{S} = 70$ and take randomly a base tree with $S$ scenarios per stage. The first and second columns report the lower and upper bounds for the base tree after making 70 forward-backward iterations. We realize an out-of-sample analysis of the resulting cutting-planes model taking $L = 400$ in step 2 of Algorithm 6.4.1. The new groups of $S$ scenarios inside step 2 are sampled without replacement. We observe that the closer $S$ is to $\overline{S}$, the more representative the results of the base tree are. Note that expectation of $\hat{v}_1^l$ is within a 3% range from the lower bound for the base tree, while the gaps for the base tree (UB - LB) are also around 3%.

For Table 6.4, we use the same settings of Table 6.3, but with a uniform noise with 40% of the original range. Note that for $S = \overline{S}$ the out-of-sample test is equivalent to randomly reordering the scenarios, which makes more numerical errors from the multiplier $\hat{\lambda}$ in (6.3) become apparent. This is the reason that for $S = 70$ we have the lower bound different from the expectation of $\hat{v}_1^l$. Notice that for $S = 10$, the standard deviation of $\hat{v}_1^l$ is already around 2%, showing that adding many more scenarios does not change the results much.

Selecting a subset of size $S$ randomly helps to verify the effectiveness of Algorithm 6.4.1 independently of any scenario reduction algorithm. We can observe on Table 6.3 and Table 6.4 exactly the behavior we would expect from pure statistical reasoning. For instance, out of the $\overline{S} = 70$ scenarios per stage, we expect that half of those already contain most of the overall statistical information on the scenarios. Moreover, the more $S$ is closer to $\overline{S}$, we would expect to see less and less sensitivity of the optimal value to the scenarios left out by the scenario reduction method, which is observed on the standard deviation of $\hat{v}_1^l$ and on its max deviation from the mean. Therefore, the results of Algorithm 6.4.1 are consistent with overall statistical intuition.

| $S$ | Lower Bound | Upper Bound | Expectation $\hat{v}_1^l$ | Std. Dev. $\hat{v}_1^l$ | $\max_l \lvert\hat{v}_1^l - \mathbb{E}\hat{v}_1^l\rvert$ |
|---|---|---|---|---|---|
| 1 | -743.37 | -729.29 | -736.09 | 20.02 | 56.89 |
| 5 | -721.61 | -705.45 | -742.30 | 13.01 | 36.19 |
| 10 | -742.42 | -725.64 | -740.23 | 06.94 | 21.40 |
| 30 | -718.12 | -698.56 | -744.64 | 03.85 | 11.65 |
| 50 | -725.42 | -703.48 | -742.37 | 01.62 | 06.19 |
| 70 | -727.42 | -711.02 | -739.51 | 00.19 | 00.69 |

Table 6.3: The average of $\hat{v}_1^l$ across $S$ is around 740.85, with standard deviation less than 1%, in spite of possibly big variations of $\hat{v}_1^l$ for each $S$. This is in line with Figure 6.1, which reports stable averages across $S$.

| $S$ | Lower Bound | Upper Bound | Expectation $\hat{v}_1^l$ | Std. Dev. $\hat{v}_1^l$ | $\max_l \lvert\hat{v}_1^l - \mathbb{E}\hat{v}_1^l\rvert$ |
|---|---|---|---|---|---|
| 1 | -522.11 | -511.82 | -483.03 | 22.38 | 61.05 |
| 5 | -487.49 | -475.99 | -481.63 | 09.13 | 25.07 |
| 10 | -476.29 | -467.22 | -481.72 | 06.89 | 18.02 |
| 30 | -480.82 | -468.74 | -481.95 | 02.88 | 09.13 |
| 50 | -483.77 | -474.89 | -481.88 | 01.71 | 05.40 |
| 70 | -481.42 | -470.39 | -482.11 | 00.01 | 00.05 |

Table 6.4: Note that the average of $\hat{v}_1^l$ is stable across $S$ and reports the true lower bound obtained with $S = \overline{S}$. This is qualitatively similar to the results reports on Figure 6.1.

### 6.5.3 Continuous distributions with unbounded supports

We use noises with compact support on Subsections 6.5.1 and 6.5.2. In this subsection we use our technique to evaluate the effect of sampling from continuous distributions with unbounded supports or heavy tails on the sensitivity of cutting-plane models.

For our particular application, the occurrence of extreme values is equivalent to either a lot of rain flooding the reservoirs or no rain at all. For some other applications, the extreme event is at ultimate analysis also characterized by high costs. In other words, one would need to know the cost associated with the realizations to determine if it is extreme or not because the joint effects of the random variables could change the judgment. From an algorithmic perspective, our technique might be helpful because one would (i) solve a base tree, (ii) get estimates of the cost for many trees and (iii) select trees with the worst costs to be called extreme realizations, possibly improving the cost estimates for the trees selected and iterating back to (ii).

Next we show sensitivities of cutting-plane models to the tree when the noise is normal. We take $L = 2000$ trees for the out-of-sample analysis and make 18 forward-backward iterations. This is much less than the previous experiments due to memory usage issues, because the amount of scenarios in this section reaches almost the triple of the previous experiments ($S = 200$). However, such an early stop does not affect the convergence, as can be seen on the magnitude of the sensitivities, because 18 forward-backward passes for $S = 200$ scenarios obtains enough lower bound information.

Tables 6.5 and 6.6 report the statistical significance of the lower bounds obtained. The difference between Tables 6.5 and 6.6 is that the normal distribution of Table 6.6 has 50% of the standard deviation of the one for Table 6.5. Again, the normal noise is discretized with $S$ samples to build the sample average problem. The first two columns of Tables 6.5 and 6.6 refer to the base tree solved. We observe that "convergence" for Table 6.6 happens easier than for Table 6.5 because the standard deviation of $\hat{v}_1^l$ is smaller relative to the lower bounds. For Table 6.5 the standard deviation of $\hat{v}_1^l$ is around 1.3% and the maximum deviation is around 4.7%, while for Table 6.6 these numbers are 1.0% and 3.7%, respectively.

Again we note that the averages of $\hat{v}_1^l$ are stable across $S$, as also observed in Figure 6.1. As explained before, for big $S \geq 100$ we have the lower bound slightly different from the expectation of $\hat{v}_1^l$ because of numerical errors from the free-floating part of the cut. The quantity $\max_l |\hat{v}_1^l - \mathbb{E}\hat{v}_1^l|$ goes to zero because the problem has bounded feasible sets and is always feasible. Otherwise, sampling enough would make it diverge. The evolution of the histograms for $\hat{v}_1^l$ as a function of $S$ for both Tables 6.5 and 6.6 are shown in Figure 6.4.

| $S$ | Lower Bound | Upper Bound | Expectation $\hat{v}_1^l$ | Std. Dev. $\hat{v}_1^l$ | $\max_l \lvert \hat{v}_1^l - \mathbb{E}\hat{v}_1^l \rvert$ |
|---|---|---|---|---|---|
| 1 | -306.42 | -259.73 | -424.46 | 91.87 | 305.69 |
| 5 | -477.26 | -423.01 | -484.57 | 39.91 | 144.39 |
| 10 | -426.86 | -345.59 | -467.20 | 28.50 | 111.90 |
| 30 | -447.75 | -430.41 | -439.40 | 17.26 | 60.29 |
| 50 | -461.76 | -460.61 | -486.18 | 12.52 | 42.03 |
| 70 | -461.55 | -433.26 | -486.50 | 10.79 | 38.04 |
| 100 | -467.25 | -443.87 | -484.30 | 9.08 | 33.20 |
| 150 | -453.88 | -418.31 | -483.25 | 7.72 | 25.77 |
| 200 | -468.83 | -435.76 | -487.70 | 6.35 | 22.23 |

Table 6.5: This tables shows the sensitivity of the cutting-planes for a normal noise with zero mean. Recall that resulting negative inflows are truncated to zero.

| $S$ | Lower Bound | Upper Bound | Expectation $\hat{v}_1^l$ | Std. Dev. $\hat{v}_1^l$ | $\max_l \lvert \hat{v}_1^l - \mathbb{E}\hat{v}_1^l \rvert$ |
|---|---|---|---|---|---|
| 1 | -368.58 | -336.22 | -288.87 | 52.06 | 214.27 |
| 5 | -399.61 | -353.66 | -359.04 | 22.16 | 82.26 |
| 10 | -355.54 | -311.00 | -358.68 | 16.40 | 69.45 |
| 30 | -345.73 | -304.63 | -349.68 | 9.49 | 35.13 |
| 50 | -338.62 | -324.00 | -345.04 | 7.21 | 25.36 |
| 70 | -336.81 | -286.73 | -343.06 | 6.26 | 22.30 |
| 100 | -345.41 | -352.81 | -349.11 | 5.09 | 16.79 |
| 150 | -349.43 | -317.25 | -349.11 | 4.14 | 16.87 |
| 200 | -344.54 | -324.00 | -348.55 | 3.72 | 13.72 |

Table 6.6: The normal noise for this experiments has 50% the standard deviation of the one for Table 6.5. Note that $\hat{v}_1^l$ is stable across $S$.

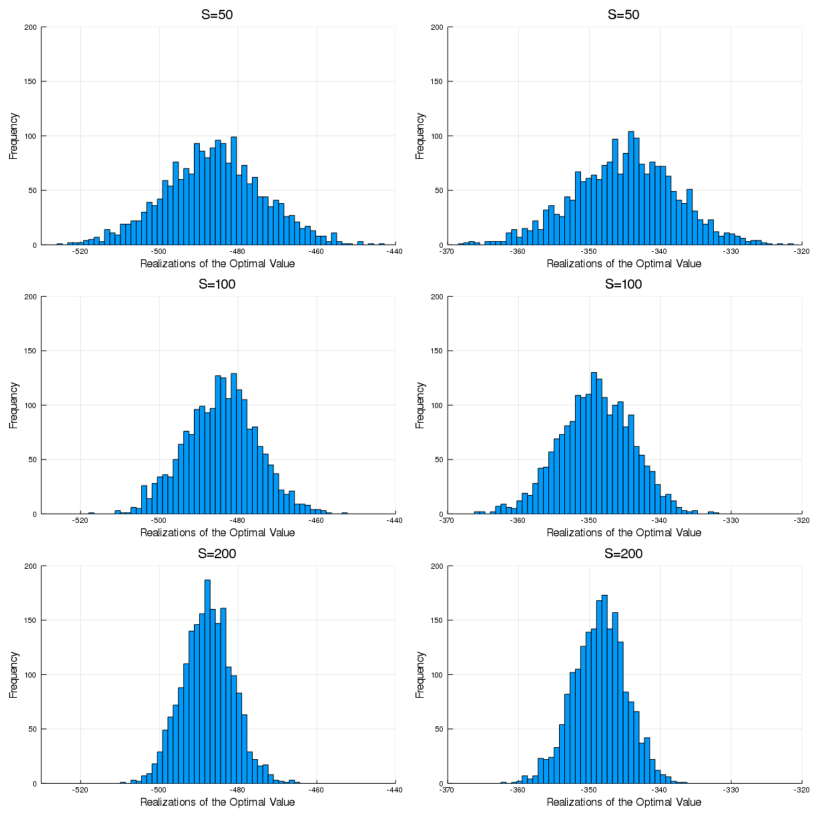

Figure 6.4: Histograms for the realizations of $\hat{v}_1^l$ for Table 6.5 (left) and Table 6.6 (right). The results are similar to Figure 6.1. Note that the histogram for $S = 200$ on the right has a range of about 13 units around the mean, as reported on Table 6.6 as the maximum deviation, which represents about 4% of the optimal value.

## 6.6 Extensions via duality

In this section we replicate the dual problems found in [GSC20] and explain how our strategy for cut-sharing across trees applies. The dynamic programming equations (DPE) for the dual SDDP method [Lec+20; GSC20] can be derived by (i) writing the dual of the deterministic equivalent primal problem and (ii) identifying stage-wise decomposition for the dual using concave value functions. Once this is performed, in spite of problems relating to feasibility that should be dealt with carefully [GSC20], the solution procedures for the dual tend to look familiar because the structure of the dual is somewhat similar to the usual primal.

Note that the derivation of the dynamic programming equations for the dual SDDP and the proper explanation of the underlying issues relating to feasibility is a delicate topic, which we do not intend to deal with here. For the application of our strategy, it is enough to examine the structure of the DPE and recognize it is similar enough to the primal equations. Once one realizes the structure is the same, it is enough to transform the maximization problems to minimization problems and apply the cut-sharing as described before.

The data process is denoted by $\xi_t = (c_t, W_t, B_t, h_t)$ and assumed to have finitely many realizations for all stages denoted by $\xi_{ts}$ for $s = 1, \dots, S$. We also use $c_t = c_t(\xi_t)$, $W_t = W_t(\xi_t)$, $B_t = B_t(\xi_t)$ and $h_t = h_t(\xi_t)$. The history of the process is denoted by $\xi_{[t]} = (\xi_1, \dots, \xi_t)$. Note that $\xi_1$ is deterministic. The stage-wise independence translates into assuming that $\xi_t$ is independent of $\xi_{[t-1]}$. The primal and dual multistage stochastic deterministic equivalent problems are respectively given by

$$\min_{x_t(\cdot)} \quad \mathbb{E}\big[\sum_{t=1}^{T} c_t x_t\big] \quad \text{s.t.} \quad x_t \geq 0, \quad W_1 x_1 = h_1, \quad W_t x_t = h_t - B_t x_{t-1} \quad \forall t = 2, \dots, T \tag{6.25}$$

and

$$\max_{\pi_t(\cdot)} \quad \mathbb{E}\big[\sum_{t=1}^{T} h_t \pi_t\big] \quad \text{s.t.} \quad W_T^{\mathsf{T}} \pi_T \leq c_T, \quad W_{t-1}^{\mathsf{T}} \pi_{t-1} + \mathbb{E}_{|\xi[t-1]}\big[B_t^{\mathsf{T}} \pi_t\big] \leq c_{t-1} \quad \forall t = 2, \dots, T \tag{6.26}$$

where the optimization is performed over policies $x_t = x_t(\xi_{[t]})$ and $\pi_t = \pi_t(\xi_{[t]})$ so that the nonanticipativity constraints are already implicitly being considered and the probability space where the expectation is considered is properly constructed. Standard duality theory applies to the pair (6.25)-(6.26).

As is widely known, the upper bound for the primal SDDP is calculated via Monte Carlo simulation. It is considered a weakness of the method because this procedure is slow and random. One of the motivations of [Lec+20; GSC20] was to generate deterministic upper bounds for the optimal value of a SDDP problem. The first stage problem of the dual SDDP is given

$$\max_{\pi_1} \quad h_1^{\mathsf{T}} \pi_1 + V_2(\pi_1, \xi_1). \tag{6.27}$$

For $t = 2, \dots, T-1$ the definition of $V_t(\pi_{t-1}, \xi_{t-1})$ is given by

$$\max_{\pi_{t1}, \dots, \pi_{tS}} \quad S^{-1} \sum_{s=1}^{S} h_{ts}^{\mathsf{T}} \pi_{ts} + S^{-1} \sum_{s=1}^{S} V_{t+1}(\pi_{ts}, \xi_{ts}) \quad \text{s.t.} \quad W_{t-1}^{\mathsf{T}} \pi_{t-1} + S^{-1} \sum_{s=1}^{S} B_{ts}^{\mathsf{T}} \pi_{ts} \leq c_{t-1}. \tag{6.28}$$

Finally, the definition of $V_T(\pi_{T-1}, \xi_{T-1})$ is

$$\max_{\pi_{T1}, \dots, \pi_{TS}} \quad S^{-1} \sum_{s=1}^{S} h_{Ts}^{\mathsf{T}} \pi_{Ts} \quad \text{s.t.} \quad W_{T-1}^{\mathsf{T}} \pi_{T-1} + S^{-1} \sum_{s=1}^{S} B_{Ts}^{\mathsf{T}} \pi_{Ts} \leq c_{T-1}, \quad W_{Ts}^{\mathsf{T}} \pi_{Ts} \leq c_{Ts} \quad \forall s.$$

First of all, note that (6.27) is unconstrained. Therefore, we run into bad iterates if the first forward passes are performed without care. This is so because we use upper bounding cutting-plane models for these value functions. Secondly, the subproblems might be bigger because they involve all $\pi_{t1}, \dots, \pi_{tS}$ for all scenarios.

For our purposes, we have to note only that inside (6.28) the cost vectors of the primal appear in the RHS as well as $\pi_{t-1}$. Therefore, we can generate free-floating cuts approximating $V_t$ in terms not only of $\pi_{t-1}$ but also in terms of the realizations of the cost. This information is passed backwards replicating the RHS parameters in (6.28) analogously to what we have done before. The two-stage case is written below to illustrate. The first stage problem is still (6.27) and the second stage value function $V_2(\pi_1, c_{21}, \dots, c_{2S})$ is

$$\max_{\pi_{21}, \dots, \pi_{2S}} \quad S^{-1} \sum_{s=1}^{S} h_{2s}^{\mathsf{T}} \pi_{2s} \quad \text{s.t.} \quad W_1^{\mathsf{T}} \pi_1 + S^{-1} \sum_{s=1}^{S} B_{2s}^{\mathsf{T}} \pi_{2s} \leq c_1, \quad W_{2s}^{\mathsf{T}} \pi_{2s} \leq c_{2s} \quad \forall s.$$

Assume given an iterate $(\hat{\pi}_1, \hat{c}_{21}, \ldots, \hat{c}_{2S})$ where $\hat{c}_{21}, \ldots, \hat{c}_{2S}$ are obtained via sampling. Using formulas similar to (6.3) we can generate supergradient vectors $(\hat{\alpha}_1, \hat{\beta}_{21}, \ldots, \hat{\beta}_{2S})$ such that

$$V_2(\pi_1, c_{21}, \ldots, c_{2S}) \leq V_2(\hat{\pi}_1, \hat{c}_{21}, \ldots, \hat{c}_{2S}) + \hat{\alpha}^\intercal(\pi_1 - \hat{\pi}_1) + \sum_{s=1}^{S} \hat{\beta}_{2s}^\intercal(c_{2s} - \hat{c}_{2s}) \quad \forall \pi_1, \quad \forall c_{2s}. \tag{6.29}$$

Inside a cutting-planes method, formula (6.29) is used to improve a polyhedral approximation of the first stage problem (6.27) so that upper bounds for new cost vectors can be computed fast once one base tree is solved.

## 6.7 Conclusions

In this chapter we presented algorithms to compute valid lower bounds for right-hand side perturbations of multistage stochastic optimization problems that can be obtained at minor computational costs. The approach is based on the extensive use of certain extended cutting-planes, which we call free-floating cuts. We presented numerical experiments justifying the intuition that such cutting-planes can be used to measure the convergence of discretized problems to the true problem. It would be a nice future development to perform the statistical analysis of the algorithms proposed. Regarding extensions of the approach, it can be applied as soon as all the parameters of a value function are in the RHS and the optimization problem defining the value function is convex via extensions of (6.2). Therefore, it can also be applied if prox-centers are added to the objective at each stage.

In spite of not being explored in enough detail in this thesis, computationally cheap upper bound estimates for different trees (other than the initial one) can also be obtained with our developments as is explained in Algorithm 6.4.2. Comparatively to resolving the new trees from scratch with the traditional SDDP, the cost of the so-called backward pass is removed, which is the most expensive part of the method. This is possible because the approximations that we provide for the cost-to-go functions are valid for all stages and all trees with the same amount of scenarios. Therefore, a sequence of the so-called forward passes can be performed to obtain the desired statistical upper bounds. These ideas are shown in Algorithm 6.4.2.

# Concluding Remarks

The techniques developed for this Ph.D. work consider value functions and solution mappings of fully parameterized convex problems and extensions of Benders cuts for linear problems with right-hand side parameters. Naturally, one could ask if our techniques could be extended for non-convex fully parameterized problems and how restrictive is the right-hand side parameterization of linear optimization problems.

As it is widely known, the Sequential Quadratic Programming (SQP) methods and alike [XYZ15] replace a non-convex problem with a quadratic program that approximates the former locally. Sometimes the quadratic term is used to induce superlinear convergence and sometimes it is used to guarantee existence of solutions of the approximations. For the latter, we can use box constraints as well. Usually, such technique is employed in the non-parametric setting, where the linearization is performed only in the decision variables. Nonetheless, the linearization of both the objective function and constraints can be performed in both decision variables and parameters. This procedure possibly approximates the fully parameterized problem by a quadratic problem with only right-hand side parameters around centering points for both decisions and parameters.

Naturally, one asks if such approximation can be done in a controlled manner so that algorithms can be proposed for more general classes of hierarchical optimization problems. The answer to the question is positive if we know in advance a Lipschitz constant to the problem data, in the sense that the approximation can be performed with good properties (from above and below). However, to propose algorithms based on this idea we need to manage the parameters associated with the approximation carefully, which must be the subject of future research.

The proposed approximation of the fully non-convex parameterized problem is as follows. Take $f(x,p)$ a smooth function with Lipschitz constant $L$. It is widely known that

$$|f(x+td_x, p+sd_p) - f(x,p) - t\nabla_x f(x,p)^\intercal d_x - s\nabla_p f(x,p)^\intercal d_p| \leq \frac{L}{2}t^2\|d_x\|^2 + \frac{L}{2}s^2\|d_p\|^2 \quad \forall x,p,d_x,d_p,t,s.$$

Taking $t=1$ and $s=1$ and imposing $\|d_x\| \leq \delta$ and $\|d_p\| \leq \delta$ we obtain

$$|f(x+d_x, p+d_p) - f(x,p) - \nabla_x f(x,p)^\intercal d_x - \nabla_p f(x,p)^\intercal d_p| \leq L\delta^2 \quad \forall x,p,d_x,d_p.$$

The conditions $\|d_x\| \leq \delta$ and $\|d_p\| \leq \delta$ define the balls over which the approximation is valid. With such approximations we can build local polyhedral approximations from above and below to the value function of very general optimization problems. For simplicity, in the unconstrained case we replace the problem $\min_x f(x,p)$ around $x$ and $p$ by two linear problems. For all $\|d_p\| \leq \delta$, we approximate from below by (note that we use box constraints instead of the usual quadratic term)

$$\min_{d_x} \quad f(x,p) + \nabla_x f(x,p)^\intercal d_x + \nabla_p f(x,p)^\intercal d_p - L\delta^2 \quad s.t. \quad \|d_x\|_\infty \leq \delta,$$

and from above by

$$\min_{d_x} \quad f(x,p) + \nabla_x f(x,p)^\intercal d_x + \nabla_p f(x,p)^\intercal d_p + L\delta^2 \quad s.t. \quad \|d_x\|_\infty \leq \delta.$$

The difference between the two problems above is the sign of the term $L\delta^2$. Note that the minimization for the approximating problems is on $d_x$. Moreover, the constant terms can be removed from the objective, which are $f(x,p) + \nabla_p f(x,p)^\intercal d_p \mp L\delta^2$. Therefore, the approximations from below and above are given, respectively, by

$$f(x,p) + \nabla_p f(x,p)^\intercal d_p \mp L\delta^2 + v(x,p,d_p),$$

where

$$v(x,p,d_p) \quad := \quad \min_{d_x} \quad \nabla_x f(x,p)^\intercal d_x \quad s.t. \quad \|d_x\|_\infty \leq \delta.$$

For the constrained case, a similar approach can be carried out, in which case $v$ actually depends on $d_p$ and the parameters $d_p$ appear only linearly in the right-hand side of the constraints. Therefore, the Benders cuts, or free floating cuts, for $v$ can be computed as functions of $d_p$.

Therefore, local polyhedral approximations "sandwiching" the value function of the fully non-convex problem around $(x,p)$ can be obtained. In particular, the rigth-hand side parameterization of linear problems is locally general. We believe these approximations would be useful for extending the techniques presented in this work for fully non-convex problems both in the decisions and in the parameters.

This page is intentionally left blank.

# Bibliography

[ADCM91] I. Adler and R. D. C. Monteiro. “Limiting behavior of the affine scaling continuous trajectories for linear programming problems”. In: *Mathematical Programming* 50.1 (1991), 29–51.

[Ahm06] S. Ahmed. “Convexity and decomposition of mean-risk stochastic programs”. In: *Mathematical Programming* 106.3 (2006), pp. 433–446.

[And+08] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. “On Augmented Lagrangian Methods with General Lower-Level Constraints”. In: *SIAM Journal on Optimization* 18.4 (2008), pp. 1286–1309.

[ASP14] A. Alvarado, G. Scutari, and J.-S. Pang. “A New Decomposition Method for Multiuser DC-Programming and Its Applications”. In: *IEEE Transactions on Signal Processing* 62.11 (2014), pp. 2984–2998.

[Att77] H. Attouch. “Convergence de fonctions convexes, de sous-differentiels et semi-groupes”. In: *Comptes Rendus de l’Academie des Sciences de Paris* 284.1 (1977), pp. 539–542.

[Ban+83] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer. *Non-Linear Parametric Optimization*. Springer, 1983.

[BCC12] J. F. Bonnans, Z. Cen, and T. Christel. “Sensitivity Analysis of Energy Contracts by Stochastic Programming Techniques”. In: *Springer Proceedings in Mathematics*. Springer Berlin Heidelberg, 2012, pp. 447–471.

[Bel57] R. Bellman. *Dynamic Programming*. Dover Publications, 1957.

[Ber95] D. Bertsekas. *Nonlinear programming*. Belmont, Massachusetts: Athena Scientific, 1995.

[Bez+17] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. “Julia: A fresh approach to numerical computing”. In: *SIAM Review* 59.1 (2017), pp. 65–98.

[BH13] J. V. Burke and T. Hoheisel. “Epi-convergent Smoothing with Applications to Convex Composite Functions”. In: *SIAM Journal on Optimization* 23.3 (2013), pp. 1457–1479.

[BH16] J. V. Burke and T. Hoheisel. “Epi-convergence Properties of Smoothing by Infimal Convolution”. In: *Set-Valued and Variational Analysis* 25.1 (2016), pp. 1–23.

[BHK13] J. V. Burke, T. Hoheisel, and C. Kanzow. “Gradient Consistency for Integral-convolution Smoothing Functions”. In: *Set-Valued and Variational Analysis* 21.2 (2013), pp. 359–376.

[Bir82] J. R. Birge. “The value of the stochastic solution in stochastic linear programs with fixed recourse”. In: *Mathematical Programming* 24.1 (1982), pp. 314–325.

[BM06] G. Bayraksan and D. P. Morton. “Assessing solution quality in stochastic programs”. In: *Mathematical Programming* 108.2-3 (2006), pp. 495–514.

[Bon+06] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization. Theoretical and Practical Aspects*. Universitext. 2nd. edition, xiv+423 pp. Berlin: Springer-Verlag, 2006.

[Bor+21] P. Borges, C. Sagastizábal, L. Liberti, C. D’Ambrósio, and M. Solodov. “Profit Sharing Mechanisms in Multi-Owned Cascaded Hydro Systems”. In: *Submitted* (2021).

[BS00] J. F. Bonnans and A. Shapiro. *Perturbation Analysis Of Optimization Problems*. Springer, 2000.

[BSS20] P. Borges, C. Sagastizábal, and M. Solodov. “A regularized smoothing method for fully parameterized convex problems with applications to convex and nonconvex two-stage stochastic programming”. In: *Mathematical Programming* (2020).

[BSS21] P. Borges, C. Sagastizábal, and M. Solodov. “Decomposition Algorithms for Some Deterministic and Two-Stage Stochastic Single-Leader Multi-Follower Games”. In: *Computational Optimization and Applications* 1.78 (2021), pp. 675–704.

[BT12] A. Beck and M. Teboulle. "Smoothing and First Order Methods: A Unified Framework". In: *SIAM Journal on Optimization* 22.2 (2012), pp. 557–580.

[CA13] F. Cicconet and K. C. Almeida. "Decentralized dispatch with price consistency in predominantly hydro systems". In: *IEEE Grenoble Conference* (2013), pp. 1–6.

[Cat+09] J. P. S. Catalao, S. J. P. S. Mariano, V. M. F. Mendes, and L. A. F. M. Ferreira. "Scheduling of Head-Sensitive Cascaded Hydro Systems: A Nonlinear Approach". In: *IEEE Transactions on Power Systems* 24.1 (2009), pp. 337–346.

[CF10] W. Chung and J. D. Fuller. "Subproblem Approximation in Dantzig-Wolfe Decomposition of Variational Inequality Models with an Application to a Multicommodity Economic Equilibrium Model". In: *Operations Research* 58.5 (2010), pp. 1318–1327.

[Che12] X. Chen. "Smoothing methods for nonsmooth, nonconvex minimization". In: *Mathematical Programming* 134.1 (2012), pp. 71–99.

[CP99] Z. Chen and W. Powell. "Convergent cutting plane and partial sampling algorithm for multistage stochastic linear programs with recourse". In: *Journal of Optimization Theory and Applications* 102.1 (1999), pp. 497–524.

[CPM10] J. Catalão, H. Pousinho, and V. Mendes. "Scheduling of head-dependent cascaded hydro systems: Mixed-integer quadratic programming approach". In: *Energy Conversion and Management* 51.3 (2010), pp. 524–530.

[CQS98] X. Chen, L. Qi, and D. Sun. "Global and Superlinear Convergence of the Smoothing Newton Method and Its Application to General Box Constrained Variational Inequalities". In: *Mathematics of Computation* 67.222 (1998), pp. 519–540.

[DB06] C. Donohue and J. Birge. "The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse". In: *Algorithmic Operations Research* 1.1 (2006), pp. 20–30.

[Deá06] I. Deák. "Two-stage stochastic problems with correlated normal variables: computational experiences". In: *Annals of Operations Research* 142.1 (2006), pp. 79–97.

[Dem02] S. Dempe. *Foundations of Bilevel Programming*. Springer, US, 2002.

[DGKR03] J. Dupacova, N. Growe-Kuska, and W. Romisch. "Scenario reduction in stochastic programming: An approach using probability metrics". In: *Mathematical Programming* 95 (2003), pp. 493–511.

[DGL12] N. Dinh, M. Goberna, and M. López. "On the stability of the optimal value and the optimal set in optimization problems". In: *Journal of Convex Analysis* 19 (2012), pp. 927–953.

[DJW17] J. Deride, A. Jofré, and R. J.-B. Wets. "Solving Deterministic and Stochastic Equilibrium Problems via Augmented Walrasian". In: *Computational Economics* 53.1 (2017), pp. 315–342.

[DKK20] S. Dempe, O. Khamisov, and Y. Kochetov. "A special three-level optimization problem". In: *Journal of Global Optimization* 76.1 (2020), pp. 519–531.

[DM02] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical Programming* 91.2 (2002), pp. 201–213.

[DM14] S. Dempe and P. Mehlitz. "Lipschitz continuity of the optimal value function in parametric optimization". In: *Journal of Global Optimization* 61.2 (2014), pp. 363–377.

[DRS09] D. Dentcheva, A. Ruszczyński, and A. Shapiro. *Lectures on Stochastic Programming*. SIAM, Philadelphia, 2009.

[DS99] L. M. G. Drummond and B. F. Svaiter. "On Well Definedness of the Central Path". In: *Journal of Optimization Theory and Applications* 102.2 (1999), pp. 223–237.

[DSS09] A. Daniilidis, C. Sagastizábal, and M. Solodov. "Identifying Structure of Nonsmooth Convex Functions by the Bundle Technique". In: *SIAM Journal on Optimization* 20 (2009), pp. 820–840.

[Esc+07] L. F. Escudero, A. Garín, M. Merino, and G. Pérez. "The value of the stochastic solution in multistage problems". In: *TOP* 15.1 (2007), pp. 48–64.

[FC05] J. D. Fuller and W. Chung. "Dantzig-Wolfe Decomposition of Variational Inequalities". In: *Comput. Econ.* 25 (4 2005), pp. 303–326.

[FI90] A. V. Fiacco and Y. Ishizuka. "Sensitivity and stability analysis for nonlinear programming". In: *Annals of Operations Research* 27.1 (1990), pp. 215–235.

[Fia83] A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. 1983.

[FJQ99] F. Facchinei, H. Jiang, and L. Qi. "A smoothing method for mathematical programs with equilibrium constraints". In: *Mathematical Programming* 85.1 (1999), pp. 107–134.

[FM68] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Wiley, 1968.

[FM99] M. C. Ferris and T. S. Munson. "Interfaces to PATH 3.0: Design, Implementation and Usage". In: *Computational Optimization*. Springer US, 1999, pp. 207–227.

[FP03] F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems, volumes 1 and 2*. Springer Series in Operations Research and Financial Engineering, 2003.

[FPS11] F. Facchinei, V. Piccialli, and M. Sciandrone. "Decomposition Algorithms for Generalized Potential Games". In: *Comput. Optim. Appl.* 50.2 (2011), pp. 237–262.

[Fra02] A. Frangioni. "Generalized Bundle Methods". In: *SIAM Journal on Optimization* 13.1 (2002), pp. 117–156.

[GF10] S. A. Gabriel and J. D. Fuller. "A Benders Decomposition Method for Solving Stochastic Complementarity Problems with an Application in Energy". In: *Computational Economics* 35.4 (2010), pp. 301–329.

[GMH10] A. Gjelsvik, B. Mo, and A. Haugstad. "Long- and Medium-term Operations Planning and Stochastic Modelling in Hydro-dominated Power Systems Based on Stochastic Dual Dynamic Programming". In: *Handbook of Power Systems* (2010), pp. 33–35.

[GSC20] V. Guigues, A. Shapiro, and Y. Cheng. "Duality and sensitivity analysis of multistage linear stochastic programs". In: *http://www.optimization-online.org/DB_FILE/2019/11/7483.pdf* (2020).

[Guo+14] L. Guo, G.-H. Lin, J. J. Ye, and J. Zhang. "Sensitivity Analysis of the Value Function for Parametric Mathematical Programs with Equilibrium Constraints". In: *SIAM Journal on Optimization* 24.3 (2014), pp. 1206–1237.

[HBT18] L. Hellemo, P. Barton, and A. Tomasgard. "Decision-dependent probabilities in stochastic programs with recourse". In: *Computational Management Science* 15.3 (2018), pp. 369–395.

[HS06] J. L. Higle and S. Sen. "Multistage stochastic convex programs: Duality and its implications". In: *Annals of Operations Research* 142.1 (2006), pp. 129–146.

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I and II*. Springer Berlin Heidelberg, 1993.

[IM96] G. Infanger and D. P. Morton. "Cut sharing for multistage stochastic linear programs with interstage dependency". In: *Mathematical Programming* 75.2 (1996), pp. 241–256.

[Int18] International Hydropower Association. *The World's Water Battery: Pumped Hydropower Storage and the Clean Energy Transition*. 2018.

[IRE20] IRENA. "Renewable Energy Highlights". In: *irena.org/publications/2020/Jul/Renewable-energy-statistics-2020* (2020).

[IS06] A. Izmailov and M. Solodov. "A Note on Error Estimates for some Interior Penalty Methods". In: *Recent Advances in Optimization. Lecture Notes in Economics and Mathematical Systems* 563 (2006).

[IS14] A. Izmailov and M. Solodov. *Newton-type methods for optimization and variational problems*. Springer Series in Operations Research and Financial Engineering. Springer, Cham, 2014.

[JJBW02] A. Jofré and R. J.-B. Wets. "Continuity Properties of Walras Equilibrium Points". In: *Annals of Operations Research* 114.1-3 (2002), pp. 229–243.

[KBP01] R. Kelman, L. Barroso, and M. Pereira. "Market power assessment and mitigation in hydrothermal systems". In: *IEEE Transactions on Power Systems* 16.3 (2001), pp. 354–359.

[Kel60] J. Kelley. "The cutting-plane method for solving convex programs". In: *Journal of the Society for Industrial and Applied Mathematics* 8.1 (1960), pp. 703–712.

[Kel99] R. Kelman. "Competitive Schemes in Hydrothermal Systems: Economic Efficiency and Strategic Behaviour". In: *M.Sc Thesis, COPPE/UFRJ* (1999).

[KF19] Y. Kim and M. C. Ferris. “Solving equilibrium problems using extended mathematical programming”. In: *Mathematical Programming Computation* 11.3 (2019), pp. 457–501.

[KFS18] D. Kourounis, A. Fuchs, and O. Schenk. “Toward the Next Generation of Multiperiod Optimal Power Flow Solvers”. In: *IEEE Transactions on Power Systems* 33.4 (2018), pp. 4005–4014.

[KHF17] Y. Kim, O. Huber, and M. C. Ferris. “A structure-preserving pivotal method for affine variational inequalities”. In: *Mathematical Programming* 168.1-2 (2017), pp. 93–121.

[KM14] V. Kozmík and D. P. Morton. “Evaluating policies in risk-averse multi-stage stochastic programming”. In: *Mathematical Programming* 152.1–2 (2014), pp. 275–300.

[KS12] A. A. Kulkarni and U. V. Shanbhag. “Revisiting Generalized Nash Games and Variational Inequalities”. In: *Journal of Optimization Theory and Applications* (2012), pp. 1–12.

[KS16] C. Kanzow and D. Steck. “Augmented Lagrangian Methods for the Solution of Generalized Nash Equilibrium Problems”. In: *SIAM Journal on Optimization* 26.4 (2016), pp. 2034–2058.

[KSM02] A. J. Kleywegt, A. Shapiro, and T. H. de Mello. “The Sample Average Approximation Method for Stochastic Discrete Optimization”. In: *SIAM Journal on Optimization* 12.2 (2002), pp. 479–502.

[Lan93] S. Lang. *Real and Functional Analysis*. Graduate Texts in Mathematics, Springer, 1993.

[Lec+20] V. Leclère, P. Carpentier, J.-P. Chancelier, A. Lenoir, and F. Pacaud. “Exact Converging Bounds for Stochastic Dual Dynamic Programming via Fenchel Duality”. In: *SIAM Journal on Optimization* 30.2 (2020), pp. 1223–1250.

[Li+18] L. Li, L. Chang, P. Liu, and B. Yu. “Multi-Owner Scheduling for Cascade Hydro Power Using Multi-Objective Optimization Technique”. In: (2018).

[Liu+20] J. Liu, Y. Cui, J.-S. Pang, and S. Sen. *Two-Stage Stochastic Programming with Linearly Bi-parameterized Quadratic Recourse*. Tech. rep. 3. 2020, pp. 2530–2558.

[LP05] K. Linowsky and A. Philpott. “On the convergence of sampling based decomposition algorithms for multistage stochastic programs”. In: *Journal of Optimization Theory and Applications* 125.1 (2005), pp. 349–366.

[LQ15] F.-F. Li and J. Qiu. “Multi-Objective Reservoir Optimization Balancing Energy Generation and Firm Power”. In: *Energies* 8.7 (2015), pp. 6962–6976.

[LSS12] J. P. Luna, C. Sagastizábal, and M. Solodov. “A class of Dantzig–Wolfe type decomposition methods for variational inequality problems”. In: *Mathematical Programming* 143.1-2 (2012), pp. 177–209.

[LSS13] J. P. Luna, C. Sagastizábal, and M. Solodov. “Complementarity and Game-Theoretical Models for Equilibria in Energy Markets: Deterministic and Risk-Averse Formulations”. In: *Handbook of Risk Management in Energy Production and Trading* (2013), pp. 231–258.

[LSS16] J. P. Luna, C. Sagastizábal, and M. Solodov. “An approximation scheme for a class of risk-averse stochastic equilibrium problems”. In: *Mathematical Programming* 157.2 (2016), pp. 451–481.

[LSW06] J. Linderoth, A. Shapiro, and S. Wright. “The empirical behavior of sampling methods for stochastic programming”. In: *Annals of Operations Research* 142.1 (2006), pp. 215–241.

[LW03] J. Linderoth and S. Wright. “Decomposition Algorithms for Stochastic Programming on a Computational Grid”. In: *Computational Optimization and Applications* 24.1 (2003), pp. 207–250.

[MM20] R. Moita and D. Monte. “Hydroelectric Generators Competing in Cascades”. In: *Revista Brasileira de Economia* 74.1 (2020).

[MMF11] T. H. de Mello, V. L. de Matos, and E. C. Finardi. “Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling”. In: *Energy Systems* 2.1 (2011), pp. 1–31.

[MMF16] V. L. de Matos, D. P. Morton, and E. C. Finardi. “Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling”. In: *Annals of Operations Research* 253.2 (2016), pp. 713–731.

[MNY07] B. S. Mordukhovich, N. M. Nam, and N. D. Yen. “Subgradients of marginal functions in parametric mathematical programming”. In: *Mathematical Programming* 116.1-2 (2007), pp. 369–396.

[Mor06] B. S. Mordukhovich. *Variational Analysis and Generalized Differentiation I*. Springer Berlin Heidelberg, 2006.

[Mor18] B. S. Mordukhovich. *Variational Analysis and Applications*. Springer International Publishing, 2018.

[MW09] J. J. Moré and S. M. Wild. “Benchmarking Derivative-Free Optimization Algorithms”. In: *SIAM Journal on Optimization* 20.1 (2009), pp. 172–191.

[MZ98] R. D. Monteiro and F. Zhou. “On the Existence and Convergence of the Central Path for Convex Programming and Some Duality Results”. In: *Computational Optimization and Applications* 10.1 (1998), pp. 51–77.

[Nes04] Y. Nesterov. “Smooth minimization of non-smooth functions”. In: *Mathematical Programming* 103.1 (2004), pp. 127–152.

[Oli+10] W. L. de Oliveira, C. Sagastizábal, D. D. J. Penna, M. E. P. Maceira, and J. M. Damázio. “Optimal scenario tree reduction for stochastic streamflows in power generation planning problems”. In: *Optimization Methods and Software* 25.6 (2010), pp. 917–936.

[OS14] W. Oliveira and C. Sagastizábal. “Level bundle methods for oracles with on-demand accuracy”. In: *Optimization Methods and Software* 29.6 (2014). Charles Broyden Prize for best paper published by the journal in 2014., pp. 1180–1209.

[OSL14] W. Oliveira, C. Sagastizábal, and C. Lemaréchal. “Convex proximal bundle methods in depth: a unified analysis for inexact oracles”. In: *Mathematical Programming* 148.1-2 (2014), pp. 241–277.

[OSS11] W. Oliveira, C. Sagastizábal, and S. Scheimberg. “Inexact Bundle Methods for Two-Stage Stochastic Programming”. In: *SIAM Journal on Optimization* 21.2 (2011), pp. 517–544.

[PA20] P. Pérez-Aros. “Ergodic Approach to Robust Optimization and Infinite Programming Problems”. In: *preprint* (2020).

[PF18] A. Philpott and M. Ferris. “Dynamic Risked Equilibrium”. In: *preprint* (2018).

[PFW16] A. Philpott, M. Ferris, and R. Wets. “Equilibrium, uncertainty and risk in hydro-thermal electricity systems”. In: *Mathematical Programming* 157.2 (2016), pp. 483–513.

[PMF13] A. Philpott, V. de Matos, and E. Finardi. “On Solving Multistage Stochastic Programs with Coherent Risk Measures”. In: *Operations Research* 61.4 (2013), pp. 957–970.

[PP91] M. Pereira and L. Pinto. “Multi-stage stochastic optimization applied to energy planning”. In: *Mathematical Programming* 52.1 (1991), pp. 359–375.

[QSZ00] L. Qi, D. Sun, and G. Zhou. “A new look at smoothing Newton methods for nonlinear complementarity problems and box constrained variational inequalities”. In: *Mathematical Programming* 87.1 (2000), pp. 1–35.

[Roc99] R. Rockafellar. “Duality and optimality in multistage stochastic programming”. In: *Annals of Operations Research* 85.0 (1999), pp. 1–19.

[RU02] R. Rockafellar and S. Uryasev. “Conditional value-at-risk for general loss distributions”. In: *Journal of Banking & Finance* 26.7 (2002), pp. 1443–1471.

[RW09] T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer, 2009.

[RX05] D. Ralph and H. Xu. “Implicit Smoothing and Its Application to Optimization with Piecewise Smooth Equality Constraints”. In: *Journal of Optimization Theory and Applications* 124.3 (2005), pp. 673–699.

[Sag12] C. Sagastizábal. “Divide to conquer: decomposition methods for energy optimization”. In: *Mathematical Programming* 134.1 (2012), pp. 187–222.

[Sca73] H. E. Scarf. *The Computation of Economic Equilibria*. Yale University Press, 1973.

[Sch12] S. Schommer. “Computing equilibria in economies with incomplete markets, collateral and default penalties”. In: *Annals of Operations Research* 206.1 (2012), pp. 367–383.

[Sch+21] K. Schindler, N. Rujeerapaiboon, D. Kuhn, and W. Wiesemann. “A Planner-Trader Decomposition for Multi-Market Hydro Scheduling”. In: *preprint* (2021).

[Scu+11] G. Scutari, D. P. Palomar, F. Facchinei, and J.-S. Pang. “Distributed dynamic pricing for MIMO interfering multiuser systems: A unified approach”. In: *International Conference on NETwork Games, Control and Optimization* (2011).

[Scu+13] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang. “Decomposition by partial linearization in multiuser systems”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.

[Sha11] A. Shapiro. “Analysis of stochastic dual dynamic programming method”. In: *European Journal of Operational Research* 209.1 (2011), pp. 63–72.

[SM98] A. Shapiro and T. H. de Mello. “A simulation-based approach to two-stage stochastic programming with recourse”. In: *Mathematical Programming* 81.3 (1998), pp. 301–325.

[SMK18] B. Swenson, R. Murray, and S. Kar. “On Best Response Dynamics in Potential Games”. In: *SIAM Journal on Control and Optimization* 56.4 (2018), pp. 2734–2767.

[SW69] R. M. V. Slyke and R. Wets. “L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming”. In: *SIAM Journal on Applied Mathematics* 17.4 (1969), pp. 638–663.

[TD17] R. Taktak and C. D’Ambrosio. “An overview on mathematical programming approaches for the deterministic unit commitment problem in hydro valleys”. In: *Energy Systems* 8.1 (2017), pp. 57–79.

[TW20] G. Terça and D. Wozabal. “Envelope Theorems for Multi-Stage Linear Stochastic Optimization”. In: `http://www.optimization-online.org/DB_FILE/2018/06/6666.pdf` (2020).

[WB05] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106.1 (2005), pp. 25–57.

[WC17] X. Wu and A. J. Conejo. “An Efficient Tri-Level Optimization Model for Electric Grid Defense Planning”. In: *IEEE Transactions on Power Systems* 32.4 (2017), pp. 2984–2994.

[Wet66] R. J. B. Wets. “Programming Under Uncertainty: The Equivalent Convex Program”. In: *SIAM Journal on Applied Mathematics* 14.1 (1966), pp. 89–105.

[Wri97] S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1997.

[XWY14] M. Xu, S.-Y. Wu, and J. J. Ye. “Solving semi-infinite programs by smoothing projected gradient method”. In: *Computational Optimization and Applications* 59.3 (2014), pp. 591–616.

[XY10a] H. Xu and J. J. Ye. “Approximating Stationary Points of Stochastic Mathematical Programs with Equilibrium Constraints via Sample Averaging”. In: *Set-Valued and Variational Analysis* 19.2 (2010), pp. 283–309.

[XY10b] H. Xu and J. J. Ye. “Necessary Optimality Conditions for Two-Stage Stochastic Programs with Equilibrium Constraints”. In: *SIAM Journal on Optimization* 20.4 (2010), pp. 1685–1715.

[XY13] M. Xu and J. J. Ye. “A smoothing augmented Lagrangian method for solving simple bilevel programs”. In: *Computational Optimization and Applications* 59.1-2 (2013), pp. 353–377.

[XYZ14] M. Xu, J. J. Ye, and L. Zhang. “Smoothing augmented Lagrangian method for nonsmooth constrained optimization problems”. In: *Journal of Global Optimization* 62.4 (2014), pp. 675–694.

[XYZ15] M. Xu, J. J. Ye, and L. Zhang. “Smoothing SQP Methods for Solving Degenerate Nonsmooth Constrained Optimization Problems with Applications to Bilevel Programs”. In: *SIAM Journal on Optimization* 25.3 (2015), pp. 1388–1410.

[Yao+07] Y. Yao, T. Edmunds, D. Papageorgiou, and A. R. “Trilevel Optimization in Power Network Defense”. In: *IEEE Transactions on Systems* 37.4 (2007), pp. 712–718.

# Appendix

# Appendix A

# Appendix with tables

Table A.1: Deterministic run, Algorithm 1, values in Figure 5.2.

| Policy Type | Profit $l = 1$ | Profit $l = 2$ | Profit $l = 3$ | Tot. Profit |
|---|---|---|---|---|
| Social | 332.17 | 439.23 | 718.04 | 1489.45 |
| Individual | +4.18% | -5.97% | -3.73% | -2.62% |
| $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.02$ | +4.18% | -5.97% | -3.73% | -2.62% |
| $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.05$ | +4.18% | -5.59% | -3.70% | -2.50% |
| $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.10$ | +4.40% | -3.35% | -2.81% | -1.36% |
| $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.20$ | +5.32% | -1.66% | -2.40% | -0.04% |
| $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.40$ | +8.36% | -2.00% | -3.21% | -0.02% |
| $\tau_{2\to1} = \tau_{3\to2} = 0.20$, $\tau_{3\to1} = 0.00$ | +4.58% | -3.48% | -2.68% | -1.29% |
| $\tau_{2\to1} = \tau_{3\to2} = 0.40$, $\tau_{3\to1} = 0.00$ | +6.19% | -2.19% | -2.36% | -0.04% |

Table A.2: Stochastic run with profit sharing, values in Figure 5.7. The values reported are raw profits before any transfer. Percentage variations of the profits after transfer are reported in Table A.3.

| | Deterministic | | | Stochastic (Expected Value) | | |
|---|---|---|---|---|---|---|
| Policy Type | Profit $l = 1$ | Profit $l = 2$ | Profit $l = 3$ | Profit $l = 1$ | Profit $l = 2$ | Profit $l = 3$ |
| Social | 108.31 | 146.24 | 230.76 | 109.87 | 143.40 | 227.20 |
| Individual | 121.50 | 114.74 | 202.67 | 123.12 | 114.40 | 202.43 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.05$ | 121.48 | 115.21 | 202.67 | 122.25 | 116.80 | 205.52 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.10$ | 120.74 | 123.07 | 209.98 | 119.33 | 121.81 | 208.93 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.20$ | 120.74 | 122.78 | 211.76 | 119.40 | 126.09 | 211.32 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.30$ | 117.29 | 135.11 | 221.22 | 118.34 | 131.47 | 215.65 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.40$ | 117.30 | 135.11 | 221.22 | 115.87 | 133.04 | 217.70 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.50$ | 113.15 | 141.27 | 227.38 | 114.54 | 134.41 | 219.19 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.60$ | 111.68 | 143.92 | 227.38 | 109.61 | 136.78 | 222.52 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.70$ | 109.87 | 143.40 | 227.20 | 112.99 | 142.60 | 226.89 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.80$ | 109.06 | 146.24 | 229.70 | 107.68 | 144.28 | 228.38 |
| $\tau_{2\to1} = \tau_{3\to2} = 0.90$ | 108.31 | 146.24 | 230.76 | 106.73 | 143.14 | 227.76 |

Table A.3: This table continues the analysis of Table A.2. It shows the percentage variations of profits relative to the social policy after the profit-sharing payments. Recall that the payments are based on what exceeds the individualistic profits. It is interesting to note the results for $\tau_{2\to1}$ and $\tau_{3\to2}$ closer to 1. The profits before profit-sharing payments are closer to social profits and after the payments are closer to the individualistic profits. In these cases, hydro $l = 1$ acts analogously to a confiscatory agent.

| | Deterministic | | | Stochastic (Expected Value) | | |
|---|---|---|---|---|---|---|
| Policy Type | Profit $l = 1$ | Profit $l = 2$ | Profit $l = 3$ | Profit $l = 1$ | Profit $l = 2$ | Profit $l = 3$ |
| Individual | +12 % | -22 % | -12 % | +12 % | -20 % | -11 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.05$ | +12 % | -21 % | -12 % | +11 % | -19 % | -10 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.10$ | +12 % | -16 % | -09 % | +09 % | -15 % | -08 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.20$ | +13 % | -16 % | -09 % | +11 % | -13 % | -08 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.30$ | +15 % | -09 % | -07 % | +13 % | -10 % | -07 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.40$ | +19 % | -10 % | -07 % | +14 % | -10 % | -07 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.50$ | +22 % | -08 % | -07 % | +17 % | -10 % | -07 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.60$ | +27 % | -10 % | -08 % | +19 % | -11 % | -07 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.70$ | +31 % | -12 % | -09 % | +32 % | -11 % | -08 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.80$ | +40 % | -14 % | -10 % | +35 % | -13 % | -09 % |
| $\tau_{2\to1} = \tau_{3\to2} = 0.90$ | +47 % | -18 % | -11 % | +39 % | -17 % | -10 % |

# Appendix B

# Stochastic algorithm with cost sharing

We now give the precise description of the stochastic cost sharing method. The notation is the same of Section 5.4, except that the forward iterates associated with a forward path of scenarios $s = (s_1, \dots, s_T)$ is denoted as by $\hat{Z}_l^t = (\hat{x}_{ls}^t, \dots, \hat{x}_{ls}^{(\mathtt{T}-1)})$, where $\hat{x}_{ls}^t = \hat{x}_{lw}^t$, $w = s_t$, and with $\hat{Z}_l^{\mathtt{T}}$ an empty vector. The sequence of Algorithms 4 to 9 matches the sequence of steps described in Section 5.4. The name of those steps related to a forward-backward SDDP pass is shortened to FB Pass, for convenience.

---

**Algorithm 4** STOCHASTIC ALGORITHM WITH COST SHARING (PART 1)

---

**Initialization.** *Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$. Set $k = 1$. For $l = 1, 2, 3$, and $t \geq 2$, let $Q_{ls}^{t,k} \equiv -M$ and $U_{ls}^{t,k} \equiv -M$.*

**Step 1: Sampling.** *Obtain a sample $s_t^k \in \{1, \dots, S\}$ for each $t = 1, \dots, \mathtt{T}$.*

**Step 2: Get Feasible Iterates at** $l = 1$**.** *For each $t = 1, \dots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute*

$$\hat{x}_{ls}^{tk} \quad \textit{solving} \quad \begin{cases} \min & {f_{ls}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ \textit{s.t.} & y \in F_{ls}^t(\hat{x}_{l\omega}^{t-1,k}). \end{cases}$$

*For each $t = \mathtt{T}, \dots, 2$ and for each $r = 1, \dots, S$, take $\omega = s_{t-1}^k$ and compute*

$$x_{lr}^{tk} \quad \textit{solving} \quad \begin{cases} \min & {f_{lr}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ \textit{s.t.} & y \in F_{lr}^t(\hat{x}_{l\omega}^{t-1,k}). \end{cases}$$

**Step 3: Get Feasible Iterates at** $l = 2$**.** *For each $t = 1, \dots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute*

$$\hat{x}_{ls}^{tk} \quad \textit{solving} \quad \begin{cases} \min & {f_{ls}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ \textit{s.t.} & y \in F_{ls}^t(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,s}^{tk}). \end{cases}$$

*For each $t = \mathtt{T}, \dots, 2$ and for each $r = 1, \dots, S$, take $\omega = s_{t-1}^k$ and compute*

$$x_{lr}^{tk} \quad \textit{solving} \quad \begin{cases} \min & {f_{lr}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ \textit{s.t.} & y \in F_{lr}^t(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,r}^{tk}). \end{cases}$$

---

**Algorithm 5** STOCHASTIC ALGORITHM WITH COST SHARING (PART 2)

**Step 4: FB Pass at $l = 3$.** For each $t = 1, \dots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute

$$\hat{x}_{ls}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{ls}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, {Z_{l-1}^{t+1,k}}) \\ \text{s.t.} & y \in F_{ls}^t(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,s}^{tk}) \,. \end{cases}$$

**Step 4.1.** Take $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

**Step 4.2.** For $t = \mathtt{T}, \dots, 2$.

**Step 4.2.1: Cut computation.** For $r = 1, \dots, S$, compute

$$x_{lr}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{lr}^t}^\top y + \mathfrak{Q}_l^{t+1,k+1}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{lr}^t(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,r}^{tk}) \,. \end{cases}$$

Obtain subgradients such that for all $x_l^{t-1}, x_{l-1,r}^t$ and $Z_{l-1}^{t+1}$ the value function $Q_{lr}^t(x_l^{t-1}, x_{l-1,r}^t, Z_{l-1}^{t+1})$ lies below

$$\begin{array}{lcll} Q_{lr}^t(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,r}^{t,k}, Z_{l-1}^{t+1,k}) & + & (\lambda_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l\omega}^{t-1,k}) & + \\ (\mu_{l-1,r}^{t,k})^\top (x_{l-1,r}^t - x_{l-1,r}^{t,k}) & + & (\nu_{l-1}^{t+1,k})^\top (Z_{l-1}^{t+1} - Z_{l-1}^{t+1,k}). & \end{array} \tag{B.1}$$

**Step 4.2.1: Cut Aggregation.** Average the cuts in (B.1) to obtain a cut such that $\mathfrak{Q}_{lr}^t(x_l^{t-1,k}, Z_{l-1}^t)$ lies below

$$\mathfrak{Q}_{lr}^t(\hat{x}_{l\omega}^{t-1,k}, Z_{l-1}^{tk}) + (\phi_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l\omega}^{t-1,k}) + (\rho_{l-1}^{tk})^\top (Z_{l-1}^t - Z_{l-1}^{tk}). \tag{B.2}$$

Define $\mathfrak{Q}_{lr}^{t,k+1}$ as a maximum between $\mathfrak{Q}_{lr}^{t,k}$ and (B.2).

**Step 4.3: Calculation of Bounds.** Take $\overline{u}_l^k = \sum_t f_{ls}^t \hat{x}_{ls}^t$ where $s = s_t^k$. Take $\underline{u}_l^k$ as the optimal value of the first state problem after the backward step (Step 4.2).

**Algorithm 6** STOCHASTIC ALGORITHM WITH COST SHARING (PART 3)

**Step 5: Cost-Sharing to $l = 2$.** *For each stage $t = 1, \dots, \mathtt{T}$ and each scenario $s = 1, \dots, S$, take $\omega = s_{t-1}^k$ and compute subgradients*

$$(\alpha_{lts}^k, \beta_{lts}^k) \in \partial U_{l+1,s}^t(\hat{x}_{l+1,\omega}^{t-1,k}, x_{ls}^{tk}).$$

*Then, take $U_{l+1,s}^{t,k+1}$ as a maximum between $U_{l+1,s}^{t,k}$ and the affine function*

$$U_{l+1,s}^t(\hat{x}_{l+1,\omega}^{t-1,k}, x_{ls}^{tk}) + (\alpha_{lts}^k)^\top (x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) + (\beta_{lts}^k)^\top (x_{ls}^t - x_{ls}^{tk}).$$

*Note that the variables are $x_{l+1}^{t-1}$ and $x_{ls}^t$, which represent, respectively the forward decision at stage $t-1$ at level 3 and the decision taken at scenario $s$ and stage $t$ at level 2.*

**Algorithm 7** STOCHASTIC ALGORITHM WITH COST SHARING (PART 4)

**Step 6: FB Pass at** $l = 2$**.** *For each* $t = 1, \ldots, \mathtt{T}$*, take* $s = s_t^k$ *and* $\omega = s_{t-1}^k$*, and compute*

$$\hat{x}_{ls}^{tk} \quad \textit{solving} \quad \begin{cases} \min & {f_{ls}^{t}}^{\top} y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ s.t. & y \in F_{ls}^{t}(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,s}^{tk}). \end{cases}$$

**Step 6.1.** *Take* $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

**Step 6.2.** *For* $t = \mathtt{T}, \ldots, 2$.

**Step 6.2.1: Cut computation.** *For* $r = 1, \ldots, S$*, compute*

$$x_{lr}^{tk} \quad \textit{solving} \quad \begin{cases} \min & {f_{lr}^{t}}^{\top} y + \mathfrak{Q}_l^{t+1,k+1}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ s.t. & y \in F_{lr}^{t}(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,r}^{tk}). \end{cases}$$

*Obtain subgradients such that for all* $x_l^{t-1}, x_{l-1,r}^{t}, x_{l+1}^{t-1}, \hat{Z}_{l+1}^{t}$ *and* $Z_{l-1}^{t+1}$ *the value function* $Q_{lr}^{t}(x_l^{t-1}, x_{l-1,r}^{t}, x_{l+1}^{t-1}, \hat{Z}_{l+1}^{t}, Z_{l-1}^{t+1})$ *lies below*

$$\begin{array}{lclc} Q_{lr}^{t}(\hat{x}_{l\omega}^{t-1,k}, x_{l-1,r}^{t,k}, \hat{x}_{l+1,\omega}^{t-1}, \hat{Z}_{l+1}^{t,k}, Z_{l-1}^{t+1,k}) & + & (\lambda_l^{t-1,k})^{\top}(x_l^{t-1} - \hat{x}_{l\omega}^{t-1,k}) & + \\ (\mu_{l-1,r}^{t,k})^{\top}(x_{l-1,r}^{t} - x_{l-1,r}^{tk}) & + & (\nu_{l-1}^{t+1,k})^{\top}(Z_{l-1}^{t+1} - Z_{l-1}^{t+1,k}) & + \\ (\xi_{l+1,r}^{t,k})^{\top}(x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) & + & (\pi_{l+1}^{t+1,k})^{\top}(\hat{Z}_{l+1}^{t} - \hat{Z}_{l+1}^{t,k}). & \end{array} \qquad \text{(B.3)}$$

**Step 6.2.1: Cut Aggregation.** *Average the cuts in* (B.3) *to obtain a cut such that* $\mathfrak{Q}_{lr}^{t}(x_l^{t-1,k}, Z_{l-1}^{t}, \hat{Z}_{l+1}^{t-1})$ *lies below*

$$\begin{array}{lclc} \mathfrak{Q}_{lr}^{t}(x_l^{t-1,k}, Z_{l-1}^{tk}, \hat{Z}_{l+1}^{t,k}) & + & (\phi_l^{t-1,k})^{\top}(x_l^{t-1} - \hat{x}_{l\omega}^{t-1,k}) & + \\ (\rho_{l-1}^{tk})^{\top}(Z_{l-1}^{t} - Z_{l-1}^{tk}) & + & (\psi_{l+1}^{tk})^{\top}(\hat{Z}_{l+1}^{t-1} - \hat{Z}_{l+1}^{t-1,k}). & \end{array} \qquad \text{(B.4)}$$

*Define* $\mathfrak{Q}_{lr}^{t,k+1}$ *as a maximum between* $\mathfrak{Q}_{lr}^{t,k}$ *and* (B.4).

**Step 6.3: Calculation of Bounds.** *Take* $\overline{u}_l^k = \sum_t f_{ls}^t \hat{x}_{ls}^t + \tau_{\to l} \overline{u}_{l+1}^k$ *where* $s = s_t^k$*. Take* $\underline{u}_l^k$ *as the optimal value of the first state problem after the backward step (Step 6.2).*

**Algorithm 8** STOCHASTIC ALGORITHM WITH COST SHARING (PART 5)

**Step 7: Cost-Sharing to** $l = 1$**.** *For each stage* $t = 1, \ldots, \mathtt{T}$ *and each scenario* $s = 1, \ldots, S$*, take* $\omega = s_{t-1}^k$ *and compute subgradients*

$$(\alpha_{lts}^k, \beta_{lts}^k, \gamma_{lts}^k) \in \partial U_{l+1,s}^{t}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, x_{ls}^{tk}).$$

*Then, take* $U_{l+1,s}^{t,k+1}$ *as a maximum between* $U_{l+1,s}^{t,k}$ *and the affine function*

$$U_{l+1,s}^{t}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, x_{ls}^{tk}) + (\alpha_{lts}^k)^{\top}(x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) + (\beta_{lts}^k)^{\top}(x_{l+2}^{t-1} - \hat{x}_{l+2,\omega}^{t-1,k}) + (\gamma_{lts}^k)^{\top}(x_{ls}^{t} - x_{ls}^{tk}).$$

**Algorithm 9** Stochastic Algorithm with Cost Sharing (Part 6)

**Step 8: FB Pass at** $l = 1$**.** *For each* $t = 1, \dots, \mathtt{T}$*, take* $s = s_t^k$ *and* $\omega = s_{t-1}^k$*, and compute*

$$\hat{x}_{ls}^{tk} \quad \textit{solving} \quad \begin{cases} \min & f_{ls}^{t\,\top} y + \mathfrak{Q}_l^{t+1,k}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ s.t. & y \in F_{ls}^t(\hat{x}_{l\omega}^{t-1,k}) \,. \end{cases}$$

**Step 8.1.** *Take* $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

**Step 8.2.** *For* $t = \mathtt{T}, \dots, 2$.

**Step 8.2.1: Cut computation.** *For* $r = 1, \dots, S$*, compute*

$$x_{lr}^{tk} \quad \textit{solving} \quad \begin{cases} \min & f_{lr}^{t\,\top} y + \mathfrak{Q}_l^{t+1,k+1}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ s.t. & y \in F_{lr}^t(\hat{x}_{l\omega}^{t-1,k}) \,. \end{cases}$$

*Obtain subgradients such that for all* $x_l^{t-1}, \hat{Z}_{l+1}^t, \hat{Z}_{l+2}^t$ *and* $x_{l+1}^{t-1}, \hat{x}_{l+2}^{t-1}$ *the value function* $Q_{lr}^t(x_l^{t-1}, \hat{Z}_{l+1}^t, \hat{Z}_{l+2}^t, x_{l+1}^{t-1}, x_{l+2}^{t-1})$ *lies below*

$$\begin{array}{lclcl} Q_{lr}^t(\hat{x}_{l\omega}^{t-1,k}, \hat{Z}_{l+1}^{tk}, \hat{Z}_{l+2}^{t,k}, \hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}) & + & (\lambda_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l\omega}^{t-1,k}) & + \\ (\mu_{l+1,r}^{t,k})^\top (x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) & + & (\nu_{l+1}^{t+1,k})^\top (\hat{Z}_{l+1}^t - \hat{Z}_{l+1}^{t,k}) & + \\ (\xi_{l+2,r}^{t,k})^\top (x_{l+2}^{t-1} - \hat{x}_{l+2,\omega}^{t-1,k}) & + & (\pi_{l+2}^{t+1,k})^\top (\hat{Z}_{l+2}^t - \hat{Z}_{l+2}^{t,k}). & \end{array} \tag{B.5}$$

**Step 8.2.1: Cut Aggregation.** *Average the cuts in* (B.5) *to obtain a cut such that* $\mathfrak{Q}_{lr}^t(x_l^{t-1,k}, \hat{Z}_{l+1}^{t-1}, \hat{Z}_{l+2}^{t-1})$ *lies below*

$$\begin{array}{lclcl} \mathfrak{Q}_{lr}^t(x_l^{t-1,k}, \hat{Z}_{l+1}^{tk}, \hat{Z}_{l+2}^{t,k}) & + & (\phi_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l\omega}^{t-1,k}) & + \\ (\rho_{l+1}^{tk})^\top (\hat{Z}_{l+1}^{t-1} - Z_{l+1}^{tk}) & + & (\psi_{l+2}^{tk})^\top (\hat{Z}_{l+2}^{t-1} - \hat{Z}_{l+2}^{t-1,k}). & \end{array} \tag{B.6}$$

*Take* $\mathfrak{Q}_{lr}^{t,k+1}(x_l^{t-1,k}, \hat{Z}_{l+1}^{t-1}, \hat{Z}_{l+2}^{t-1})$ *as a maximum between* $\mathfrak{Q}_{lr}^{t,k}(x_l^{t-1,k}, \hat{Z}_{l+1}^{t-1}, \hat{Z}_{l+2}^{t-1})$ *and* (B.6).

**Step 8.3: Calculation of Bounds.** *Take* $\overline{u}_l^k = \sum_t f_{ls}^t \hat{x}_{ls}^t + \tau_{\to l} \overline{u}_{l+1}^k$ *where* $s = s_t^k$*. Take* $\underline{u}_l^k$ *as the optimal value of the first state problem after the backward step (Step 8.2).*

**Step 9: Stopping Test.** *Stop if for all* $l = 1, 2, 3$ *the average of* $\overline{u}_l^k$ *and* $\underline{u}_l^k$ *across k are close enough or the lower bounds* $\underline{u}_l^k$ *stabilized. Else, set* $k = k + 1$ *and go back to Step 1.*