Universidade Federal do Rio de Janeiro

On finite and infinite horizon problems

Vitor Luiz Pinto de Pina Ferreira

Rio de Janeiro

Maio de 2022

Universidade Federal do Rio de Janeiro

On finite and infinite horizon problems

Vitor Luiz Pinto de Pina Ferreira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Advisor: Bernardo Freitas Paulo da Costa

Rio de Janeiro

Maio de 2022

Ferreira, Vitor Luiz Pinto de Pina
On finite and infinite horizon optimization problems / Vitor Luiz Pinto de Pina Ferreira. - Rio de Janeiro: UFRJ/IM, 2022.
94f.: il.; 29,7cm.
Orientador: Bernardo Freitas Paulo da Costa
Dissertação (Mestrado) - UFRJ/IM/Programa de Pós-Graduação em Matemática, 2022.
Referências Bibliográficas: f. 81-83
a. Stochastic programming. b. Convex optimization. c. Multistage problems. d. Infinite horizon. e. Cutting-plane method. I. Freitas Paulo da Costa, Bernardo. II. Universidade Federal do Rio de Janeiro, Programa de Pós-Graduação em Matemática. III. Título.

On finite and infinite horizon optimization problems

Vitor Luiz Pinto de Pina Ferreira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Aprovada por:

Prof. Bernardo Freitas Paulo da Costa (Orientador)

Prof. Claudia Alejandra Sagastizabal

Prof. Daniel Sadoc Menasché

Prof. Vincent Leclère

Rio de Janeiro

Maio de 2022

Resumo

On finite and infinite horizon optimization problems

Vitor Luiz Pinto de Pina Ferreira

Orientador: Bernardo Freitas Paulo da Costa

Nós estudamos programas estocásticos convexos em um contexto multiestágio e programas convexos em um contexto de horizonte infinito periódico. Nossas principais contribuições estão relacionadas ao caso de horizonte infinito periódico. Primeiro, inspirados por métodos de planos cortantes existentes, nós apresentamos dois algoritmos de solução, chamados exaustivo e exploratório, e provamos sua convergência. O algoritmo exaustivo calcula cortes apenas em um conjunto finito pré-selecionado de estados. O algoritmo exploratório gera uma sequência finita de estados onde novos cortes são calculados. Segundo, conjecturas naturais sobre tais problemas são examinadas.

Palavras-chave: Stochastic programming; Convex optimization; Multistage problems; Infinite horizon; Cutting-plane method.

Abstract

On finite and infinite horizon optimization problems

Vitor Luiz Pinto de Pina Ferreira

Advisor: Bernardo Freitas Paulo da Costa

We study convex stochastic programs in a multistage setting and convex programs in a periodical infinite horizon setting. Our main contributions are related to the periodical infinite horizon case. First, inspired by existing cutting-plane methods, we present two solution algorithms, called exhaustive and exploratory, and prove their convergence. The exhaustive algorithm computes cuts only at a finite, preselected set of states. The exploratory algorithm generates a finite sequence of states where new cuts are computed. Second, natural conjectures about such problems are examined.

Keywords: Stochastic programming; Convex optimization; Multistage problems; Infinite horizon; Cutting-plane method.

Contents

Chapter 1

Introduction

Decision-making inquires what decisions are possible, and which of those decisions is the best. Our focus is to answer the latter question for problems that can be described under the framework of mathematical optimization. The formulation of an optimization model dictates the methods available to solve it efficiently, often by exploiting the particular structure of a class of problems. It is here that the importance of convexity in optimization cannot be understated. The available tools for analyzing and solving convex optimization problems make them more tractable than general nonconvex problems. In fact, they often serve to develop solution methods for nonconvex problems; for example, via successive convex approximations.

We further concentrate on a paradigm fit for sequential decision-making: multistage optimization. The basic goal of multistage optimization is to obtain the sequence of decisions to be carried out over a finite amount of discrete periods of time. Predicting the future is a different—but related—problem, hence planning in anticipation of future events falls under the purview of decision-making under uncertainty. In that context, we introduce stochastic optimization problems as our approach to incorporate forecasts into our models. As time passes and new information becomes known, that data can be used to instruct our decisions. This characterizes the decision taken at the beginning of the planning period not as a single value, but as a function of this data.

In a sequential problem, performing the chosen action (regardless of how it was chosen) and advancing to the next time period results in a problem one time period smaller. This recursive property facilitates the study of multistage optimization problems. Dynamic Programming, introduced by Bellman in the early 1950s (cf. [1]), is a paradigm that makes use of this feature to mainly solve multistage problems where decisions and resources are discrete (often specifically finite). A key development for the continuous setting was the SDDP algorithm (cf. [23]). Originally constructed to solve linear multistage problems by means of approximating the cost of future actions by piecewise linear

functions, many variations have since been developed, including extensions to convex multistage problems.

As the planning horizon of models become longer and longer, the mathematical mind may ponder about problems that cover an infinite number of time periods. We are going to see that such problems can be defined precisely, and lay requirements for their well-posedness. Furthermore, rather than a Sisyphean task, a practical solution can be computed as long as the problem has a cyclical nature. Many phenomenons of interest—the climate; mass consumption of goods and services—are periodical or approximately periodical, and thus fit this assumption.

Prior to presenting our main contributions, we highlight two existing approaches to solving periodical infinite horizon optimization problems with continuous state and control variables. The first approximates the problem by discretizing the state or control variables. Such procedures had already been developed before SDDP (cf. [3]), however the GDDP algorithm (cf. [30]) is a recent proposal that combines state discretization with lower bounding functions (not necessarily linear, as those used in SDDP). The second approach is to solve finite horizon approximations. Suggestions of SDDP-type algorithms that implement this approach have been made in the literature (cf. [12] and [27]), but without a convergence proof. Another extension of SDDP to infinite horizon problems, the Benders squared algorithm, was proposed in [21] with a convergence proof.

The main contributions of this work relate to the aforementioned infinite horizon problems. To be precise, they are related to solution algorithms and conjectures regarding the properties of such problems, and we list them below.

- The *exploratory algorithm*, a SDDP-type method for solving infinite horizon optimization problems, alongside a convergence proof. The solution method is similar to the proposals in [12] and [27];
- The *exhaustive algorithm*, combining discretization of the state variable with linear lower bounding functions;
- Simple counterexamples to natural conjectures about properties of optimal solutions of periodical infinite horizon optimization problems, *e.g.* if the optimal solution is also periodic with the same period (the precise mathematical formulation is posed and answered later).

This dissertation is divided into three chapters. Each chapter focuses on a particular kind of optimization problem, building on top of the previous by introducing the theory and methods required by the problems of interest.

- In chapter Preliminaries, we introduce basic concepts of convex analysis and optimization.

Section 2.1 defines convex sets, lists examples, and shows properties useful to certify convexity or construct new convex sets.

Section 2.2 presents convex functions and their extended-valued extensions, defines the notions of epigraph, and explains how convex sets and convex functions are related to each other.

Section 2.3 characterizes a general mathematical optimization problem.

Section 2.4 is dedicated to the important notion of duality in optimization.

Section 2.5 explains a generalization of differentials particularly applicable in the context of convex functions.

- In chapter Finite horizon, we discuss both multistage and stochastic optimization problems.

  Section 3.1 presents the simpler case of deterministic two-stage problems and its nested decomposition.

  Section 3.2 introduces uncertainty via stochastic two-stage problems.

  Section 3.3 extends the discussion to multistage problems of arbitrary finite length, both deterministic and stochastic.

  Section 3.4 lists solution methods for multistage problems.

- In chapter Infinite horizon, we analyze optimization problems over an infinite number of periods.

  Section 4.1 introduces the infinite horizon setting, which is analyzed by means of the Wald-Bellman operator.

  Section 4.2 lists solution methods for infinite horizon problems.

  Section 4.3 contains numerical examples of practical and theorical interest.

  Section 4.4 concludes the dissertation by suggesting extensions to the examples previously presented.

- In the Appendix, section A.1 lists solution methods for the optimization problems discussed in chapter 2.

Chapter 2

Preliminaries

Suppose that an organization is contractually responsible for supplying energy to a third-party. The organization must satisfy a demand d. To that end it disposes of two power generation facilities, to which it must assign generation targets: one hydro power plant h and one thermal power plant g. Both plants have a maximum capacity of $\overline{h}$ and $\overline{g}$ respectively. Each unit of energy produced by the thermal plant incurs a cost of c, while the costs of energy production at the hydro plant are deemed negligible. Finally, if the organization fails to meet the demand, it must pay a penalty p that scales linearly with the deficit df in power supply, and cannot have a deficit larger than $\overline{df}$. The goal of the organization is to find an operation plan that minimizes the operational costs and yet satisfies the power demand. The described problem can be mathematically formulated as

$$\begin{array}{ll} \min\limits_{h,g,df} & cg + pdf \\ \text{s.t.} & h + g + df = d, \\ & 0 \leq h \leq \overline{h}, \\ & 0 \leq g \leq \overline{g}, \\ & 0 \leq df \leq \overline{df}. \end{array} \tag{2.1}$$

Problem (2.1) is called a *mathematical optimization problem*, or simply an *optimization problem*. The quantities $(h, g, df) \in \mathbb{R}^3$ are called *optimization variables*, the function $f : \mathbb{R}^3 \to \mathbb{R}$ given by $f(h, g, df) = cg + pdf$ is called the *objective function*, and the equality and inequality equations listed above are the *constraints* the optimization variables are subject to.

Generally, optimization problems can take the form

$$\begin{array}{ll} \min\limits_{x} & f_0(x) \\ \text{s.t.} & f_i(x) \leq b_i, \quad \forall i, \end{array} \tag{2.2}$$

where the f_i are constraint functions, and the b_i are bounds for the constraints. For example, setting problem 2.1 in this form, one might take $f_1(h, g, df) = h$

with $b_1 = \overline{h}$ for the constraint $h \leq \overline{h}$; and $f_2(h, g, df) = -h$ with $b_2 = 0$ for the constraint $0 \leq h$; and so on for each other inequality in problem 2.1. The remaining equality constraint $h + g + df = d$ can be broken down into a pair of inequality constraints, $h + g + df \leq d$ and $-h - g - df \leq -d$.

Problem 2.1 in particular falls into a class of optimization problems known as *linear programs* (abbreviated *LP*), *i.e.*, in the general notation of problem 2.2, the objective function f_0 and the constraint functions f_i are *linear*. Remember that a function $f : \mathbb{R}^n \to \mathbb{R}$ of n real variables is said to be linear if

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$.

Linear programming theory is rich, and it finds practical applications in a wide range of fields. Unfortunately, not all practical problems can be solved efficiently exclusively through the use of linear programs. Nonlinear optimization problems are, in general, much harder to solve efficiently than linear problems. Fortunately, there is a larger class of optimization problems which has been studied thoroughly that is of extreme theorical and practical interest: *convex optimization problems*. *Convex functions* can be thought of a generalization of linear functions, where a convex function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1$, $\alpha \geq 0$, and $\beta \geq 0$.

In order to introduce the theory of convex optimization, we find it best to begin with convex sets, and trace back to convex functions after a solid foundation has been stablished. Section 2.1 defines convex sets and exhibits some examples. Additionally, it presents some methods that can be used to construct new convex sets or to verify that a set is convex. Section 2.2, in turn, defines convex functions and shows their fundamental property that makes them particularly useful for optimization. Also, it defines the epigraph of a function, which allows us to link convex functions and convex sets. Section 2.3 details what a mathematical optimization problem is and its elements. Section 2.4 presents the idea of the dual optimization problem, and how it can be used to bound the original problem (called the primal problem). Section 2.5 introduces a generalization of differentials, useful when dealing with nondifferentiable convex functions. Solution methods for the optimization problems discussed in this chapter are explained in section A.1 of the Appendix: subsection A.1.1 focuses on convex problems, while subsection A.1.2 pinpoints on linear problems.

2.1 Convex sets

In this section, we introduce the notion of convex sets and some of their properties.

Let X be a Banach space over $\mathbb{R}$, *i.e.* a complete normed vector space. For most of this dissertation, the only Banach spaces considered are $\mathbb{R}^n$. For every $x_0, x_1 \in X$, and every $\theta \in [0,1]$, let

$$x_\theta = \theta x_1 + (1-\theta)\, x_0.$$

The set $\{x_\theta : \theta \in [0,1]\} =: [x_0, x_1]$ is called the *line segment* between x_0 and x_1, and is the natural generalization of line segments laying in $\mathbb{R}^n$ for $n \in \mathbb{N}$, as the notation suggests.

Definition 2.1.1. A set $C \subset X$ is called a *convex set* if the line segment between any two of its points lays in C. In other words, for any $x, y \in C$, and any $\theta \in [0,1]$, we have

$$\theta x + (1-\theta)\, y \in C.$$

Examples of convex sets include: the empty set, sets with a single point, and the whole space X. A less trivial example arises from problem 2.1. The set

$$\left\{ (h, g, df) \in \mathbb{R}^3 \,:\, h+g+df = d, \begin{array}{rcl} 0 \leq & h & \leq \overline{h}, \\ 0 \leq & g & \leq \overline{g}, \\ 0 \leq & df & \leq \overline{df}. \end{array} \right\}$$

is a convex set of a kind known as *polyhedron*.

Definition 2.1.2 (Polyhedron)**.** A set $P \subset X$ is called a *polyhedron* if it can be described by a *finite* number of linear equations, that is, there are finite sets $\Phi, \Psi \subset X^*$ of continuous linear operators and their respective bounds $\{a_\phi\}_{\phi\in\Phi}, \{b_\psi\}_{\psi\in\Psi} \subset \mathbb{R}$ such that

$$P = \{x \in X \,:\, \phi(x) \leq a_\phi, \forall \phi \in \Phi,\, \psi(x) = b_\psi, \forall \psi \in \Psi\}.$$

A polyhedron described by a single linear equality is called a *hyperplane*, while one described by a single linear inequality is called a *halfspace*. Therefore, polyhedra are finite intersections of hyperplanes and/or halfspaces.

A hyperplane described by the equation $\phi(x) = a$ divides X into two halfspaces, $\phi(x) \leq a$ and $\phi(x) \geq a$, hence the name.

A *vertex* of a polyhedron is any point $v \in P$ that can only be described as a convex combination trivially, *i.e.*, if $v = \theta x + (1-\theta)y$, then x or y equals v.

Below, we prove that polyhedra are indeed convex sets directly from the definition. Afterwards, we will see properties that simplify the process of determining if a given set is convex, rather than needing to return to the definition each time.

Theorem 2.1.3 (Polyhedra are convex)**.** *Let P be a polyhedron. Then P is convex.*

Proof. Let $x_0, x_1 \in P$, and $\theta \in [0,1]$. Then, for any $\phi \in \Phi$ we can evaluate

$$\phi(x_\theta) = \phi(\theta x_1 + (1-\theta)x_0) = \theta\phi(x_1) + (1-\theta)\phi(x_0) \leq \theta a_\phi + (1-\theta)a_\phi = a_\phi,$$

where the second equality follows from linearity, and the inequality from the fact that $x_0, x_1 \in P$. Thus $\phi(x_\theta) \leq a_\phi$. Similarly, one can show that $\psi(x_\theta) = b_\psi$. Because these equations hold for every $\phi \in \Phi$ and $\psi \in \Psi$, we have shown that $x_\theta \in P$ as desired. □

The Euclidean balls in $\mathbb{R}^n$ serve as an example of a convex set that is not a polyhedron, such as the unit ball around the origin B_0. It is nonetheless the intersection of an infinite number of halfspaces, those on the side of hyperplanes tangent to the boundary of the ball that contain the ball itself. This kind of description of the Euclidean ball—as an infinite intersection of convex sets—is sufficient to conclude that it is convex, as we will now see.

Theorem 2.1.4 (Intersection preserves convexity)**.** *Let $C_1, C_2 \subset X$ be convex sets. Then $C_1 \cap C_2$ is a convex set. In fact, let $\{C_\lambda\}_{\lambda\in\Lambda}$ be an arbitrary collection of convex subsets of X. Then $\bigcap_{\lambda\in\Lambda} C_\lambda$ is a convex set.*

Proof. Let $x, y \in \bigcap_{\lambda\in\Lambda} C_\lambda$. Then, for any particular $\lambda \in \Lambda$, $x, y \in C_\lambda$. By convexity, $[x,y] \subset C_\lambda$. Since λ was taken arbitrarily, $[x,y] \subset \bigcap_{\lambda\in\Lambda} C_\lambda$. □

Remark 2.1.5. The preceding theorem allows us to reduce the proof that polyhedra are convex to proving that hyperplanes and halfspaces are convex. (In fact, only a proof for halfspaces is necessary, as hyperplanes are the intersection of the halfspaces defined by it.)

Related to Euclidean balls in $\mathbb{R}^n$ are ellipsoids, which can be described as the image of an Euclidean ball under a linear transformation. They are, as expected, convex, which again follows from a more general fact.

Theorem 2.1.6 (Linear maps preserve convexity)**.** *Let $T : X \to Y$ be a continuous linear operator between Banach spaces, and let $C \subset X$ and $S \subset Y$ be convex sets. Then $T(C)$ and $T^{-1}(S)$ are convex sets.*

Proof. Let $x_0, x_1 \in C$, and $\theta \in [0,1]$. Then by convexity,

$$x_\theta = \theta x_1 + (1-\theta)\, x_0 \in C,$$

and thus by linearity

$$\begin{aligned} \theta T(x_1) + (1-\theta)\, T(x_0) &= T(\theta x_1 + (1-\theta)\, x_0) \\ &= T(x_\theta) \in T(C). \end{aligned}$$

Now, if $T^{-1}(S)$ is empty, it is convex by vacuity. Otherwise, let $z_0, z_1 \in T^{-1}(S)$ and $\theta \in [0,1]$. Then we can denote their convex combination by

$$z_\theta = \theta z_1 + (1-\theta) z_0 \in X.$$

Let $y_0, y_1 \in S$ refer to the elements of S such that $T(z_0) = y_0$ and $T(z_1) = y_1$. As a point in the domain of T, we can calculate the image of z_θ,

$$\begin{aligned} T(z_\theta) &= T(\theta z_1 + (1-\theta) z_0) \\ &= \theta T(z_1) + (1-\theta) T(z_0) \\ &= \theta y_1 + (1-\theta) y_0 \in S, \end{aligned}$$

showing that $z_\theta \in T^{-1}(S)$, as desired. □

In some situations, there is a nonconvex set and a convex set is desired. The following construction is useful in a variety of such cases.

Definition 2.1.7. Let $A \subset X$ be an arbitrary set. We define the *convex hull of* A as the set of all convex combinations of elements of A, that is,

$$\mathbf{conv}\, A \coloneqq \left\{ \sum_{i=0}^{n} \theta_i a_i \,:\, a_i \in A, \theta_i \in [0,1], \forall i \in [0..n], \forall n \in \mathbb{N} \right\}.$$

As its name suggests, the convex hull of A is always convex, even if A is not. Additionally, it is the smallest convex set that contains A. We now prove these two facts.

Proof. First, let $x_0, x_1 \in \mathbf{conv}\, A$, and $\theta \in [0,1]$. By definition of the convex hull, we may write $x_j = \sum_{i=0}^{n_j} \theta_{i,j} a_{i,j}$. Therefore,

$$x_\theta = \theta x_1 + (1-\theta) x_0 = \sum_{i=0}^{n_1} \theta\theta_{i,1} a_{i,1} + \sum_{i=0}^{n_0} (1-\theta)\theta_{i,0} a_{i,0},$$

which is a weighted sum of $n_0 + n_1$ elements of A. Now, the weights are all non-negative, and each is at most 1. The sum of the weights can be written as

$$\sum_{i=0}^{n_1} \theta\theta_{i,1} + \sum_{i=0}^{n_0} (1-\theta)\theta_{i,0} = \sum_{i=0}^{n_0} \theta_{i,0} + \theta \left(\sum_{i=0}^{n_1} \theta_{i,1} - \sum_{i=0}^{n_0} \theta_{i,0} \right) = 1.$$

Thus x_θ is a convex combination of elements of A, and $x_\theta \in \mathbf{conv}\, A$.

Now, let $\{C_\lambda\}_{\lambda \in \Lambda}$ be the collection of convex sets that contain A. As $\mathbf{conv}\, A$ is one such set, it is immediate that $\mathbf{conv}\, A \supset \bigcap_{\lambda \in \Lambda} C_\lambda$. Let $x \in \mathbf{conv}\, A$, that is, we may write $x = \sum_{i=0}^{n} \theta_i a_i$. For any $\lambda \in \Lambda$, we have $a_i \in C_\lambda$ for each $i \in [0..n]$. Because each C_λ is convex, $x = \sum_{i=0}^{n} \theta_i a_i$ belongs to C_λ. Since we begun by taking an arbitrary element x of $\mathbf{conv}\, A$, $\mathbf{conv}\, A \subset C_\lambda$, showing $\mathbf{conv}\, A \subset \bigcap_{\lambda \in \Lambda} C_\lambda$, and in fact $\mathbf{conv}\, A = \bigcap_{\lambda \in \Lambda} C_\lambda$. □

2.2 Convex functions

In this section, we characterize convex functions and relate them to convex sets.

Definition 2.2.1. Let $C \subset X$ be a convex set. A function $f : C \to \mathbb{R}$ is called a *convex function* if the line segment between two points in its graph lies above said graph. In mathematical notation, for any $x_0, x_1 \in C$, and $\theta \in [0, 1]$, we have

$$f(\theta x_1 + (1 - \theta)x_0) \leq \theta f(x_1) + (1 - \theta)f(x_0).$$

A function is said to be *strictly convex* if the inequality is strict whenever $x_0 \neq x_1$. A function f is a *concave function* if $-f$ is convex, and, likewise, it is *strictly concave* if $-f$ is strictly convex.

We have seen examples of convex functions in previous sections. *Linear functions*, and, more generally, *affine functions*, are examples of convex functions. However, affine functions are not only convex functions, but also concave. The converse is true: any function that is both convex and concave is affine.

Recall problem 2.1: its objective function,

$$f_0(h, g, df) = cg + pdf,$$

is a linear function of h, g, and df; meanwhile, the constraint function

$$f_1(h, g, df) = h + g + df - d$$

is an affine function of the same variables. Remember that c, p, and d are constant real numbers. Note that this results in the constraint $f_1(h, g, df) = 0$. Alternatively, we could define

$$\tilde{f}_1(h, g, df) = h + g + df,$$

a linear function, and use the constraint $\tilde{f}_1(h, g, df) = d$ instead. (In order to cast problem 2.1 as an LP, $\tilde{f}_1$ should be used.)

Another family of examples that arise naturally are norms defined on a Banach space. For instance, as functions of real numbers x and y, both $|x|+|y|$ and $\sqrt{x^2 + y^2}$ are convex functions. [1]

One of the main reasons convex functions are useful for optimization is that *a local minimum of a convex function is the global minimum.* Note that

[1] Convexity is independent of the norm of the ambient space. In particular, two norms that are not equivalent (in an infinite-dimensional context) are nonetheless convex, whichever norm is chosen to define the Banach space at hand. Continuity, on the other hand, is affected.

the global minimum can be attained at multiple points (as in the constant functions) or not at all (as in the exponential function). Prior to formally stating and proving this property, we need the following definition.

Definition 2.2.2 (Relative openness)**.** Let $A \subset C \subset X$. We say that A is *relatively open relative to* C if there exists an open $U \subset X$ such that $A = C \cap U$.[2]

The motivation for the preceding definition is the common need of considering a convex set $C \subset X$ whose dimension is less than that of the entire space. For example, the set $\{(x, y) \in \mathbb{R}^2 : x = 0, 0 \leq y \leq 1\}$ is a line segment in two-dimensional space. An open ball around an arbitrary point $(0, y)$ in the line segment would contain points of the form (ε, y), where $\varepsilon > 0$, outside of the segment. We do not need to make this notion of dimension precise; nonetheless, the next theorem codifies the required concept of local minimum.

Theorem 2.2.3 (Fundamental property of convex functions)**.** *Let $f : C \subset X \to \mathbb{R}$ be a convex function, and let $A \subset C$ be a non-empty set relatively open to C. If f attains a minimum at $x^\star \in A$, then $x^\star$ minimizes f over C.*

Proof. Assume, for contradiction purposes, that there exists $x \in C$ with $f(x) < f(x^\star)$. Denote by $d = x - x^\star$ the vector in the direction from $x^\star$ to x. Because A is open, there exists a sufficiently small $\theta \in [0, 1]$ such that $x^\star + \theta d \in A$. However, by convexity,

$$f(x^\star + \theta d) = f((1 - \theta)x^\star + \theta x) \leq (1 - \theta)f(x^\star) + \theta f(x) < f(x^\star),$$

contradicting the fact that $f(x^\star) = \min_{x \in A} f(x)$. □

There is another simple yet useful class of convex functions, that require the introduction of the following concepts. The *extended real numbers* are

$$\overline{\mathbb{R}} \coloneqq \{-\infty\} \cup \mathbb{R} \cup \{\infty\}.$$

Another way to denote them is $\overline{\mathbb{R}} = [-\infty, \infty]$, which highlights the shared properties of $\overline{\mathbb{R}}$ and the compact intervals of $\mathbb{R} = (-\infty, \infty)$.

Definition 2.2.4. Let $f : C \subset X \to \mathbb{R}$ be a convex function. The *extended-value extension of* f is the function $\overline{f} : X \to \overline{\mathbb{R}}$ given by

$$\overline{f}(x) = \begin{cases} f(x), & \text{if } x \in C, \\ \infty, & \text{otherwise.} \end{cases}$$

We denote both f and $\overline{f}$ by the same symbol whenever there is no ambiguity. The *domain of* $\overline{f}$ is the set $\mathbf{dom}\, \overline{f}$ where $\overline{f}$ is finite. Here, for example, $\mathbf{dom}\, \overline{f} = C$.

[2] In topology, this is known as subspace topology, relative topology, or induced topology.

Note that the above definition precludes convex functions from attaining the value $-\infty$. It is possible to include such constructions in the definition, but we have no need for them in this text.

Now, we can introduce the family of *extended-value indicator functions*. For any set $C \subset X$, we define the function $\mathbb{I}_C : X \to \overline{\mathbb{R}}$ by

$$\mathbb{I}_C(x) = \begin{cases} 0, & \text{if } x \in C, \\ \infty, & \text{otherwise.} \end{cases}$$

If C is a convex set, $\mathbb{I}_C$ is a convex function. In fact, $\mathbb{I}_C$ can be seen as the extension of the constant function defined only on C and equal to 0 in it. We may also write $\mathbb{I}[x \in C]$ to refer to $\mathbb{I}_C$, especially when the set C can be described in simple terms (*e.g.* $\mathbb{I}[x \geq 0]$ when $C = \{x \geq 0 : x \in \mathbb{R}\}$).

The reader may have come across another definition of indicator functions, namely a function f that is piecewise-constant, taking the value 1 on a set C, and the value 0 everywhere else. These functions are usually neither convex nor concave, and thus not particularly useful to us.

Theorem 2.2.5. *Let $\{f_n\}_{n\in\mathbb{N}}$ be a sequence of convex functions defined on the same Banach space X. If f_n converges to f pointwise, then f is convex.*

Proof. Let $x_0, x_1 \in X$, and $\theta \in [0, 1]$. Then for $x_\theta = \theta x_1 + (1-\theta)x_0$, and any $n \in \mathbb{N}$,

$$f_n(x_\theta) \leq \theta f_n(x_1) + (1-\theta) f_n(x_0),$$

since f_n is convex. Taking the limit as $n \to \infty$ on both sides, we obtain

$$f(x_\theta) \leq \theta f(x_1) + (1-\theta) f(x_0).$$

Because this inequality is valid for arbitrary x_0, x_1, and θ, we conclude that f is convex. □

The following definition helps bridge the gap between convex sets and convex functions. It is also an useful tool for analysing and modeling convex optimization problems (as we will see in *e.g.* example 2.3.5).

Definition 2.2.6. Let f be a real-valued function. The *epigraph of f* is the set

$$\operatorname{epi} f = \{(x, t) : f(x) \leq t\},$$

that is, the epigraph of a function is the region above (and including) the graph of a function.

Figure 2.1 depicts an example.

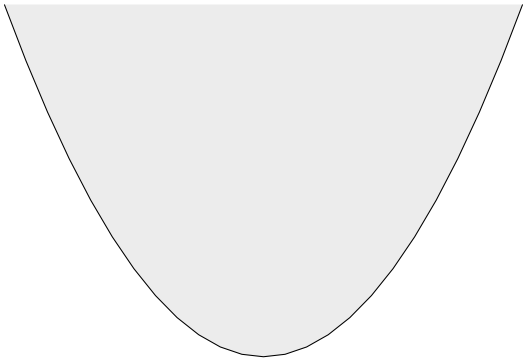

Figure 2.1: Epigraph of the quadratic function $f(x) = x^2$ (shaded in gray).

The epigraph is a particular set associated with any real-valued function. However, when the function under consideration is convex, its epigraph is a convex set. Conversely, a function whose epigraph is convex is a convex set. This is precisely the connection between the two different kinds of convex objects we studied so far.

Theorem 2.2.7. *Let f be a real-valued function. Then f is a convex function if and only if its epigraph is a convex set.*

Proof. Let $f : C \to \mathbb{R}$ be a convex function, $x_0, x_1 \in C$, and $\theta \in [0, 1]$. Take $t_0, t_1 \in \mathbb{R}$ such that $y_0 = (x_0, t_0)$ and $y_1 = (x_1, t_1)$ are points in $\operatorname{epi} f$. Then the point $y_\theta = \theta y_1 + (1 - \theta)y_0$ has coordinates

$$\Big(\theta x_1 + (1 - \theta)x_0, \quad \theta t_1 + (1 - \theta)t_0\Big).$$

Because y_0 and y_1 are points in the epigraph of f, $t_0 \geq f(x_0)$ and $t_1 \geq f(x_1)$, and thus

$$\theta t_1 + (1 - \theta)t_0 \geq \theta f(x_1) + (1 - \theta)f(x_0) \geq f(\theta x_1 + (1 - \theta)x_0),$$

where the second inequality follows from convexity. This shows $y_\theta \in \operatorname{epi} f$.

Now, assume $f : X \to \mathbb{R}$ is a function whose epigraph is a convex set. Let $x_0, x_1 \in X$, $y_0 = (x_0, f(x_0))$ and $y_1 = (x_1, f(x_1))$, and $\theta \in [0, 1]$. By convexity of the epigraph, $y_\theta = \theta y_1 + (1 - \theta)y_0 \in \operatorname{epi} f$. Because y_θ has coordinates

$$\Big(\theta x_1 + (1 - \theta)x_0, \quad \theta f(x_1) + (1 - \theta)f(x_0)\Big),$$

this implies $\theta f(x_1)+(1-\theta)f(x_0) \geq f(\theta x_1+(1-\theta)x_0)$, which shows f is a convex function. (Note that $f(\theta x_1+(1-\theta)x_0)$ is a real number because f is real-valued, and not *extended real-valued*. It can be calculated by $\inf_{(x_\theta, t) \in \operatorname{epi} f} t$.) □

We now define a subclass of convex functions that serve to extend results about single-variate strictly convex functions to the multi-variate case.

Definition 2.2.8 (Strong convexity.)**.** Let $C \subset X$ be a convex set and $f : C \to \mathbb{R}$ be a convex function. We say that f is *strongly convex with parameter* m, or just *strongly convex*, when

$$\nabla^2 f(x) \geq mI$$

for $m > 0$ and any $x \in C$, or, equivalently, if the smallest eigenvalue of $\nabla^2 f$ is uniformly bounded away from the origin for all x. We note that there are extensions of this definition that use only the function's gradient, or even avoid both the gradient and the Hessian, but we do not use them.

2.3 Optimization problems

In this section, we present the structure and notation of a general optimization problem, which includes convex optimization problems, but also two other important problem classes: linear programs and mixed-integer linear programs.

Definition 2.3.1 (Optimization problem)**.** An *optimization problem* is a problem of the form

$$\begin{array}{ll} \min\limits_{x} & f_0(x) \\ \text{s.t.} & f_i(x) \leq b_i, \quad \forall i, \end{array} \tag{2.2 revisited}$$

and the goal is to find an x that achieves the minimum value of $f_0(x)$ among all x such that $f_i(x) \leq b_i$. The quantity x is called the *optimization variable*, the function f_0 is called the *objective function*, and the inequality equations are called the *constraints*. Optimization problem can also include equality equations, or no constraints at all, *i.e.* be *unconstrained*.

An alternative way to write optimization problems is

$$\begin{array}{ll} \min\limits_{x} & f_0(x) \\ \text{s.t.} & x \in X, \end{array} \tag{2.3}$$

We name the set X below.

in which the constraints are implied by the definition of X. For example, in order for problems 2.3 and 2.2 to be the same, we take

$$X = \{x \,:\, f_i(x) \leq b_i,\, \forall i\}.$$

The abstract notation of problem 2.3 is compact and useful for exposition, while the more concrete notation of problem 2.2 can be more useful for analysis (such as when discussing duality in section 2.4), and developing algorithms

(as in section A.1). We will use each notation, or a mixture of both, when appropriate.

A specific x is called a *feasible solution* when $f_0(x) < \infty$ and $x \in X$. An optimization problem is called *feasible* when a feasible solution exists, and *unfeasible* otherwise. The set of feasible solutions is called *feasible set* or *constraint set*, and we often assume the condition $f_0(x) < \infty$ implicitly, as we require it for every problem considered in this text; accordingly, we refer to X as the feasible set.

The finiteness requirement is explained below.

The set X is defined above.

A value $p^\star \in \overline{\mathbb{R}}$ is called an *optimal value* if

$$p^\star = \inf \{ f_0(x) \,:\, x \in X \} .$$

When a problem is unfeasible, we write $p^\star = \infty$. A feasible solution $x^\star \in X$ such that $p^\star = f_0(x^\star)$ is called an *optimal solution*.

This agrees with the convention $\inf \varnothing = \infty$.

Optimizations problems are often classified into families where the optimization variables, the objective function, and the constraints share specific properties. Besides *convex optimization problems*, with convex objective and constraint functions, another important such class are *linear programs*.

Definition 2.3.2 (Linear programs)**.** *Linear programs* (LP) are characterized by a linear objective function and a polyhedral constraint set. A general linear program has the form

$$\begin{array}{ll} \min\limits_{x} & c^\top x \\ \text{s.t.} & Ax = b, \\ & Tx \leq w. \end{array} \tag{2.4}$$

In applications, such as in the simplex algorithm, it may be useful to have a linear program written in *standard form*.

$$\begin{array}{ll} \min\limits_{x} & c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0. \end{array} \tag{2.5}$$

There are families of nonconvex optimization problems for which techniques used in convex—or, more accurately, linear—programming can be used for great effect.

Definition 2.3.3 (Integer programs)**.** *Integer programs* (IP) are linear programs with the additional constraint that the optimization variable must be integer-valued. A general integer program has the form

$$\begin{array}{ll} \min\limits_{z} & c^\top z \\ \text{s.t.} & Az = b, \\ & Tz \leq w, \\ & z \in \mathbb{Z}^{n_z}. \end{array} \tag{2.6}$$

A subclass of integer programs are *binary programs*, or *0-1 integer programs*, where each optimization variable is further constrained to take values in the set $\{0, 1\}$.

Definition 2.3.4 (Mixed-integer programs)**.** *Mixed-integer linear programs* (MILP) are linear programs with both real and integer-valued optimization variables. A general mixed-integer linear program has the form

$$\begin{array}{ll} \min\limits_{x,z} & c^\top x + d^\top z \\ \text{s.t.} & Ax + Bz = b, \\ & Tx + Wz \leq w, \\ & z \in \mathbb{Z}^{n_z}. \end{array} \tag{2.7}$$

There is often a certain degree of freedom when designing an optimization problem that can allow it to be cast as a problem of a particular class. We say that two optimization problems are *equivalent* if it is possible obtain a solution for one problem from a solution to the other, and vice-versa. We do not make this definition formal, but we give two examples.

Example 2.3.5 ("Nonlinear" LP)**.** Let $X \subset \mathbb{R}$ be a polyhedron. Then the problem

$$\begin{array}{ll} \min\limits_{x} & |x| \\ \text{s.t.} & x \in X, \end{array}$$

can be cast as a LP by adding a variable, two constraints, and changing the objective function as follows

$$\begin{array}{ll} \min\limits_{x,t} & t \\ \text{s.t.} & x \in X, \\ & x \leq t, \\ & -x \leq t. \end{array}$$

Remember that $|x| = \max\{x, -x\}$.

This is possible because the epigraph of the absolute value function is a polyhedron. In this light, the pair $(x, t) \in \mathbb{R}^2$ represents a point in the epigraph such that its first-coordinate projection belongs to the feasible set X. For this reason, we refer to the original problem as a LP even if its objective function is not linear.

$x = x^+ - x^-$ with $x^+, x^- \geq 0$.

Example 2.3.6 (Converting LPs into standard form)**.** Any general form linear program 2.4 can be cast into standard form. The conversion procedure is as follows: each variable x becomes a nonnegative pair x^+ and x^-, representing its positive and negative parts, respectively; and each inequality constraint $t^\top x \leq w$ can be written as $t^\top x + s = w$, where $s \geq 0$ is an additional variable called a *slack variable*. The resulting standard form problem resembles the

following

$$\begin{array}{rl} \min\limits_{x^+,x^-,s} & c^\top x^+ - c^\top x^- \\ \text{s.t.} & Ax^+ - Ax^- = b, \\ & Tx^+ - Tx^- + s = w, \\ & x^+ \geq 0, x^- \geq 0, s \geq 0. \end{array} \tag{2.8}$$

2.4 Duality

In this section, we explain what is duality in optimization.

In optimization theory, the main idea behind duality is taking an optimization problem and obtaining from it a new problem, called the *dual problem*. The dual problem is useful for both developing the theory and for constructing algorithms in applications, because, under certain conditions, solving the dual problem is equivalent to solving the original problem. Stronger results are available when the problems under consideration exhibit certain properties, such as convexity or linearity, but weaker results stand in more general conditions.

Consider the following optimization problem.

$$\begin{array}{rll} \min\limits_{x\in\mathbb{R}^n} & f_0(x) & \\ \text{s.t.} & f_i(x) \leq 0, & i \in [1..m], \\ & h_i(x) = 0, & i \in [1..p]. \end{array}$$

At the moment, we make no assumptions over the objective and constraint functions, except that the set

$$\mathcal{D} = \bigcap_{i=1}^{m} \mathbf{dom}\, f_i \cap \bigcap_{i=1}^{p} \mathbf{dom}\, h_i = \{x \in \mathbb{R}^n \,:\, f_i(x) < \infty, h_i(x) < \infty\}\,.$$

is nonempty.

To begin with, we want to relax the problem above by replacing the hard constraints with penalty terms in a new objective function. A simple penalty term is a linear function of how much each constraints is violated, which can be quantified by the difference between the left and right hand sides of each constraint equation. Because the right hand sides are all zero, we are led to the following definition.

Definition 2.4.1. The *Lagrangian* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with the above problem is the function

$$L(x; \lambda, \mu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i h_i(x). \tag{2.9}$$

The variables λ and μ are called *dual variables*, or *Lagrange multipliers*, and we refer to λ_i (resp. μ_i) as the *dual variable associated with the constraint*

$f_i(x) \leq 0$ (resp. $h_i(x) = 0$). The original variable x is called the *primal variable*.

The feasible set of the original problem is

$$\mathcal{F} = \{x \in \mathbb{R}^n \,:\, f_0(x) < \infty, f_i(x) \leq 0, h_i(x) = 0\}\,.$$

Meanwhile the domain of the Lagragian is $\mathcal{D}$; it is clear that $\mathcal{F} \subset \mathcal{D}$.

Definition 2.4.2. The *Lagrange dual function*, or just *dual function*, $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is the infimum of the Lagrangian over x

$$g(\lambda, \mu) = \inf_{x \in \mathcal{D}} L(x; \lambda, \mu). \tag{2.10}$$

Remark 2.4.3. For a fixed $x \in \mathcal{D}$, we can find vectors a_x, b_x, and c_x that allow us to write the Lagrangian as

$$L(x; \lambda, \mu) = \lambda^\top a_x + \mu^\top b_x + c_x.$$

Thus each $L(x;\,\cdot\,,\,\cdot\,)$ is an affine function of the dual variables. As the infimum of affine functions, the dual function is concave. Observe we made no assumptions on the convexity of the original problem. Therefore *the dual function is always concave, even if the original optimization problem is nonconvex.*

The dual function underestimates the primal problem. This property will be shown later (in theorem 2.4.6). Because the dual functions provides many lower bounds to the original problem, we may seek the best such lower bound. This motivates the following definition.

Definition 2.4.4. The *Lagrange dual problem*, or just *dual problem*, is the optimization problem

$$\begin{aligned} \max_{(\lambda,\mu)\in\mathbb{R}^{m\times p}} \quad & g(\lambda, \mu) \\ \text{s.t.} \quad & \lambda \geq 0. \end{aligned} \tag{2.11}$$

For particular problems, it may be possible to formulate the dual problem explicitly. An important case are linear programs.

Example 2.4.5 (Dual linear program)**.** Consider a general LP,

$$\begin{aligned} \min_x \quad & c^\top x \\ \text{s.t.} \quad & Ax = b, \\ & Tx \leq w. \end{aligned}$$

The corresponding Lagrangian is

$$L(x; \lambda, \mu) = c^\top x + \lambda^\top (Tx - w) + \mu^\top (Ax - b).$$

The dual function is

$$g(\lambda, \mu) = -\lambda^\top w - \mu^\top b + \min_x (c^\top + \lambda^\top T + \mu^\top A)x.$$

Observe that for fixed λ and μ, the term being minimized is a linear function of the primal variables, and thus the unconstrained minimum is only finite if the linear function is constant. The only such linear function is the zero function, so

$$g(\lambda,\mu) = \begin{cases} -\lambda^\top w - \mu^\top b, & \text{if } \lambda^\top T + \mu^\top A = -c^\top, \\ -\infty, & \text{otherwise.} \end{cases}$$

Then, the dual problem is

$$\begin{aligned} \max_{\lambda,\mu} \quad & -\lambda^\top w - \mu^\top b \\ \text{s.t.} \quad & \lambda^\top T + \mu^\top A = -c^\top, \\ & \lambda \geq 0. \end{aligned}$$

As a particular explicit case, the Lagragian associated with problem 2.1 is

$$\begin{aligned} L(h,g,df;\lambda,\mu) = cg + pdf + \lambda_1(h-\overline{h}) - \lambda_2 h + \lambda_3(g-\overline{g}) - \lambda_4 g \\ + \lambda_5(df-\overline{df}) - \lambda_6 df + \mu(h+g+df-d). \end{aligned} \quad (2.12)$$

Rearranging, we obtain

$$\begin{aligned} L(h,g,df;\lambda,\mu) = (\lambda_1-\lambda_2+\mu)h + (c+\lambda_3-\lambda_4+\mu)g + (p+\lambda_5-\lambda_6+\mu)df \\ - \mu d - \lambda_1\overline{h} - \lambda_3\overline{g} - \lambda_5\overline{df}. \end{aligned} \quad (2.13)$$

Here, the minimum over h, g, and df can only be finite whenever $\lambda_1 - \lambda_2 + \mu = 0$, $\lambda_3 - \lambda_4 + \mu = -c$, and $\lambda_5 - \lambda_6 + \mu = -p$. We can now calculate the corresponding dual function explicitly.

$$g(\lambda,\mu) = \begin{cases} -\mu d - \lambda_1\overline{h} - \lambda_3\overline{g} - \lambda_5\overline{df}, & \text{if } \lambda_1-\lambda_2+\mu = 0, \lambda_3-\lambda_4+\mu = -c, \\ & \text{and } \lambda_5-\lambda_6+\mu = -p, \\ -\infty, & \text{otherwise.} \end{cases}$$

When writing the dual problem, we can denote the domain of g by adding constraints.

$$\begin{aligned} \max_{\lambda,\mu} \quad & -\mu d - \lambda_1\overline{h} - \lambda_3\overline{g} - \lambda_5\overline{df} \\ \text{s.t.} \quad & \lambda_1 - \lambda_2 + \mu = 0, \\ & \lambda_3 - \lambda_4 + \mu = -c, \\ & \lambda_5 - \lambda_6 + \mu = -p, \\ & \lambda \geq 0. \end{aligned}$$

This problem in particular can be further simplified by noting we can remove the variables λ_2, λ_4, and λ_6 by changing the equalities into inequalities.

$$\begin{aligned} \max_{\lambda,\mu} \quad & -\mu d - \lambda_1 \overline{h} - \lambda_3 \overline{g} - \lambda_5 \overline{df} \\ \text{s.t.} \quad & \lambda_1 + \mu \geq 0, \\ & \lambda_3 + \mu \geq -c, \\ & \lambda_5 + \mu \geq -p, \\ & \lambda \geq 0. \end{aligned}$$

Note that, in this last problem, $\lambda = (\lambda_1, \lambda_3, \lambda_5)$, but we preserved the previous indexes for consistency.

Theorem 2.4.6 (Weak duality)**.** *For all primal-dual pairs of optimization problems, the following inequality holds for all primal variables x and dual variables λ and μ*

$$g(\lambda, \mu) \leq f_0(x). \tag{2.14}$$

In particular, this implies that

$$d^\star = \max_{(\lambda,\mu)\in\mathbb{R}^m_+ \times \mathbb{R}^p} g(\lambda, \mu) \leq \min_{x\in\mathcal{F}} f(x) = p^\star. \tag{2.15}$$

Proof. When x is unfeasible, $p(x) = \infty$ and the inequality holds trivially. Similarly, when $\lambda \geq 0$ does not hold or $g(\lambda, \mu) = -\infty$, the inequality also holds. Now, suppose that x is feasible, *i.e.*,

$$x \in \mathcal{F} = \{x \in \mathbb{R}^n \,:\, f_0(x) < \infty, f_i(x) \leq 0, h_i(x) = 0\},$$

and $\lambda \geq 0$. Then

$$\sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i h_i(x) \leq 0.$$

Adding $f_0(x)$ to both sides of the above inequality, we obtain

$$L(x; \lambda, \mu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i h_i(x) \leq f_0(x).$$

The feasible set $\mathcal{F}$ is a subset of $\mathcal{D}$, therefore

$$g(\lambda, \mu) = \inf_{x'\in\mathcal{D}} L(x'; \lambda, \mu) \leq L(x; \lambda, \mu) \leq f_0(x),$$

showing the desired inequality holds. □

For certain convex optimization problems, a sharper result holds: equality in equation 2.15. This result is called *strong duality*; it is not true in general, but it applies for convex optimization problems whose inequality constraints can be satisfied strictly. This strict feasibility property is called *Slater's condition.*

Theorem 2.4.7 (Strong duality)**.** *For all primal-dual pairs of optimization problems such that the primal problem satisfies* Slater's condition*, i.e., there exists a feasible $x \in \mathcal{F}$ such that*

$$f_i(x) < 0 \quad \forall i \in [1..m],$$

then

$$d^\star = p^\star.$$

Furthermore, whenever $p^\star > -\infty$ (or $d^\star > -\infty$), the dual problem has a solution.

For example: the primal problem has a solution, or the dual problem is feasible.

We refer to subsection 5.3.2 of [10] for a proof of strong duality under Slater's condition. Alternatively, confer theorem 11.39 and corollary 11.40 of [24] (note the conditions $0 \in \operatorname{int} U$ and $0 \in \operatorname{int} V$ plays a similar role to Slater's condition in this context).

Remark 2.4.8. Slater's condition can be weakened if any of the inequalities are affine. Whenever f_i is an affine function, x needs not to satisfy $f_i(x) \leq 0$ strictly for strong duality to hold. In particular, this implies the following result.

Corollary 2.4.9 (Strong duality for LPs)**.** *Strong duality holds for each pair of feasible primal and feasible dual linear programs.*

Strong duality also holds for LPs if the primal is feasible and the dual is not (or vice-versa). This follows directly from weak duality (theorem 2.4.6). Thus, strong duality is only absent for LPs where both primal and dual problems are infeasible.

There are alternatives to Slater's condition under which strong duality holds for convex optimization problems, known as *constraint qualifications*.

Strong duality allows us to characterize the optimal Langrange multipliers of an optimization problem as a measurement of the sensitivity of the problem with respect to its inequalities. To begin with, we parameterize a family of optimization problems with respect to the right-hand side of their constraints and the respective optimal values as follows:

$$p^\star(b, w) = \begin{cases} \min\limits_{x \in \mathbb{R}^n} & f_0(x) \\ \text{s.t.} & f_i(x) \leq b_i, \quad i \in [1..m], \\ & h_i(x) = w_i, \quad i \in [1..p]. \end{cases}$$

The original problem corresponds to setting (b, w) to $(0, 0)$.

Assume that strong duality holds for the original problem and that the dual problem to the original has an optimal solution $(\lambda^\star, \mu^\star)$. Suppose that x is feasible for the pertubed problem given by $(b, w) \neq (0, 0)$, that is,

$$x \in \mathcal{F}(b, w) = \left\{ x \in \mathbb{R}^n \,:\, f_0(x) < \infty, f_i(x) \leq b_i, h_i(x) = w_i \right\}.$$

Strong duality yields

$$p^\star(0,0) = g(\lambda^\star, \mu^\star) \leq L(x; \lambda^\star, \mu^\star) = f_0(x) + \sum_{i=1}^{m} \lambda_i^\star f_i(x) + \sum_{i=1}^{p} \mu_i^\star h_i(x).$$

Now, because $x \in \mathcal{F}(b, w)$ and $\lambda^\star \geq 0$, we have

$$f_0(x) + \sum_{i=1}^{m} \lambda_i^\star f_i(x) + \sum_{i=1}^{p} \mu_i^\star h_i(x) \leq f_0(x) + \sum_{i=1}^{m} \lambda_i^\star b_i + \sum_{i=1}^{p} \mu_i^\star w_i.$$

Combining the previos two inequalities, we obtain

$$p^\star(0,0) - \sum_{i=1}^{m} \lambda_i^\star b_i - \sum_{i=1}^{p} \mu_i^\star w_i \leq f_0(x).$$

Because $x \in \mathcal{F}(b, w)$ is arbitrary, we can finally assert

$$p^\star(0,0) - \sum_{i=1}^{m} \lambda_i^\star b_i - \sum_{i=1}^{p} \mu_i^\star w_i \leq p^\star(b, w). \tag{2.16}$$

A finer interpretation of this result will be possible after definition 2.5.1. At the moment, assume further that $p^\star$ is differentiable at $(b, w) = (0, 0)$. Then we can relate the optimal Lagrange multipliers $(\lambda^\star, \mu^\star)$ to the gradient of $p^\star$ as follows

$$\nabla p^\star(0,0) = (-\lambda^\star, -\mu^\star).$$

This equality can be stablished by evaluating equation 2.16 at, *e.g.*, te_i,

$$-t\lambda_i^\star \leq p^\star(0, \ldots, t, \ldots, 0) - p^\star(0, \ldots, 0, \ldots, 0),$$

dividing by t, and taking the limit when $t \to 0$. We have, in this example, assumed that $t > 0$, e_i is the i-th coordinate basis vector of $\mathbb{R}^m \times \mathbb{R}^p$, and that the nonzero entry of e_i corresponds to $\lambda_i^\star$. Using $t < 0$ yields the other inequality.

2.5 Subdifferentiability

In this section, subdifferentials are presented as a generalization of differentials.

Definition 2.5.1 (Subdifferential)**.** Let $f : C \subset X \to \mathbb{R}$ be a convex function. The *subdifferential of f at x* is the set of continuous linear functions tangent to f at x that globally underestimate f. In other words,

$$\partial_x f = \{\lambda \,:\, f(y) \geq f(x) + \langle \lambda, y - x \rangle \, \forall y \in X\}.$$

An element $\lambda \in \partial_x f$ is called a *subgradient of f at x*. A function f is called *subdifferentiable at x* if $\partial_x f \neq$ is not empty, and it is called *subdifferentiable* if it is subdifferentiable at every $x \in C$.

When f is differentiable at x, the subdifferential at x is the singleton set containing the gradient, *i.e.*, $\partial_x f = \{\nabla f(x)\}$.

An immediate property of subdifferentials is that whenever 0 is a subgradient of f at x, x is a minimizer of f. Taking $\lambda = 0$ in the subdifferential inequality,

$$f(y) \geq f(x) + \langle \lambda, y - x \rangle = f(x)$$

showing our claim holds. This property is analogous to the equivalent property of differentiable convex functions.

Whenever 0 is a gradient of f at x, x is a minimizer of f.

A simple example of a convex function that is subdifferentiable but not differentiable is the following.

Example 2.5.2. The absolute value function $f = |\cdot| : \mathbb{R} \to \mathbb{R}$ is differentiable everywhere except at the origin. Its subgradient at zero is

$$\partial_0 f = [-1, 1].$$

We have seen other examples of subdifferentiable functions before. Equation 2.16 informs us $(-\lambda^\star, -\mu^\star)$ is a subgradient of $p^\star$ at the origin even when $p^\star$ is not differentiable there.

Subgradients can be shown to exist at every point of $\operatorname{int} C$ using the supporting hyperplane theorem. See corollary 8.9 of [24] for a proof of existence of subgradients in a more general case. Alternatively, confer propositions 2.2 (respectively 2.22) of [26] for a proof for the linear (respectively convex) two-stage problem case. (This kind of problem is the topic of Finite horizon.) There are convex functions that are not subdifferentiable at some points, which must therefore be at the boundary of C.

Example 2.5.3. The negative square root function $f : \mathbb{R}_+ \to \mathbb{R}$ given by $f(x) = -\sqrt{x}$ is not subdifferentiable at the origin. Let $\lambda \in \mathbb{R}$. Suppose by contradiction that the subdifferential inequality holds for every $y \in \mathbb{R}$. Then

Note the origin is a boundary point for the domain of f.

$$-\sqrt{y} \geq \lambda y.$$

When $\lambda \neq 0$, we can substitute $y = \frac{1}{4\lambda^2}$, yielding a contradiction. Otherwise, when $\lambda = 0$, any specific value of $y > 0$ yields the desired contradiction.

The existence and boundedness of the subgradient at all points in the domain can be linked to the existence of a convex extension of the function to a larger domain. In fact, it can be related to uniform Lipschitz continuity of the function.

Definition 2.5.4 (Lipschitz continuity.)**.** A function $f : X \to \mathbb{R}$ is called *Lipschitz continuous with constant* L if

$$\|f(x) - f(y)\| \leq L \|x - y\|$$

for every $x, y \in X$.

We do not discuss these relationships in depth, but we show the following lemma that is useful to show convergence of algorithms discussed later.

Lemma 2.5.5 (Uniformly bounded subgradients)**.** *Let $\mathcal{F}$ be a family of bounded convex functions on a compact set X[3]. Let $\delta > 0$ and denote by X_δ the set of points within δ distance of X, that is,*

$$X_\delta = \{x + y \,:\, x \in X, y \in \mathrm{B}_0\}\,.$$

If each function $f \in \mathcal{F}$ can be extended to a convex function $\hat{f} \in L^\infty(X_\delta)$, and, furthermore, all functions f are bounded below by ℓ and all functions $\hat{f}$ are bounded above by u, then the subgradients of the functions in $\mathcal{F}$ are uniformly bounded.

Proof. A subgradient of f at x satisfies

$$f(y) \geq f(x) + \langle \partial_x f, y - x \rangle\,.$$

Note that $f(y) = \infty$ for $y \notin X$.

In particular, this inequality holds for $y = x + \frac{\delta}{2}\frac{\partial_x f}{\|\partial_x f\|} \in X_\delta$ whenever $\partial_x f \neq 0$. Furthermore, because $\hat{f}$ is a convex extension of f, $\partial_x f$ is also a subgradient of $\hat{f}$ at x. Substituting y in the preceding equation and rearranging its terms, we obtain

$$\|\partial_x f\| \leq \frac{2}{\delta}\left[\hat{f}\left(x + \frac{\delta}{2}\frac{\partial_x f}{\|\partial_x f\|}\right) - f(x)\right].$$

Per our assumptions, we can bound the right-hand side of the preceding equation, yielding the desired uniform bound

$$\|\partial_x f\| \leq \frac{2}{\delta}\left[u - \ell\right]. \qquad \square$$

[3]Banach spaces of functions are going to be discussed later: our focus will be on the case $\mathcal{F} \subset L^\infty(X)$.

Chapter 3

Finite horizon

The hydrothermal problem 2.1 can be used to decide upon an operation plan, followed by its execution over a certain time period. Suppose now that the organization has been contracted for two time periods, called the *planning horizon.* While it is possible to solve two different instances of the hydrothermal problem, in the beginning and in the middle of the planning horizon, the second problem instance may be unfeasible. This can happen if, for example, the resources at the end of the first period are insufficient to satisfy the constraints of the second period (we give a pictorial example after introducing some notation). Furthermore, even if it is feasible, there may exist an operation plan that incurs lower operational costs by jointly optimizing the actions for what will happen in both time periods.

Instead of considering two individual problems, we can create a model that encompasses the entire planning horizon. Each time period of interest is called a *stage*, associated with a natural number t; here, there are two stages: 1 and 2. The same variables from before are still present, now indexed by t to represent the stage they are associated with. For example, h_1 and h_2 are the generation target for the hydro power plant over the first and second stage respectively. The hydro power plant has a water reservoir, whose level *at the end of stage* t is r_t, with maximum capacity $\overline{r}_t$. The initial reservoir level at the beginning of the planning horizon is known, and it is denoted by r_0. The weather forecast predicts an inflow i_t that affects the reservoir level. The organization also has the option to spill s_t water to avoid overflow. Additionally, the organization discounts the costs incurred at the second stage by a factor $\beta \in (0, 1]$. Over long enough time periods, this term can arise out of economic considerations, but it can appear in applications for practical mathematical considerations (as we are going to see in lemma 4.1.7; also, cf. subsection 3.2 of [12], which covers a more general class of multistage problems than those discussed in this dissertation).

The resulting model is written in full in (3.1).

$$\begin{aligned} \min_{h,g,df,s,r} \quad & c_1 g_1 + p_1 df_1 + \beta \left(c_2 g_2 + p_2 df_2 \right) \\ \text{s.t.} \quad & h_t + g_t + df_t = d_t, \\ & r_t + h_t + s_t = r_{t-1} + i_t \\ & 0 \leq h_t \leq \overline{h}_t, \\ & 0 \leq g_t \leq \overline{g}_t, \\ & 0 \leq df_t \leq \overline{df}_t, \\ & 0 \leq r_t \leq \overline{r}_t, \\ & 0 \leq s_t. \end{aligned} \tag{3.1}$$

Now that the notation has been introduced, let us sketch an example of how optimizing each stage separately can lead to infeasibility. Let the demand be $d_t = 20$ in both stages, the predicted inflow be $i_1 = 0$, and the upper bounds $\overline{g}_t = 10$ and $\overline{df}_t = 5$. Furthermore, assume the initial reservoir level is $i_0 = 20$. Then the decision to generate $h_1 = 20$ units of energy is feasible for the first stage and incurs zero costs, thus being optimal. However, beginning the second stage with a reservoir level $r_1 = 0$, it becomes impossible to satisfy the demand. Constructing a two-stage problem, it is possible to see that setting generation targets $h_t = 10$ and $g_t = 10$ for both stages is a feasible alternative decision.

We begin section 3.1 by presenting general two-stage problems, their composing parts, and their decomposition into two one-stage problems. Section 3.2 covers stochastic two-stage problems and the notion of recourse. Section 3.3 broadens the scope to deterministic and stochastic multistage problems. Section 3.4 describes cutting-plane methods for the problems presented in this chapter.

3.1 Two-stage problems

In this section, the general two-stage problem and its elements are defined.

Definition 3.1.1 (Two-stage problem)**.** We consider general two-stage problems that can be written in the formulation below.

$$\begin{aligned} \min_{x,u} \quad & c_1(u_1) + \beta c_2(u_2) \\ \text{s.t.} \quad & x_1 = f_1(x_0, u_1), \\ & u_2 \in U_2(x_1) \\ & x_0 \text{ is given}, \\ & u_1 \in U_1(x_0). \end{aligned} \tag{3.2}$$

There is an important distinction between the optimization variables x and u. The variable x is called a *state variable*, and it represents the least amount of information from the first stage required to solve the second stage.

The variable u is called a *control variable*, and it represents the actions and decisions taken over the course of a stage.

The sets U_t are the sets of *feasible controls* that can be chosen given the state x_{t-1} at the beginning of stage t. Observe that, in this representation, U_t are set-valued functions. While this notation is more concise and covers a more general case, very often U_t can be described explicitly by inequalities similar to the set X in 2.3.1.

The functions f_t describe how the control u_t changes the state x_{t-1} into $x_t = f_t(x_{t-1}, u_t)$ over the stage t. For that reason, the functions f_t are called *transition functions*.

Remark 3.1.2. In problem 3.1, there is an one-to-one correspondence between stages and time periods that is monotone in time. Through this text, the notation and language chosen to address and discuss stages suggests such a connection. However, it is not necessary that stages represent time periods at all. A simple example, which we do not develop here, is the problem of finding the shortest path between two nodes on a graph, which can be described as a multistage optimization problem. Observe that *finding* the shortest path and *traversing* it are different actions.

Returning to problem 3.1, we can now identify the elements of a two-stage problem.

- The state variable is the vector of reservoir levels r, while the control variables are the remaining: h, g, df, and s;
- The cost at stage t is given by the expression $c_t g_t + p_t df_t$; [1]
- The transition functions are $f_t(r_{t-1}, h_t, g_t, df_t, s_t) = r_{t-1} + i_t - h_t - s_t$;
- The sets of feasible controls are given by the bounding constraints: $0 \leq h_t \leq \overline{h_1}$, and so on.

However, problem 3.1 has a difference from definition 3.2: it has a constraint on the state variable r: $0 \leq r_t \leq \overline{r_t}$. This is not an issue, because we can, at each stage, add an extra control variable r'_t, and the constraint $r_t = r'_t$. Furthermore, we change the previous contraints on r to contraints on r'_t: $0 \leq r'_t \leq \overline{r_t}$. The modified problem is equivalent to the original problem.

We could have used an alternative to definition 3.2 that allows for state contraints. We chose the current definition because it keeps the notation simple and causes no loss in generality for our purposes. This is because there are more modeling techniques that often allow formulating other problems in this setting,

[1] There is a minor overlap in notation between the use of the symbol c in definition 3.2, as a function, and in problem 3.1, as a constant. While it is customary to denote a linear function and its multiplier by the same symbol, note that (omitting the state index) the cost actually is $(0, c, p, 0)^\top (h, g, df, s)$.

and are thus useful to develop the theory. In practice, they often increase the number of variables and constraints, increasing computational costs, but are sometimes necessary to cast a model in a particular formulation (*e.g.* as a LP). We also note that, even when two problems are equivalent, different formulations can change the constraint qualification. As an example, Slater's condition (see theorem 2.4.6) can fail when replacing an equality constraint (*e.g.* $x = 0$) with two inequality constraints (*e.g.* $x \geq 0$ and $x \leq 0$), since there cannot exist a value satisfying both inequalities strictly simultaneously.

We can decompose a two-stage problem into two one-stage problems as follows. In the beginning of the second stage, assume the decisions for the first stage variables have already been taken. Thus, they are set to constant values, and the second stage variables remain to be decided upon. We can find an optimal operation plan *as a function of the first stage variables*. Furthermore, in our problem setting, the sole connection between the two stages is the dynamical equation given by the transition function

$$x_1 = f_1(x_0, u_1),$$

and the only variable we need knowledge of to solve the second stage problem is the state at the end of the first stage, x_1. Now, before deciding on an operation plan for the first stage, we can substitute the second stage objective for this function of the first stage state variables. The resulting decomposition is written below.

$$\begin{aligned} \min_{x_1, u_1} \quad & c_1(u_1) + \beta Q\left(x_1\right) \\ \text{s.t.} \quad & x_1 = f_1(x_0, u_1), \\ & x_0 \text{ is given}, \\ & u_1 \in U_1(x_0), \end{aligned} \tag{3.3}$$

where

$$Q\left(x_1\right) = \begin{cases} \min\limits_{u_2} & c_2(u_2) \\ \text{s.t.} & u_2 \in U_2(x_1). \end{cases} \tag{3.4}$$

In theory, we need to solve problem 3.4 for each possible value of x_1. In practice, often we only need to solve it at a finite number of possible values, and it is an alternative to solving problem 3.2 exactly for instances of multistage problems. Each formulation has its uses, both when developing the theory, as we shall see, and when solving practical problems (as in the algorithm given in definition 3.4.4).

Definition 3.1.3. The function Q defined in equation 3.4 is called the *cost-to-go function*, or *future cost function*, or *recourse function*. The reason for this last name will become clear after definition 3.2.1. (For maximization problems, the analogous function is called the *value-to-go function*.)

In this context, the function c_1 in 3.3 is called the *immediate cost function* or *instantaneous cost function*.

Let us see how the above decomposition applies to problem 3.1.

$$\begin{aligned} \min_{h_1,g_1,df_1,s_1,r_1} \quad & c_1 g_1 + p_1 df_1 + \beta Q\left(r_1\right) \\ \text{s.t.} \quad & h_1 + g_1 + df_1 = d_1, \\ & r_1 + h_1 + s_1 = r_0 + i_1 \\ & 0 \le h_1 \le \overline{h}_1, \\ & 0 \le g_1 \le \overline{g}_1, \\ & 0 \le df_1 \le \overline{df}_1, \\ & 0 \le r_1 \le \overline{r}_1, \\ & 0 \le s_1, \end{aligned} \tag{3.5}$$

where

$$Q\left(r_1\right) = \begin{cases} \min_{h_2,g_2,df_2,s_2,r_2} & c_2 g_2 + p_2 df_2 \\ \text{s.t.} & h_2 + g_2 + df_2 = d_2, \\ & r_2 + h_2 + s_2 = r_1 + i_2 \\ & 0 \le h_2 \le \overline{h}_2, \\ & 0 \le g_2 \le \overline{g}_2, \\ & 0 \le df_2 \le \overline{df}_2, \\ & 0 \le r_2 \le \overline{r}_2, \\ & 0 \le s_2. \end{cases} \tag{3.6}$$

3.2 Stochastic problems

In this section, the previously deterministic two-stage model is extended to incorporate uncertainties.

In applications, often not all problem data is known ahead of time. For example, in problem 3.1, we explained the constant i_t as the inflow of water into the reservoir expected during stage t. After deciding on an operation plan and beginning execution, the real inflow at a stage may differ from the prediction, so the operation plan is maybe not optimal anymore. Further complications can arise, such as infeasibility: an overestimated inflow in the first stage, leading to an increased expenditure of water, and a reservoir level too low to execute the original plan at the second stage.

The paradigm we discuss to handle these issues is *stochastic optimization*, where uncertainties in the decision-making process are modeled by a *random variable*. This allows us to incorporate a more detailed *uncertainty model* into our *optimization model*. Other approaches to optimization under uncertainty exist, like robust optimization, where the goal is to minimize the worst-case scenario in a set of uncertainties, but we do not discuss them here. (Cf. section 2.4 of [2] for an exposition on a robust linear programming framework.)

Before looking at a formal definition of our new problem setting, let us

naively consider how replacing a constant with a random variable leads to the randomness spreading thorough our model.

- The optimization variables of a stochastic problem can be split into two kinds. *Here-and-now variables* represent decisions and actions taken before a realization of the random variable becomes known; conversely, *wait-and-see variables* are those which can be decided after observing the random variable. They key difference is that wait-and-see variables can depend on said realizations, *i.e.* they are functions of the random variable. Here-and-now variables cannot, and this property is known as *nonanticipativity*.

- The objective function becomes a random function; in particular, its image is a random variable. Unlike real numbers, random variables lack an (interesting[2]) complete order. Instead of minimizing a random function, our goal becomes minimizing the *expected value* of such a function. The justification for this approach is that, incorporating this model in a procedure that will be repeated over time, the decisions taken will minimize the average cost.

  More generally, we may be willing to accept a small increase in the average cost in exchange for reducing the probability of a costly but unlikely event. It is possible to consider *risk measures* instead of the expectation operator (itself a risk measure), that express in the model this aversion to unfortunate events. We limit ourselves to discussing the expectation operator, but we note some risk measures of interest can be represented as the expectation of a modified random variable, in place of the original one. (Cf. chapter 6 of [26] for an introduction to risk averse optimization. Also, subsection 3.3 of [12] briefly explains how this modified random variable can be computed for some particular cases of interest.)

- The constraints become random. We take this to mean that the constraints have to be satisfied *almost surely*, that is, with probability 1. An alternative is requiring that the constraints are satisfied with some probability.

Let us briefly recall some concepts of probability theory and fix our notation. Let Ω denote an arbitrary set, Σ a σ-algebra on Ω, and $P : \Sigma \to [0, 1]$ a probability measure, such that (Ω, Σ, P) is a probability space. A function $\xi : \Omega \to \mathbb{R}$ is a *random variable* if it is measurable, and a value $\xi(\omega)$ is called a *realization*. Often, when there is no ambiguity, we suppress the dependence on $\omega \in \Omega$, and denote both a random variable and its realization by the same symbol, ξ, whenever there is no ambiguity. We denote by Ξ the *support* of the

[2]The Axiom of Choice is equivalent to the Well-ordering Theorem, *i.e.* every set can be well-ordered.

random variable ξ, and denote its elements by $\xi \in \Xi$. Instead of $\xi(\omega) \in \Xi$.

Definition 3.2.1 (Stochastic two-stage problems)**.** We consider general two-stage problems that can be written in the formulation below.

$$\begin{array}{ll} \min\limits_{x,u} & c_1(u_1) + \beta\,\mathbb{E}[c_2(u_2,\xi)] \\ \text{s.t.} & x_1 = f_1(x_0,u_1), \\ & u_2 \in U_2(x_1,\xi) \\ & x_0 \text{ is given}, \\ & u_1 \in U_1(x_0). \end{array} \tag{3.7}$$

The approach we take is to assume a realization of the random variable becomes known either at the beginning or at the end of a stage. For two-stage problems, the first stage decisions are taken before the realization is known; the second stage decisions after. More complicated behaviours are usually more appropriately represented via multistage models. Not in the middle, for example.

Stochastic two-stage problems are also called problems with *recourse*. Because the second stage decision can depend on a particular yet-to-be-observed realization of ξ, it is also called *recourse action*.

Note that unless the random variable ξ has a finite number of realizations, this problem is infinite, in that is contains infinitely many decision variables in the second stage, and correspondingly infinitely many constraints.

Let us look at an example.

Example 3.2.2 (Two-stage stochastic linear problem)**.** Here, the random variable ξ is the second stage data, *i.e.* $\xi = (c_2, T_2, W_2, w_2)$. The first stage data is deterministic and known.

$$\begin{array}{ll} \min\limits_{x,u} & c_1^\top u_1 + \beta\,\mathbb{E}[c_2^\top u_2] \\ \text{s.t.} & x_1 = A_1x_0 + B_1u_1 + b_1, \\ & T_2x_1 + W_2u_2 \le w_2 \\ & x_0 \text{ is given}, \\ & T_1x_0 + W_1u_1 \le w_1. \end{array} \tag{3.8}$$

As in the deterministic setting, we can decompose a stochastic two-stage problem into one-stage problems, as formulated below.

$$\begin{array}{ll} \min\limits_{x_1,u_1} & c_1(u_1) + \beta\,\mathbb{E}[Q(x_1,\xi)] \\ \text{s.t.} & x_1 = f_1(x_0,u_1), \\ & x_0 \text{ is given}, \\ & u_1 \in U_1(x_0), \end{array} \tag{3.9}$$

where

$$Q(x_1,\xi) = \left\{ \begin{array}{ll} \min\limits_{u_2} & c_2(u_2,\xi) \\ \text{s.t.} & u_2 \in U_2(x_1,\xi). \end{array} \right. \tag{3.10}$$

When the random variable ξ is finite, this decomposition results in $|\Xi| < \infty$ deterministic second stage problems, and one deterministic first stage problem. The reason for this is that the expectation in problem 3.9 can be written as a finite sum. Denoting by p_ξ the probability of the realization ξ being observed,[3]

$$\mathbb{E}[Q(x_1, \xi)] = \sum_{\xi \in \Xi} p_\xi Q(x_1, \xi).$$

The function $\mathbb{E}[Q(\,\cdot\,, \xi)]$ is called the *expected cost-to-go function*, or *expected future cost function*, and so on as in definition 3.1.3.

Definition 3.2.3 (Relatively complete recourse)**.** The stochastic two-stage problem 3.7 is said to have *relatively complete recourse* (RCR) if the second stage problem is feasible for every feasible first stage decision and every realization of the random variable. Precisely, let X_1 be a proper subset of $\mathbb{R}^{n_x}$. Then RCR holds if for each $x_1 \in X_1$ and $\xi \in \Xi$, the set

$$\{u \,:\, u \in U_2(x_1, \xi), c_2(u, \xi) < \infty\}$$

is nonempty. Note this implies that the expected cost-to-go function is always finite, that is, $\mathbb{E}[Q(x_1, \xi)] < \infty$ in equation 3.9.

Whenever a particular problem specifies the state variables belong to a particular subset of $\mathbb{R}^{n_x}$, X_1 can be taken as that set. A problem-agnostic definition is taking X_1 as the set

Remember x_0 is given.

$$\{x \,:\, x = f_1(x_0, u), u \in U_1(x_0)\}.$$

This definition only requires the second stage problem be feasible for states that can occur as an outcome of the first stage decisions.

That X_1 is a proper subset of $\mathbb{R}^{n_x}$ is the "relative" in relatively complete recourse. When $X_1 = \mathbb{R}^{n_x}$, this property is called simply complete recourse, but most problems do not have it.

We shall soon discuss multistage problems, and the definition of RCR can be extended to them. However, we do not consider it necessary to present the extension explicitly. Regardless, defining X_1 (or, generally, X_t) explicitly makes it easier to verify in practice, as opposed to the implicit definition as states that could occur as a consequence of previous decisions.

3.3 Multistage problems

In this section, we consider the broader case of multistage problems, initiating with the deterministic case before following into the stochastic case.

[3] To be precise and conform with traditional probability notation, we instead define $p_{\hat{\xi}} = \mathbb{P}[\xi = \hat{\xi}]$, where $\hat{\xi} \in \Xi$ is a realization of ξ, and, correspondingly, $\mathbb{E}[Q(x_1, \xi)] = \sum_{\hat{\xi} \in \Xi} p_{\hat{\xi}} Q(x_1, \hat{\xi})$.

The natural extension of two-stage problems are multistage problems, with an arbitrary finite number of stages, known as *finite horizon problems*. Returning to the hydrothermal example, suppose the organization has been contracted for $T \in \mathbb{N}$ time periods. The stage index t now belongs to the set $[1..T]$, and each stage has their own corresponding decision variables and constants. The complete model can be described as follows.

$$\begin{aligned} \min_{h,g,df,s,r} \quad & \sum_{t=1}^{T} \beta^{t-1} \left(c_t g_t + p_t df_t \right) \\ \text{s.t.} \quad & h_t + g_t + df_t = d_t, \\ & r_t + h_t + s_t = r_{t-1} + i_t \\ & 0 \leq h_t \leq \overline{h}_t, \\ & 0 \leq g_t \leq \overline{g}_t, \\ & 0 \leq df_t \leq \overline{df}_t, \\ & 0 \leq r_t \leq \overline{r}_t, \\ & 0 \leq s_t. \end{aligned} \tag{3.11}$$

Definition 3.3.1 (Multistage problem)**.** We consider general multistage problems that can be written in the formulation below.

$$\begin{aligned} \min_{x,u} \quad & \sum_{t=1}^{T} \beta^{t-1} c_t(u_t) \\ \text{s.t.} \quad & x_t = f_t(x_{t-1}, u_t), \\ & u_t \in U_t(x_{t-1}). \end{aligned} \tag{3.12}$$

A function π_t of x_t such that $\pi_t(x_t) \in U_t(x_t)$, that is, a mapping from the set of states to the set of feasible controls is called a *decision rule*. A sequence of decision-rules $\pi = \{\pi_t\}_t$, one for each stage t, is called a *policy*.

We write the optimization problem 3.11 as T one stage problems with a recursive formulation below, and comment on the boundary conditions afterwards.

$$Q_t\left(r_t\right) = \begin{cases} \min\limits_{\substack{h_{t+1}, g_{t+1}, df_{t+1}, \\ s_{t+1}, r_{t+1}}} & c_{t+1} g_{t+1} + p_{t+1} df_{t+1} + \beta Q_{t+1}\left(r_{t+1}\right) \\ \text{s.t.} \quad & h_{t+1} + g_{t+1} + df_{t+1} = d_{t+1}, \\ & r_{t+1} + h_{t+1} + s_{t+1} = r_t + i_{t+1} \\ & 0 \leq h_{t+1} \leq \overline{h}_{t+1}, \\ & 0 \leq g_{t+1} \leq \overline{g}_{t+1}, \\ & 0 \leq df_{t+1} \leq \overline{df}_{t+1}, \\ & 0 \leq r_{t+1} \leq \overline{r}_{t+1}, \\ & 0 \leq s_{t+1}, \end{cases} \tag{3.13}$$

where $t \in [0..T-1]$.

Two future cost functions are important highlighting: Q_0 and Q_T. The function Q_0 can be interpreted as the future cost function of a stage in the past, preceding the first stage. It includes the immediate cost of the first stage, here $c_1 g_1 + p_1 df_1$, and the (discounted) future cost of the first stage, $Q_1(r_1)$. When evaluated at r_0, $Q_0(r_0)$ is the optimal value of problem 3.11.

Now, Q_T, which appears when we set $t = T - 1$ in the above equation, is instead defined as $Q_T(r_T) = 0$, the constant function everywhere zero. The reasoning behind this definition is that what happens after the planning horizon does not incur neither costs nor profit for the organization. On the other hand, Q_0 is simply the optimal value of the original problem, 3.11, as a function of the initial reservoir level r_0.

There is no loss of generality in assuming Q_T is the zero function. Suppose that for a particular application, it would make sense to set $Q_T = g$, where g is a function. Then the model could be modified to have an additional stage T with cost function $c_T = g$. To be consistent with our formulation, an additional control variable z_T and the constraint $x_T = z_T$ can be added as well, so that c_T can be evaluated at the control variable z_T instead of the state x_T.

Definition 3.3.2 (Stochastic multistage problem)**.** We consider general multistage problems that can be written in the formulation below.

$$\begin{aligned} \min_{x,u} \quad & \mathbb{E}\left[\sum_{t=1}^{T} \beta^{t-1} c_t(u_t, \xi_{t-1})\right] \\ \text{s.t.} \quad & x_t = f_t(x_{t-1}, u_t, \xi_{t-1}), \\ & u_t \in U_t(x_{t-1}, \xi_{t-1}). \end{aligned} \tag{3.14}$$

Compared to the two-stage case in equation 3.7, the random variable then denoted by ξ corresponds to ξ_1. Often in multistage problems, the first stage is deterministic, *i.e.* ξ_0 is assumed to be a known real number (or vector). Whenever this is true, the term $c_1(u_1, \xi_0)$ can be moved out of the expectation. For multistage problems, one can always assume the first stage is deterministic, as it is always possible to add a costless, restrictionless stage at the beginning.

Another important and common assumption is *stagewise independence*: for every stage t, the random variable ξ_t is independent of each other random variable ξ_s, where $s \neq t$. In particular, ξ_t is independent from the history of realizations up to t, $\{\xi_s\}_{s=0}^{t-1}$. (This assumption is particularly useful for the development of algorithms, *e.g* SDDP in [23], which we explain in definition 3.4.5.)

Finally, a set of realizations at each stage $\{\xi_t\}_{t=0}^{T-1}$ is called a *scenario*.

Remark 3.3.3 (Graphical representation). It is useful to represent multistage optimization models graphically. The generic model described in equation 3.12 can be associated with a linear graph.

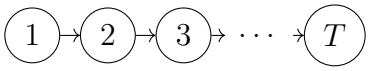

Figure 3.1: Graph representation of deterministic multistage model.

Similarly, the stochastic multistage model described by equation 3.14 can also be represented by a linear graph. The random variable that becomes known at the beginning of a stage is represented by a wavy arrow pointing at the top-left section of the corresponding node, as seen in figure 3.2.

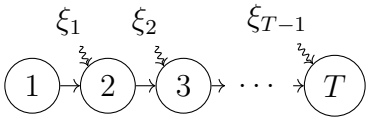

Figure 3.2: Graph representation of stochastic multistage model.

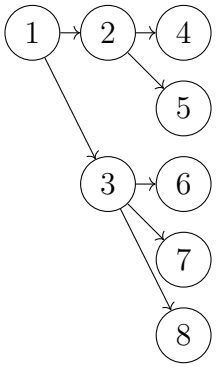

Figure 3.3: Scenario tree

Stochastic multistage models admit another graphical representation, called a *scenario tree*. Figure 3.3 depicts a scenario tree of a particular three-stage problem. The random variable ξ_1 has two possible realizations, represented by the node 1 having two children. The random variable ξ_2 has two or three possible realizations, conditioned on ξ_1. This is represented by the different number of children between nodes 2 and 3. This means stagewise-independency does not hold for the problem represented by this graph.

The scenario tree representation clearly illustrates the stagewise-independency of a model (or lack thereof). However, for large models, it can be cumbersome to depict the entire tree. In addition to its smaller size, the linear graph does have a benefit over the scenario tree in the stagewise-independent case: it merges together nodes with the same cost-to-go function. Solution algorithms often calculate (exactly or approximately) the cost-to-go functions of each stage. If two nodes share the same cost-to-go function, only a single representation (or approximation) of that cost-to-go function is necessary. As

an example of the contrary, the three-stage problem in figure 3.3 would require different cost-to-go functions at nodes 2 and 3. In this sense, the linear graph can be regarded as a condensed representation of the scenario tree. See [12] for a discussion of more general multistage problems, including problems associated with nonlinear graphs.

3.4 Algorithms

In this section, we describe methods for solving finite horizon optimization problems in the following order: stochastic two-stage problems, deterministic multistage problems, then stochastic multistage problems.

We begin discussing a cutting-plane method for solving two-stage stochastic linear optimization problems with finite support, originally proposed by Van Slyke and Wets (cf. [29]). This particular case of problem 3.7 is formulated below. Note that we do not distinguish the state and control variables from each other, as the distinction is not necessary for the method we are about to present. The inequality constraints are assumed to be nonnegativity constraints, in a similar manner to standard form LPs.

Recall example 2.3.6 of LP conversion to standard form.

$$\begin{array}{ll} \min\limits_{x} & c_1^\top x_1 + \sum\limits_{\xi\in\Xi} p_\xi c_{2,\xi}^\top x_{2,\xi} \\ \text{s.t.} & Ax_1 = b, \\ & T_\xi x_1 + W_\xi x_{2,\xi} = w_\xi, \\ & x_1 \geq 0, x_{2,\xi} \geq 0. \end{array} \tag{3.15}$$

Definition 3.4.1 (L-shaped method)**.** The idea behind the *L-shaped method* is to construct a piecewise linear function that underestimates the first-stage future cost function. This corresponds to an outer polyhedral approximation of the future cost function's epigraph. The linear inequalities in the representation of the piecewise linear underestimator are called *cuts*. For simplicity, we assume relative complete recourse holds, and discuss the implications afterwards.

At the beginning of an iteration of the L-shaped method, let $\mathcal{C}$ denote the cuts available. The problem to be solved is

$$\begin{array}{ll} \min\limits_{x_1,\theta} & c_1^\top x_1 + \theta \\ \text{s.t.} & Ax_1 = b, \\ & \theta \geq \langle a^i, x_1\rangle + b^i, \quad \text{for } (a^i, b^i) \in \mathcal{C}, \\ & x_1 \geq 0. \end{array} \tag{$P_1(\mathcal{C})$}$$

The pair (x_1, θ) can be thought of as a point in the epigraph of the piecewise linear approximation of the cost-to-go function, described by the constraints

$$\theta \geq \left\langle a^i, x_1 \right\rangle + b^i.$$

After an optimal solution $(x_1^\star, \theta^\star)$ of $P_1(\mathcal{C})$ is obtained, coefficients a and b for a new cut can be generated by solving each of the following $|\Xi|$ problems

$$\begin{array}{ll} \min\limits_{x_{2,\xi}} & c_{2,\xi}^\top x_{2,\xi} \\ \text{s.t.} & W_\xi x_{2,\xi} = w_\xi - T_\xi x_1^\star, \\ & x_{2,\xi} \geq 0, \end{array} \qquad (P_{2,\xi}(\mathcal{C}))$$

We suppress the dependency of the optimal solution on $\mathcal{C}$ for simplicity of notation. Problem $P_{2,\xi}(\mathcal{C})$ depends on $\mathcal{C}$ only through $x_1^\star$.

calculating the optimal Lagrange multiplier $\lambda_\xi^\star$ of the equality constraint, and then computing

$$a = -\sum_{\xi \in \Xi} p_\xi \left(\lambda_\xi^\star\right)^\top T_\xi \quad \text{and} \quad b = \sum_{\xi \in \Xi} p_\xi \left(\lambda_\xi^\star\right)^\top w_\xi.$$

Indeed, from LP duality, we know that

$$Q(x_1^\star, \xi) = \left(\lambda_\xi^\star\right)^\top (w_\xi - T_\xi x_1^\star).$$

Using the chain rule, a subgradient of Q at $x_1^\star$ is given by $\left(\lambda_\xi^\star\right)^\top T_\xi$. So, applying the subgradient inequality for Q yields

$$Q(x, \xi) \geq \left(\lambda_\xi^\star\right)^\top (w_\xi - T_\xi x) = \left(\lambda_\xi^\star\right)^\top w_\xi - \left(\lambda_\xi^\star\right)^\top T_\xi x.$$

Therefore, taking the expectation over ξ on both sides results in

$$\mathbb{E}\left[Q(x, \xi)\right] \geq \langle a, x \rangle + b,$$

so we see that indeed the pair (a, b) provides the coefficients of a linear underestimator of the expected cost-to-go function.

As a stopping condition, if $\theta^\star \geq a^\top x_1^\star + b$, the pair $(x_1^\star, x_2^\star)$ is an optimal solution of the original two-stage problem. Otherwise, the cut (a, b) can be added to $\mathcal{C}$, and the algorithm proceeds to the next iteration.

$x_2^\star$ denotes the function $\xi \mapsto x_{2,\xi}^\star$.

Algorithm 3.1: L-shaped method.

Data: an initial lower bound $\mathcal{C} = \{(a^0, b^0)\}$.
Result: an optimal solution.

```
while optimal solution not found do
    solve P_1(C)
    for ξ ∈ Ξ do
        solve P_{2,ξ}(C)
    end
    calculate new cut (a, b) and add it to C
end
```

Relative complete recourse guarantees that each problem $P_{2,\xi}(\mathcal{C})$ generated during the execution of the method is feasible. Otherwise, it becomes

necessary to generate and store *feasibility cuts*, which are linear constraints that approximate the domain of the expected cost-to-go function. At each iteration, either a feasibility or an optimality cut is generated, or the algorithm terminates. Generating a feasibility cut requires solving $|\Xi|$ linear problems, similarly to optimality cuts, and so we direct the reader to section 5.1 of [8] for a more complete description of the L-shaped method.

Optimality cuts are the cuts previously described.

The cutting-plane technique employed in the L-shaped method is also used for the algorithms we present for solving multistage problem. Another key idea required for their exposition is the Dynamic Programming algorithm. The use of the DP algorithm in optimization was popularized by Bellman in the early fifties. It is based on what Bellman called the *principle of optimality*, which can be informally stated as follows.

Definition 3.4.2 (Principle of optimality)**.** All optimal policies of a multistage optimization problem have the property that, beginning from any given stage, the remaining decision-rules compose an optimal policy for the multistage problem that begins at that stage.

In other words, given an optimal policy $\pi^\star = \{\pi_t^\star\}_{t=1}^T$ for a T-stage problem, the policy $\{\pi_t^\star\}_{t=s}^T$ is optimal for the $(T-s+1)$-stage subproblem composed of stage s and all future stages.

There is a straightforward, intuitive reasoning supporting the principle of optimality. Suppose A, B, and C are cities, and there is a travel route of minimal length from A to C passing through B. Then the route from B to C must be of minimal length, otherwise the original route from A to C could be improved by replacing the original B to C portion with a shorter one.

Definition 3.4.3 (DP algorithm)**.** The procedure of the DP algorithm has been hinted at by the recursive description of multistage optimization problems. This algorithm was originally developed for multistage problems with discrete state and control variables. We further assume the state, control, and random variables can only take a finite amount of values, *e.g.* the state variable belongs to a set $X \subset \mathbb{R}^{n_x}$ such that $|X| < \infty$. In this case, the cost-to-go functions $\{Q_t\}_{t=0}^T$ can be represented by a vector in $\mathbb{R}^{|X|}$, and they can be calculated exactly.

We give an outline of the proof by induction below, and direct the reader to sections 1.3 and 1.5 of [4]. The latter section contains the more rigorous mathematical proof, but the more technical issues are much simpler under the present assumptions. We also revisit a similar algorithm later when discussing the value iteration algorithm 4.1 in the infinite horizon setting, which, in contrast to the DP algorithm, is not guaranteed to converge in a finite number of iterations.

First, we can calculate Q_T at each possible state as follows:

$$Q_T(x_{T-1}) = \begin{cases} \min\limits_{u_T} & \mathbb{E}\left[c_T(u_T, \xi_{T-1})\right] \\ \text{s.t.} & x_T = f_T(x_{T-1}, u_T, \xi_{T-1}), \\ & u_T \in U_T(x_{T-1}, \xi_{T-1}). \end{cases} \tag{3.16}$$

Note the expectation is a finite sum, and, for example, all (finite) values of u_T can be checked for optimality. Then, assuming Q_{t+1} has been calculated, we can solve

$$Q_t(x_{t-1}) = \begin{cases} \min\limits_{u_t} & \mathbb{E}\left[c_t(u_t, \xi_{t-1}) + Q_{t+1}(x_t)\right] \\ \text{s.t.} & x_t = f_t(x_{t-1}, u_t, \xi_{t-1}), \\ & u_t \in U_T(x_{t-1}, \xi_{t-1}). \end{cases} \tag{3.17}$$

Algorithm 3.2: DP algorithm.

Data: the problem data required to build each cost-to-go function.
Result: the function Q_0.

```
for t ∈ [T..0] do        // Note we begin at T and descend to 0.
    for i ∈ [1..|X|] do
        solve Q_t(x_i)
    end
end
```

Multiple approaches for solving continuous state and continuous control problems have been developed. We now present the *dual* DP (DDP) algorithm, and later the *stochastic* DDP (SDDP) algorithm. The SDDP algorithm was introduced by Pereira and Pinto to solve multistage problems related to hydro power planning (cf. [23]). It can be compared to Benders decomposition and the L-shaped method in a multistage setting.

Definition 3.4.4 (DDP algorithm)**.** A basic description of the algorithm is the following. Each stage is represented by a one stage problem, where the cost-to-go term is approximated by the maximum of a set of affine functions, known as *cuts*. A sequence of feasible controls for each stage and a corresponding sequence of states are generated by solving the one stage problems, beginning with the first stage and proceding *forward* in time. Now, duality arguments allow the calculation of new cuts that improve the cost-to-go approximations. The key observation is that by calculating and updating each one stage problem by moving *backwards* in time, the new t-th stage cut will be generated after the $(t+1)$-st approximation has been improved, resulting in a better cut.

Now, let us go over the DDP algorithm in depth. First, at the k-th step of the algorithm, the one stage problem of stage t resembles the following problem,

$$\mathfrak{Q}_t^k(x_t) = \begin{cases} \min\limits_{x_{t+1},u_{t+1},\theta_{t+1}} & c_{t+1}(u_{t+1}) + \theta_{t+1} \\ \text{s.t.} & x_{t+1} = f_{t+1}(x_t, u_{t+1}), \\ & u_{t+1} \in U_{t+1}(x_t), \\ & \theta_{t+1} \geq \left\langle a_{t+1}^i, x_{t+1} \right\rangle + b_{t+1}^i, \quad \forall i \in [1..k], \end{cases} \tag{3.18}$$

where θ_t is an extra one-dimensional (control) variable that represents the cost-to-go term, which as mentioned is approximated by the maximum of the affine functions $\left\langle a_{t+1}^i, \cdot \right\rangle + b_{t+1}^i$. The symbol $\mathfrak{Q}$ denotes the letter "Q" in the Fraktur script, and is chosen to represent the above problem as it is an outer polyhedral approximation of the true cost-to-go function Q_t. The algorithm will have accumulated k additional constraints at each stage in the k-th step, incurring increasing computational costs with each iteration. A very large amount of cuts can also lead to numerical instability. An option is to perform *cut selection* procedures, which remove unnecessary cuts from the subproblems (we refer the reader to references listed after the end of this definition). The above problem is initialized with a precalculated lower bound for θ_t, which can be obtained easily in many applications. For example, if all stage costs are nonnegative, the lower bound $\theta_t \geq 0$ can be used.

Second, with the above representation fixed, we begin from $t = 0$ and solve each of the T one stage problems, up to $t = T - 1$. Remember that x_0 is known and fixed. This calculation returns to us a sequence of states $\left\{x_t^k\right\}_{t=0}^{T-1}$ and a sequence of present costs $\left\{c_{t+1}(u_{t+1}^k)\right\}_{t=0}^{T-1}$. The reason for returning a sequence of costs will become clear later. See algorithm 3.3.

Algorithm 3.3: Forward pass (DDP).

Data: an initial state x_0 and a list of one-stage problems $\left(\mathfrak{Q}_t^k\right)_{t=0}^{T-1}$.
Result: a trajectory and the corresponding costs.

```
set state ← x_0
set trajectory ← [x_0]
set cost ← [ ]
for t ∈ [0..T − 1] do
    solve 𝔔_t^k(state)
    append c_{t+1}(u_{t+1}) to cost
    update state ← x_{t+1}
    append state to trajectory
end
```

Third, recall that we wish to approximate the cost-to-go term Q_{t+1} at stage t from below. If we had both the optimal value $Q_{t+1}(x_{t+1})$ at a state x_{t+1}

and a subgradient λ of Q_{t+1} at x_{t+1}, then, because Q_{t+1} is convex, we could underestimate the cost-to-go at x by

$$Q_{t+1}(x) \geq Q_{t+1}(x_{t+1}) + \langle \lambda, x - x_{t+1} \rangle = (Q_{t+1}(x_{t+1}) - \langle \lambda, x_{t+1} \rangle) + \langle \lambda, x \rangle .$$

This suggests we take $a = \lambda$ and $b = (Q_{t+1}(x_{t+1}) - \langle \lambda, x_{t+1} \rangle)$ as cut coefficients. Although we do not have access to Q_{t+1}, we do have access to $\mathfrak{Q}_{t+1}^k$, a global convex underestimator of Q_{t+1}. Thus any function below $\mathfrak{Q}_{t+1}^k$ is also below Q_{t+1}. The simplest approach to compute a subgradient of $\mathfrak{Q}_{t+1}^k$ is to modify subproblem 3.18 by adding a dummy variable and a constraint for each state variable,

$$\mathfrak{Q}_t^k(x_t) = \begin{cases} \min\limits_{x_{t+1}, u_{t+1}, \theta_{t+1}, \hat{x}_t} & c_{t+1}(u_{t+1}) + \theta_{t+1} \\ \text{s.t.} & x_{t+1} = f_{t+1}(\hat{x}_t, u_{t+1}), \\ & u_{t+1} \in U_{t+1}(\hat{x}_t), \\ & \theta_{t+1} \geq \left\langle a_{t+1}^i, x_{t+1} \right\rangle + b_{t+1}^i, \\ & \hat{x}_t = x_t. \end{cases} \tag{3.19}$$

The dual variables λ_t^k associated with the equality constraints $\hat{x}_t = x_t$ are the desired subgradients. The final expression for the new cut coefficients are $a_t^{k+1} = \lambda_t^k$ and $b_t^{k+1} = \left(\mathfrak{Q}_t^k(x_t) - \left\langle \lambda_t^k, x_t \right\rangle\right)$. The approach used here to calculate the subgradient is also used in [14] and in [12]. Other approaches are related to the dual of the transition constraint (cf. section 3 of [23]; or the term β_t^k in definition 3.1 of [20]). See algorithm 3.4.

Algorithm 3.4: Backward pass (DDP).

Data: a trajectory and a list of one-stage problems $\left(\mathfrak{Q}_t^k\right)_{t=0}^{T-1}$.

Result: an updated set of one stage problems $\left\{\mathfrak{Q}_t^{k+1}\right\}_{t=0}^{T-1}$.

for $t \in [T-2..0]$ **do** `// Remember the terminal cost-to-go term is everywhere zero.`
  set state $\leftarrow x_{t+1}$
  solve $\mathfrak{Q}_{t+1}^k$(state)
  set $a_{t+1}^{k+1} \leftarrow$ the value of the optimal fishing dual
  set $b_{t+1}^{k+1} \leftarrow \left(\mathfrak{Q}_{t+1}^k(\text{state}) - \left\langle a_{t+1}^{k+1}, \text{state} \right\rangle\right)$
  update $\mathfrak{Q}_t^k$ to $\mathfrak{Q}_t^{k+1}$ by adding the cut $\theta_{t+1} \geq \left\langle a_{t+1}^{k+1}, x_{t+1} \right\rangle + b_{t+1}^{k+1}$
end

Finally, we need to define a condition for terminating the algorithm. At stage k, the optimal value $\mathfrak{Q}_0^k(x_0)$ is an underestimator of the optimal value $Q_0(x_0)$, in other words,

$$\mathfrak{Q}_0^k(x_0) \leq Q_0(x_0).$$

Furthermore, remember that, in the forward pass, we have a sequence of present costs $\left\{c_{t+1}(u_{t+1}^k)\right\}_{t=0}^{T-1}$ available. Because each present cost function

is known exactly, the sum of the costs over an arbitrary feasible trajectory overestimates $Q_0(x_0)$, that is

$$Q_0(x_0) \leq \sum_{t=0}^{T-1} c_{t+1}(u_{t+1}^k).$$

The previous two inequalities hold for every k, and so we can choose indexes independently to make the bounds as tight as possible. Let us denote the best lower and upper bounds found up to stage $k+1$ by $\underline{q}_{k+1} = \max\left\{\mathfrak{Q}_0^k(x_0), \underline{q}_k\right\}$ and $\overline{q}_{k+1} = \min\left\{\sum_{t=0}^{T-1} c_{t+1}(u_{t+1}^k), \overline{q}_k\right\}$ respectively. Prior to executing the DDP algorithm, we can choose an $\varepsilon > 0$, and terminate the algorithm after step k if

$$\overline{q}_k - \underline{q}_k \leq \varepsilon.$$

In-depth convergence analyses of the DDP algorithm, its variations, and other algorithms with the same general structure exist in the literature. When the multistage problem is linear, that is, the present costs c_t and transition functions f_t are linear and the feasible sets U_t are polyhedral, it can be shown by induction the functions Q_t are piecewise linear. In this case, the respective dual problems will also be linear problems, and thus Q_t can be calculated exactly in a finite number of steps (that is, for some large K, $Q_t = \mathfrak{Q}_t^K$). However, convergence only requires that $\mathfrak{Q}_t^k$ is equal to Q_t on a vicinity of an optimal trajectory. For convex multistage problems, it is this local convergence that is guaranteed, *i.e.* the approximation may not be close to the true cost-to-go away from the minimum.

Finally, we remark that proofs of convergence of the DDP algorithm are often readily extended to the SDDP algorithm we soon explain in definition 3.4.5. Rather, it may be more accurate to say the convergence of the DDP algorithm is considered a particular case of SDDP as applied to deterministic multistage problems, and it is useful to consider it separately as an expository tool. As such, we condense external references about both at the end of this subsection, after the definition of the SDDP algorithm, to avoid repetition.

Definition 3.4.5 (SDDP algorithm)**.** There are two modifications to be done to the DDP algorithm to adapt it to the stochastic setting. First, the *forward pass* is adapted to a *Monte Carlo method*, where a possible scenario is sampled, and optimization at each stage is performed conditioned on the respective sample realization. Second, during the *backward pass*, the calculation of a new cut at a stage requires *solving multiple subproblems*, corresponding to each possible random realization at the next stage. Furthermore, we assume the stochastic multistage problem has stagewise independent random variables with finite support.

The main modification to the forward pass is sampling and storing realizations of the random variable. See algorithm 3.5.

Algorithm 3.5: Forward pass (SDDP).

Data: an initial state x_0, an initial realization ξ_0, and a list of one-stage problems $\left(\left\{\mathfrak{Q}^k_{t,\xi_t}\right\}_{\xi_t\in\Xi_t}\right)_{t=0}^{T-1}$.

Result: a trajectory and the corresponding costs.

```
set state ← x_0
set sample ← ξ_0
set trajectory ← [(x_0, ξ_0)]
set cost ← [ ]
for t ∈ [0..T − 1] do
    solve Q^k_{t, sample}(state)
    append c_{t+1}(u_{t+1}, sample) to cost
    update state ← x_{t+1}
    update sample ← ξ_{t+1}
    append (state, sample) to trajectory
end
```

During the backwards pass, it becomes necessary to solve each subproblem $\left\{\mathfrak{Q}^k_{t+1,\xi_t}\right\}_{\xi_t\in\Xi_t}$ in order to calculate a new cut for the *expected* cost-to-go function of $\mathfrak{Q}^k_{t,\text{ sample}}$. However, due to the stagewise independency assumption, we can share cuts across $\left\{\mathfrak{Q}^k_{t,\xi_{t-1}}\right\}_{\xi_{t-1}\in\Xi_{t-1}}$, as their expected cost-to-go function is the exact same. See algorithm 3.6.

The reader interested in a proof of convergence for convex multistage problems can confer [14], which covers (S)DDP and similar methods. As another option, [16] treats both the linear and convex deterministic cases, with and without cut selection strategies. Another work by the same author, [15], considers SDDP and related algorithms but in a risk-averse context. In [11], the computational efficiency of SDDP under different cut selection strategies is compared numerically (the model resembles the example in subsection 4.3.1, a modification of the hydrothermal model presented in equation 3.11). Section 4 of [12] features a brief overview of existing methods and convergence proofs, and notes that earlier proofs (such as [20]) did not state an assumption regarding the application of the second Borel-Cantelli lemma[4]. An earlier method related to SDDP is the Nested Decomposition method introduced in [7].

[4]Informally, the lemma states that if the sum of probabilities of countably many events is finite, the probability that infinitely many of them happen is 0.

Algorithm 3.6: Backward pass (SDDP).

Data: a trajectory and a list of one stage problems $\left(\left\{\mathfrak{Q}^k_{t,\xi_t}\right\}_{\xi_t\in\Xi_t}\right)_{t=0}^{T-1}$.

Result: an updated list of one-stage problems $\left(\left\{\mathfrak{Q}^{k+1}_{t,\xi_t}\right\}_{\xi_t\in\Xi_t}\right)_{t=0}^{T-1}$.

for $t \in [T-2..0]$ **do** `// Remember the terminal cost-to-go term is everywhere zero.`
  set state $\leftarrow x_{t+1}$
  set sample $\leftarrow \xi_{t-1}$
  for $\xi_t \in \Xi_t$ **do** `// Remember` ξ_t `becomes known at the beginning of the` $(t+1)$`-st stage.`
    solve $\mathfrak{Q}^k_{t+1,\xi_t}$(state)
  end
  set $a^{k+1}_{t+1} \leftarrow$ the expected value of the optimal fishing dual
  set $b^{k+1}_{t+1} \leftarrow \left(\mathbb{E}\left[\mathfrak{Q}^k_{t+1,\xi_t}(\text{state})\right] - \left\langle a^{k+1}_{t+1}, \text{state}\right\rangle\right)$
  for $\xi_{t-1} \in \Xi_{t-1}$ **do**
    update $\mathfrak{Q}^k_{t,\xi_{t-1}}$ to $\mathfrak{Q}^{k+1}_{t,\xi_{t-1}}$ by adding the cut
    $\theta_{t+1} \geq \left\langle a^{k+1}_{t+1}, x_{t+1}\right\rangle + b^{k+1}_{t+1}$
  end
end

Chapter 4

Infinite horizon

A natural question that follows finite horizon optimization problems is if it is possible to optimize over an infinite number of time periods. Recall the finite horizon hydrothermal problem 3.11. It's possible that the organization contractually responsible for maintaining a steady supply of energy doesn't intend to perform its activities for a specific amount of periods, but rather an indefinite amount, and it periodically solves sufficiently large finite horizon problems to update its operation plan. One alternative is to formulate an infinite horizon model, as follows

$$\begin{aligned} \min_{h,g,df,s,r} \quad & \sum_{t=1}^{\infty} \beta^{t-1} \left(c_t g_t + p_t df_t \right) \\ \text{s.t.} \quad & h_t + g_t + df_t = d_t, \\ & r_t + h_t + s_t = r_{t-1} + i_t \\ & 0 \leq h_t \leq \overline{h}_t, \\ & 0 \leq g_t \leq \overline{g}_t, \\ & 0 \leq df_t \leq \overline{df}_t, \\ & 0 \leq r_t \leq \overline{r}_t, \\ & 0 \leq s_t. \end{aligned} \tag{4.1}$$

Despite the seemingly minor changes when compared to the finite horizon model, note that our optimization variables are now infinite dimensional vectors, that is, they belong to some Banach space of sequences of real numbers, $\mathbb{R}^{\mathbb{N}}$, equipped with some metric.

Section 4.1 defines infinite horizon problems and presents the assumptions that restrict our problem class of interest. The section also presents the Wald-Bellman operator and variants, that are then used to show useful properties and characterizations of optimal solutions. The results obtained allow the development of the solution methods featured in section 4.2: subsection 4.2.1 for the simpler discrete and finite case, and subsection 4.2.2 returning to continuous state and control problems. Section 4.3 showcases several examples: subsection 4.3.1 compares the finite horizon and infinite horizon hydrothermal

problems as a model of the Brazilian interconnected power system, and subsection 4.3.2 features examples motivated by the investigation of properties of infinite horizon models. Section 4.4 ends by proposing improvements to the examples and topics of investigation motivated by them.

4.1 Infinite horizon problems

In this section, infinite horizon problems are defined and analyzed mainly through the application of the Wald-Bellman operator (or modified versions).

Definition 4.1.1 (Infinite horizon problem)**.** We consider infinite horizon problems that can be written in the formulation below, and satisfies some assumptions to be discussed later.

$$\begin{array}{ll} \min\limits_{x,u} & \sum\limits_{t=1}^{\infty} \beta^{t-1} c_t(u_t) \\ \text{s.t.} & x_{t+1} = f_t(x_t, u_t), \\ & u_t \in U_t(x_t). \end{array} \tag{4.2}$$

There are two ways of making the preceding formulation precise. The first is to interpret the infinite series in the objective function as a limit of finite sums. Note we then have an optimization problem with an infinite number of variables and of constraints. The second is through theorem 4.1.8. At the moment, it can be thought of intuitively as the limit of T-stage problems as T tends to infinity. Both are used as definition in the literature, and are known to be equivalent under certain hypotheses. (See theorem 4.1.14.)

We will restrict ourselves to a class of infinite horizon problems that we will call *stationary*.

Definition 4.1.2 (Periodicity and Stationarity)**.** An infinite horizon optimization problem of the form 4.2 is said to be *periodic with period* p if

$$c_t = c_{t+p}, \quad f_t = f_{t+p}, \quad \text{and} \quad U_t = U_{t+p} \qquad \text{for every } t \in \mathbb{N},$$

and also if $p \in \mathbb{N}$ is the smallest number such that these three equalities hold (*i.e.* p is the *least* period). Observe that the stage costs c_t, the transition functions f_t, and the admissible controls U_t comprise the problem data, inherent to each particular situation, while the state variables x_t and the control variables u_t remain to be chosen according to some optimizing procedure.

When a periodic problem has period 1, we say that it is *stationary*. Stationarity allows us to drop the index t from the stage cost, the transition functions, and the admissible control sets.

Periodic infinite horizon problems can be reformulated as stationary problems. Intuitively, instead of considering p separate stages, each representing a time period with a certain length, one can consider a larger stage representing

p time periods in a bundle. For a p-periodic problem, the infinite sum in problem 4.2 can be rewritten as

$$\sum_{t=1}^{\infty} \beta^{t-1} c_t(u_t) = \sum_{t=1}^{\infty} \beta^{(t-1)p} \sum_{i=1}^{p} \beta^{i-1} c_{p(t-1)+i}(u_{p(t-1)+i}) = \sum_{t=1}^{\infty} \beta'^{t-1} c'(u').$$

Here, $\beta' = \beta^p$, and c' and u' can be thought of as Cartesian products of p functions and p variables, respectively (although the costs need to incorporate the weights β to be summed). Similarly, the state variable is replaced by x', the transition function by f' and the admissible controls by U'. The preceding argument is informal, but hopefully sufficient.

Remark 4.1.3 (Graphical representation). As with finite horizon models, it is useful to represent infinite horizon models as a graph. However, the linear graph in this setting has an infinite number of nodes, as shown in figure 4.1. For periodic models, it becomes possible to represent them with a cyclical graph, as in figure 4.2.

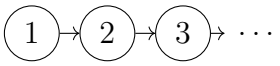

Figure 4.1: Linear graph representation of infinite horizon model.

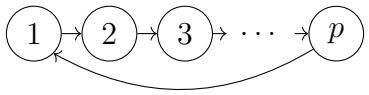

Figure 4.2: Cyclical graph representation of infinite horizon model.

Remark 4.1.4 (Aperiodicity). It is possible to convert aperiodic infinite horizon problems into periodic problems by introducing a state variable representing time. Denote by $\hat{x}_t$ this additional variable at period t. This variable has an initial condition $\hat{x}_1 = 1$, and a dynamic constraint $\hat{x}_{t+1} = \hat{x}_t + 1$. The data of the problem is modified to depend on this new variable rather than on the time index, *e.g.* the cost at stage t is reformulated from $c_t(u_t)$ to $c(u_t, \hat{x}_t) = c_{\hat{x}_t}(u_t)$.

However, this reformulation conflicts with our approach as it requires the introduction of a *discrete* state variable, and our focus is on continuous state and continuous control variables.

To maintain the previous structure, one can introduce a new control variable $\hat{u}$ and a constraint $\hat{u}_t = \hat{x}_t$.

In this setting, it is possible to introduce a functional operator that is of great theoretical use, as well as an useful tool for developing algorithms.

Definition 4.1.5. Given a particular instance of problem 4.2, the associated *Wald-Bellman operator*, or simply Bellman operator, is the mapping B such

that

$$B(V)(x_0) = \begin{cases} \min\limits_{x,u} & c\left(u\right) + \beta V\left(x\right) \\ \text{s.t.} & x = f(x_0, u), \\ & u \in U\left(x_0\right), \end{cases} \tag{4.3}$$

defined on an appropriate function space (to be discussed later). Alternatively, one can succintly write

$$B(V)(x_0) = \min_{u \in U(x_0)} c(u) + \beta V(f(x_0, u)).$$

Note that the index t has been dropped from almost all terms, except for the initial state x_0. This is possible because the minimization problem is a single stage problem; in this sense, the expression $\beta V(x)$ can be seen as a cost for terminating at the state x.

Remark 4.1.6 (Domain of the Wald-Bellman operator). It is necessary to set the stage in order to specify the function space we consider the Wald-Bellman operator is defined on. The main motivation behind the following asumptions and subsequent definition is a theorem yet to be discussed (theorem 4.1.8).

Implicit in the definition of the infinite horizon problem are the sets where the state and control variables take values in. Remember the set-valued function $U(x)$ represents the feasible controls given a state x. Denote by $X \subset \mathbb{R}^{n_x}$ the domain of U, by $\mathcal{U} = \bigcup\limits_{x \in X} U(x) \subset \mathbb{R}^{n_u}$ the image of X by U, and the domain of the transition function f is

The dimensions n_x and n_u are nonnegative integers.

$$\operatorname{graph} U \coloneqq \{(x, u) \,:\, x \in X, u \in U(x)\},$$

which is a subset of $X \times \mathcal{U}$.

The important assumptions to be considered relate to *convexity*, *continuity*, and *compactness*. We assume the cost function c and transition function f are continuous convex functions with compact domains; that U is a continuous set-valued function whose graph $\operatorname{graph} U$ is a convex set; and, furthermore, that for each state $x \in X$, $U(x)$ is non-empty.

In this context, the Wald-Bellman operator is the previously defined function

$$B : L^\infty(X) \to L^\infty(X),$$

where $L^\infty(X)$ is the Banach space of bounded functions defined on X [1]. The norm of a function $V \in L^\infty(X)$ is given by

$$\|V\|_\infty = \sup_{x \in X} |V(x)|.$$

[1] More accurately, $L^\infty(X)$ is the Banach space of equivalence classes of essentially bounded Borel functions defined on X.

The extent to which these assumptions are useful will become clear later. As an example, let $\overline{c} = \sup_{u \in \mathcal{U}} c(u)$ and $\underline{c} = \inf_{u \in \mathcal{U}} c(u)$, which are both finite due to compactness and continuity. Without loss of generality, assume $\underline{c} = 0$. Then it becomes possible to bound the total cost of an arbitrary policy by 0 from below and by $\frac{\overline{c}}{1-\beta}$ from above.

Finally, we note some theorems require analysis on $L^\infty(\hat{X})$, where $\hat{X} \supset X$ is an extended state space.

As an example, consider the hydrothermal problem 4.1, and assume it is stationary. The Wald-Bellman operator 4.3 becomes

$$B(V)(r_0) = \begin{cases} \min\limits_{h,g,df,s,r} & cg + pdf + \beta V(r) \\ \text{s.t.} & h + g + df = d, \\ & r + h + s = r_0 + i \\ & 0 \leq h \leq r_0, \\ & 0 \leq g \leq \overline{g}, \\ & 0 \leq df \leq \overline{df}, \\ & 0 \leq s \leq r_0. \end{cases} \tag{4.4}$$

Setting $V = 0$, the constant function everywhere zero, $B(V) = Q$, which is the value function of a one-stage problem, defined on equation 3.6. Now, computing $B^2(V) = B(B(V)) = B(Q)$, one obtains the value function of a two-stage hydrothermal problem, defined on equation 3.5. Proceding in this manner, it is natural to wonder whether the sequence of k-stage value functions

$$\left\{B^k(V)\right\}_{k \in \mathbb{N}}$$

converges, and whether it converges to the value function of the infinite horizon problem 4.1.

We now verify that the Wald-Bellman operator has some useful properties, and use them to prove theorems.

Lemma 4.1.7. *Let $V, V' \in L^\infty(X)$, and let $r \in \mathbb{R}$, but denote also by r the function in $L^\infty(X)$ everywhere equal to c.*

1. *If $V \leq V'$, then $B(V) \leq B(V')$. (Monotonicity.)*
2. *$B(V + r) = B(V) + \beta r$. (Scalar additivity.)*
3. *If V is convex and $\beta > 0$, then $B(V)$ is convex. (Convexity preserving.)*
4. *If $\beta \in (0, 1)$, then $\|B(V) - B(V')\|_\infty \leq \beta \|V - V'\|_\infty$. (Contracting.)*

Proof. 1. For every $x \in D$ and every $u \in U(x)$,

$$c(u) + \beta V(f(x,u)) \leq c(u) + \beta V'(f(x,u)).$$

It is immediate that

$$B(V)(x) = \min_{u \in U(x)} c(u) + \beta V(f(x,u)) \leq c(u) + \beta V(f(x,u)),$$

and thus that

$$B(V)(x) \leq c(u) + \beta V'(f(x,u)).$$

Compared to the original inequality, we have eliminated the left-hand side dependency on u. We can now minimize over $u \in U(x)$ on the right-hand side, yielding

$$B(V)(x) \leq B(V')(x)$$

for arbitrary $x \in X$. This asserts $B(V) \leq B(V')$ as stated.

2. By definition,

$$\begin{aligned} B(V+r) &= \min_{u \in U(x)} c(u) + \beta \left(V(f(x,u)) + r\right) \\ &= \left(\min_{u \in U(x)} c(u) + \beta V(f(x,u))\right) + \beta r \\ &= B(V) + \beta r. \end{aligned}$$

3. Remember that by assumption the original optimization problem is convex. Since the function f is linear, $V \circ f$ is a convex function of both x and u. Then $c + \beta V \circ f$ is also a convex function of both x and u. In addition, as $U(x)$ is a convex set for each x, $\mathbb{I}_{U(x)}(u)$ is a convex function.

4. Since $V, V' \in L^\infty(X)$, $\|V - V'\|_\infty < \infty$ and the inequality $V \leq V' + \|V - V'\|_\infty$ holds. Applying the Wald-Bellman operator on both sides, one can see that

$$B(V) \leq B(V') + \beta \|V - V'\|_\infty .$$

By symmetry, also $B(V') \leq B(V) + \beta \|V - V'\|_\infty$, showing that

$$\|B(V) - B(V')\|_\infty \leq \beta \|V - V'\|_\infty$$

as desired.

□

Lemma 4.1.7 allows us to show the following key result.

Theorem 4.1.8. *There exists a unique function $V^\star \in L^\infty(X)$ such that*

$$V^\star = B(V^\star). \tag{4.5}$$

Moreover, $V^\star$ is convex.

Equation 4.5 is called Bellman's equation*, and it can be more explicitly written as*

$$V^\star(x) = \min_{u \in U(x)} c(u) + \beta V^\star \circ f(x, u).$$

Proof. The existence and uniqueness of $V^\star$ follow from the Banach Fixed Point Theorem. (Remember that B is a contraction mapping from $L^\infty(X)$ into itself.) Now, the sequence $\left\{B^k(V)\right\}_{k\in\mathbb{N}}$, beginning from the zero function on X, is a sequence of convex functions that converges uniformly to $V^\star$ on X. The (pointwise) limit of convex functions is convex (theorem 2.2.5), and this completes the proof. □

We claim that problem 4.1, as a function of the initial reservoir level, is the solution $V^\star$ of the fixed-point problem

$$B\left(V^\star\right) = V^\star.$$

Theorem 4.1.8 guarantees that the optimal value function is stationary. It is natural to wonder if the optimal policy is also stationary, that is, if it is composed by a single decision-rule repeated at each stage. In that case, the optimal policy can be represented via the respective decision-rule associated with it. This property is particularly useful when the state and control spaces are finite, as decision-rule can be represented by a vector in $\mathcal{U}^{n_x} \subset \mathbb{R}^{n_x}$, and allows the application of algorithm 8.

In order to determine the answer to this question, for a fixed decision-rule π, we introduce a modified version of the Bellman operator, B_π.

$$B_\pi(V)(x) = c\Big(\pi(x)\Big) + \beta V \circ f\Big(x, \pi(x)\Big) \tag{4.6}$$

This new operator corresponds to acting according to the decision-rule π instead of the decision-rule that minimizes the right hand side of the equation. The same properties shown for the Wald-Bellman operator in lemma 4.1.7 can be shown for B_π, and therefore there exists a unique $V_\pi \in L^\infty(X)$ such that

$$V_\pi = B_\pi(V_\pi).$$

The value function V_π is the value function associated with the stationary policy composed by acting according to the decision-rule π at every stage. We can now state and prove the following result.

Theorem 4.1.9. *A stationary policy composed by the decision-rule π is optimal if and only if π attains the minimum in Bellman's equation for every $x \in D$, that is,*

$$B(V^\star) = B_\pi(V^\star).$$

Proof. If the above equation is satisfied, theorem 4.1.8 tells us $V^\star = B(V^\star) = B_\pi(V^\star)$. Because V_π is the unique solution of $V_\pi = B_\pi(V_\pi)$, we must have $V^\star = V_\pi$.

On the other hand, optimality of the stationary policy π means that $V^\star = V_\pi$. Using again the property $V_\pi = B_\pi(V_\pi)$, we obtain that $V^\star = B_\pi(V^\star)$. Finally, theorem 4.1.8 allows us to write $B(V^\star) = V^\star = B_\pi(V^\star)$. □

Given $V^\star$, we can calculate

$$\pi^\star(x) = \arg\min_{u \in U(x)} c(u) + \beta V^\star \circ f(x, u).$$

Whenever $\pi^\star : X \to \mathcal{U}$ is well-defined, the optimal stationary policy $\pi^\star$ is optimal. This is the case when we have relatively complete recourse (because then $U(x)$ is nonempty for every $x \in X$) and U is a compact operator (because both c and $V^\star \circ f(x, \cdot)$ are continuous functions of the control, and thus bounded on their domain). In particular, this holds when the state and control sets are finite.

Note this optimal policy does not need to be unique. For example: when there are multiple controls that attain the minimum for a specific x.

Theorem 4.1.10. *Let π and π' be decision-rules such that π minimizes the sum of the present cost and the discounted value function, assuming we continue with the stationary policy given by π'. In other words,*

$$B_\pi(V_{\pi'}) = B(V_{\pi'}),$$

or, written in another way,

$$c(\pi(x)) + \beta V_{\pi'} \circ f(x, \pi(x)) = \min_{u \in U(x)} c(u) + \beta V_{\pi'} \circ f(x, u).$$

Then $V_\pi \leq V_{\pi'}$, and strict inequality holds for at least one state x unless π' is optimal.

Proof. Using that $V_{\pi'} = B_{\pi'}(V_{\pi'})$ and the hypothesis, we obtain, for every $x \in D$,

$$V_{\pi'}(x) = B_{\pi'}(V_{\pi'})(x) = c(\pi'(x)) + \beta V_{\pi'} \circ f(x, \pi'(x)) \geq \min_{u \in U(x)} c(u) + \beta V_{\pi'} \circ f(x, u) = B(V_{\pi'})(x) = B_\pi(V_{\pi'})(x). \quad (4.7)$$

That is, $V_{\pi'} \geq B_\pi(V_{\pi'})$. We can apply B_π to both sides of this equation k times to get $B_\pi^k(V_{\pi'}) = B_\pi^{k+1}(V_{\pi'})$. All these inequalities together tell us

$$V_{\pi'} \geq B_\pi(V_{\pi'}) \geq B_\pi^2(V_{\pi'}) \geq \cdots \geq B_\pi^k(V_{\pi'}) \geq \cdots \geq \lim_{k\to\infty} B_\pi^k(V_{\pi'}) = V_\pi.$$

Now, if the equality $V_{\pi'} = V_\pi$ holds, then also $V_{\pi'} = B_\pi(V_{\pi'})$. Because by hypothesis $B_\pi(V_{\pi'}) = B(V_{\pi'})$, we obtain $V_{\pi'} = B(V_{\pi'})$ and thus $V_{\pi'} = V^\star$ is optimal by theorem 4.1.8. Therefore, if $V_{\pi'}$ is not optimal, strict inequality holds for at least one state x. □

We now define a way with which to measure how suboptimal any $V \in L^\infty(X)$ is in comparison to $V^\star$.

Definition 4.1.11 (Bellman error.)**.** Let $V \in L^\infty(X)$ and $x \in X$. We denote the *Bellman error of* V *at* x by

$$\mathrm{BE}_x(V) = |V(x) - B(V)(x)|\,.$$

We also denote the *Bellman error of* V by

$$\mathrm{BE}(V) = \sup_{x\in X} \mathrm{BE}_x(V).$$

Note that the Bellman error of V is the L^∞-distance between V and $B(V)$, *i.e.* $\mathrm{BE}(V) = \|V - B(V)\|_\infty$.

Lemma 4.1.12. *Let $V \in L^\infty(X)$ and $x \in X$. Then*

$$|V(x) - V^\star(x)| \leq \|V - V^\star\|_\infty \leq \frac{1}{1-\beta}\,\mathrm{BE}(V).$$

Proof. The first inequality is immediate from the definition of the L^∞-norm. Now, for the second inequality, we can repeatedly apply the triangle inequality to obtain

$$\|V - V^\star\|_\infty \leq \sum_{i=0}^{I} \left\|B^i(V) - B^{i+1}(V)\right\|_\infty + \left\|B^{I+1}(V) - V^\star\right\|_\infty.$$

Then, using that the Wald-Bellman operator is a contraction, we can bound each term of the form $\|B^i(V) - B^{i+1}(V)\|_\infty$ by $\beta^i \|V - B(V)\|_\infty$. Thus we get

$$\begin{aligned}\|V - V^\star\|_\infty &\leq \sum_{i=0}^{I} \beta^i \|V - B(V)\|_\infty + \left\|B^{I+1}(V) - V^\star\right\|_\infty \\ &= \frac{1-\beta^{I+1}}{1-\beta}\|V - B(V)\|_\infty + \left\|B^{I+1}(V) - V^\star\right\|_\infty.\end{aligned}$$

Taking the limit when $I \to \infty$, both β^{I+1} and $\left\|B^{I+1}(V) - V^\star\right\|_\infty$ vanish to zero, showing the desired result,

$$\|V - V^\star\|_\infty \leq \frac{1}{1-\beta}\|V - B(V)\|_\infty = \frac{1}{1-\beta}\,\mathrm{BE}(V).$$

□

We now show that iterating $V_0 = 0$ through the Wald-Bellman operator yields a k-stage finite horizon problem. Let $V_k = B^k(V_0)$ for $k \in \mathbb{N}$. For example, V_1 corresponds to an one-stage problem.

$$V_1(x_0) = B(V_0)(x_0) = \begin{cases} \min\limits_{x_1,u_1} & c(u_1) \\ \text{s.t.} & x_1 = f(x_0, u_1), \\ & u_1 \in U(x_0). \end{cases}$$

However, for the base of our induction, we show V_2 is a two-stage problem, as it better reflects the general case.

Lemma 4.1.13. *For each $x_0 \in X$, $V_k(x_0)$ is a k-stage finite horizon problem.*

Proof. To begin with, by definition,

$$V_2(x_0) = B(V_1)(x_0) = \begin{cases} \min\limits_{x_1,u_1} & c(u_1) + \beta \left(\min\limits_{\substack{x_2 = f(x_1,u_2), \\ u_2 \in U(x_1)}} c(u_2) \right) \\ \text{s.t.} & x_1 = f(x_0, u_1), \\ & u_1 \in U(x_0). \end{cases}$$

The nested minima can be merged into a single minimum over all variables. Denoting by $x = (x_1, x_2)$ and $u = (u_1, u_2)$, we have,

$$V_2(x_0) = \begin{cases} \min\limits_{x,u} & c(u_1) + \beta c(u_2) \\ \text{s.t.} & x_t = f(x_{t-1}, u_t), \quad t \in [1..2], \\ & u_t \in U(x_{t-1}), \quad t \in [1..2]. \end{cases}$$

Now, assume that for $t \in [0..k]$, V_t corresponds to a t-stage subproblem. Evaluating V_{k+1}, we obtain

$$V_{k+1}(x_0) = B(V_k)(x_0) = \begin{cases} \min\limits_{x_1,u_1} & c(u_1) + \beta V_k(x_1) \\ \text{s.t.} & x_1 = f(x_0, u_1), \\ & u_1 \in U(x_0). \end{cases}$$

Expanding $V_k = B(V_{k-1})$, the objective can be written as

$$c(u_1) + \beta \left(\min_{\substack{x_2 = f(x_1,u_2), \\ u_2 \in U(x_1)}} c(u_2) + \beta V_{k-1}(x_2) \right).$$

Grouping the minimization operators as before,

$$V_{k+1}(x_0) = \begin{cases} \min\limits_{x,u} & c(u_1) + \beta c(u_2) + \beta^2 V_{k-1}(x_2) \\ \text{s.t.} & x_t = f(x_{t-1}, u_t), \quad t \in [1..2], \\ & u_t \in U(x_{t-1}), \quad t \in [1..2]. \end{cases}$$

Repeating this process recursively, it finally yields,

$$V_{k+1}(x_0) = \begin{cases} \min\limits_{x,u} & \sum_{t=1}^{k+1} \beta^{t-1} c(u_t) \\ \text{s.t.} & x_t = f(x_{t-1}, u_t), \quad t \in [1..k+1], \\ & u_t \in U(x_{t-1}), \quad t \in [1..k+1]. \end{cases}$$

The induction is complete. Note in this final problem, the symbols x and u represent $(k+1)$ variables each, unlike the previous instances in this proof where *e.g.* $x = (x_1, x_2)$ represented two variables (also recall each x_t is a vector in $\mathbb{R}^{n_x}$). Furthermore, despite sharing the same symbol, the value of $x_1^\star$ in an optimal solution of V_{k+1} and V_2 need not be equal. □

Our goal now is to show that both definitions of the infinite horizon problem—as fixed-point of the Wald-Bellman operator, and as minimum of an infinite series—are equivalent under the convexity, continuity, and compactness assumptions of remark 4.1.6. In this setting, for any $x_0 \in X$, there exists a sequence $\{(x_t, u_t)\}_{t=1}^{\infty}$ such that $x_t = f(x_t, u_t)$ and $u_t \in U(x_{t-1})$. Also, we assume $\underline{c} = 0$, which ensures the limit in the equation below is well-defined for every such feasible sequence of decisions. Denote by V_∞ the formulation of the infinite horizon problem given in equation 4.2, that is,

Recall $U(x)$ is non-empty for every $x \in X$.

$$V_\infty(x_0) = \begin{cases} \min\limits_{x,u} & \lim\limits_{T\to\infty} \sum\limits_{t=1}^{T} \beta^{t-1} c_t(u_t) \\ \text{s.t.} & x_{t+1} = f_t(x_t, u_t), \\ & u_t \in U_t(x_t). \end{cases} \tag{4.8}$$

Theorem 4.1.14. *Under the previously stated assumptions,*

$$V_\infty = V^\star.$$

Proof. We split the objective of V_∞ into the costs accumulated up to (and including) a stage k, and the costs incurred at later stages.

$$\begin{array}{ll} \min\limits_{x,u} & \sum\limits_{t=1}^{k} \beta^{t-1} c_t(u_t) + \lim\limits_{T\to\infty} \sum\limits_{t=k+1}^{T} \beta^{t-1} c_t(u_t) \\ \text{s.t.} & x_{t+1} = f_t(x_t, u_t), \\ & u_t \in U_t(x_t). \end{array}$$

The costs incurred from stage $k+1$ onwards can be bounded above regardless of the (feasible) sequence of controls chosen.

$$\lim_{T\to\infty} \sum_{t=k+1}^{T} \beta^{t-1} c_t(u_t) \leq \lim_{T\to\infty} \sum_{t=k+1}^{T} \beta^{t-1} \overline{c} = \frac{\beta^k \overline{c}}{1-\beta}.$$

In particular, for any $x_0 \in X$, $V_\infty(x_0) \leq \frac{\overline{c}}{1-\beta}$.

Now, fix $x_0 \in X$. Let $\left\{(x_t^{k,\star}, u_t^{k,\star})\right\}_{t=1}^{k}$ be an optimal solution of $V_k(x_0)$ and $\{(x_t^{\infty,\star}, u_t^{\infty,\star})\}_{t=1}^{\infty}$ be an optimal solution of $V_\infty(x_0)$. Denote by $\{(\hat{x}_t^{\infty,\star}, \hat{u}_t^{\infty,\star})\}_{t=1}^{\infty}$ the sequence of decisions formed by appending to the optimal solution of $V_k(x_0)$ an optimal solution of $V_\infty(x_k^{k,\star})$.

$$\begin{aligned}
V_k(x_0) &= \sum_{t=1}^{k} \beta^{t-1} c(u_t^{k,\star}) \leq \sum_{t=1}^{k} \beta^{t-1} c(u_t^{\infty,\star}) \qquad \text{(By optimality of } V_k(x_0).) \\
&\leq \sum_{t=1}^{k} \beta^{t-1} c(u_t^{\infty,\star}) + \sum_{t=k+1}^{\infty} \beta^{t-1} c(u_t^{\infty,\star}) = V_\infty(x_0) \\
&\leq \sum_{t=1}^{k} \beta^{t-1} c(\hat{u}_t^{\infty,\star}) + \sum_{t=k+1}^{\infty} \beta^{t-1} c(\hat{u}_t^{\infty,\star}) = V_k(x_0) + \beta^k V_\infty(x_k^{k,\star}) \\
&\qquad\qquad \text{(By optimality of } V_\infty(x_0).) \\
&\leq V_k(x_0) + \frac{\beta^k \overline{c}}{1-\beta}.
\end{aligned}$$

In summary,

$$V_k(x_0) \leq V_\infty(x_0) \leq V_k(x_0) + \frac{\beta^k \overline{c}}{1-\beta}.$$

Because $x_0 \in X$ was fixed arbitrarily, $\|V_\infty - V_k\|_\infty \leq \frac{\beta^k \overline{c}}{1-\beta}$. Taking the limit when $k \to \infty$ completes the proof. □

See proposition 1.2.1 of [5] for an alternative proof.

4.2 Algorithms

In this section, solution methods for infinite horizon problems are discussed. We briefly interrupt our previous focus to present the relatively simpler case of discrete state and control spaces, as much of the theory developed so far is directly applicable. Afterwards, we examine algorithms for the continuous state and control space case.

4.2.1 Discrete infinite horizon problems

First, we consider algorithms for solving discounted infinite horizon problems where the state, control, and random variables can take a finite number of values. In this case, the value function $V^\star$ and each other function in $L^\infty(D^\star)$ can be represented by a vector in $\mathbb{R}^{|X|}$. Because lemma 4.1.7 guarantees the Wald-Bellman operator is a contraction, taking any initial value function V_0, the sequence $\left\{B^k(V_0)\right\}_{k\in\mathbb{N}}$ converges to $V^\star$. This algorithm is known as *value iteration*, and by lemma 4.1.12 we can terminate it when the Bellman error of the current value function estimate is sufficiently small.

Algorithm 4.1: Value iteration.

Data: an initial value function V_0 and a tolerance $\varepsilon > 0$.
Result: a value function.

```
1  set V ← V_0
2  set V_next ← V_0
3  set bound ← ∞
4  while bound ≥ ε do
5  |   for i ∈ [1..|X|] do
6  |   |   solve B(V)(x_i) and save at V_next(x_i)
7  |   end
8  |   set bound ← 1/(1−β) BE(V)        // Note that V_next = B(V).
9  |   set V ← V_next
10 end
```

In line 8, it is possible to obtain a tighter bound by computing the maximum and minimum of $V - B(V)$ instead (note the absence of absolute value), but we use the Bellman error for a simpler pseudocode. Furthermore, the algorithm above keeps two vectors of length n_x stored in memory during its execution: V and V_{next}. An alternative, in line 6, is to overwrite $V(x_i)$ with $B(V)(x_i)$. Thus, during the next iteration of the for-loop, we would be calculating $B(V')(x_{i+1})$, where $V'(x_i) = B(V)(x_i)$ is possibly different from $V(x_i)$. This is a variant of the value iteration algorithm, known as the *Gauss-Seidel version* of value iteration. We refer the reader to subsection 1.3.1 of [5] for a more detailed discussion of value iteration with tighter error bounds, as well as subsection 1.3.2 which discusses variants (including Gauss-Seidel).

Although value iteration is guaranteed to converge to the optimal value function, it may require an infinite number of iterations, unlike the DP algorithm. Under the assumption both the state and control spaces are finite, there is a finite number of decision-rules. As such, there is a finite number of stationary policies, one of which is guaranteed to be optimal by theorem 4.1.9. Recall that theorem 4.1.10 guarantees a policy can always be strictly improved upon unless it is optimal. These properties suggest the following algorithm, known as *policy iteration*.

Algorithm 4.2: Policy iteration.

Data: an initial decision-rule π_0.
Result: a decision-rule.

set $\pi \leftarrow \pi_0$
while $V_\pi \neq B(V_\pi)$ **do**
  set $V_\pi(x) = \frac{1}{1-\beta} c(x, \pi(x))$ `// Policy evaluation step.`
  calculate π_{next} such that $B_{\pi_{\text{next}}} V_\pi = B(V_\pi)$ `// Policy improvement step.`
  set $\pi \leftarrow \pi_{\text{next}}$
end

Note the policy improvement step requires solving n_x optimization problems. Subsection 1.3.3 of [5] presents a comprehensive description of the policy iteration method, as well as a combination of value and policy iteration called *asynchronous policy iteration*.

4.2.2 Continuous infinite horizon problems

We now discuss algorithms of similar structure to the DDP algorithm for solving discounted infinite horizon problems where the state and control spaces are continuous. In this setting, the Wald-Bellman operator is a contraction, and we can find $T \in \mathbb{N}$ such that a T-stage problem approximates the infinite horizon problem to a specified degree of accuracy. However, such T can be very large, and in that case be challenging to solve. We suggest two alternatives, the first of which we call *exhaustive*, and the second we call *exploratory*. Both algorithms keep a piecewise linear underestimator of the optimal value function. The exhaustive algorithm operates by improving the estimate at a preselected finite set of states. It is based on the GDDP algorithm (cf. [30]) and it is similar to other existing algorithms such as approximate value iteration. For an early work on discretization procedures in finite and infinite horizon settings, see [3] (note its structure is *not* similar to DDP). The exploratory algorithm uses the value function estimate to solve a finite horizon problem, and improves the estimate at the states visited. The time horizon is then increased, and the process is repeated using the updated estimate. It is based on the Benders squared algorithm (cf. [21]), and can be seen as an extension of the DDP algorithm to infinite horizon problems.

We now describe how the exhaustive algorithm attains a good approximation of the optimal value function $V^\star$. Denote by $\underline{V} \in L^\infty(X)$ a known underestimator of $V^\star$. Let $\mathcal{N}$ be an ε-net of X, that is, for any $x \in X$ there is an $x_{\mathcal{N}} \in \mathcal{N}$ such that $\|x - x_{\mathcal{N}}\| \leq \varepsilon$.

Letting $M = \max_{x \in \mathcal{N}} \text{BE}(\underline{V})(x)$, and using a Lipschitz constant L, we

get

$$\|\underline{V} - V^\star\|_\infty \leq \frac{1}{1-\beta}\left(M + L\varepsilon\right).$$

Lemma 4.1.7 tells us that $B(\underline{V})$ is also an underestimator of $V^\star$, and that

$$\|B(\underline{V}) - V^\star\|_\infty \leq \beta \|\underline{V} - V^\star\|_\infty .$$

However, rather than calculating the sequence of iterates $\left\{B^k(\underline{V})\right\}_{k\in\mathbb{N}}$, we use $B(\underline{V})$ to generate a new cut with which to improve our underestimator, that is

$$\underline{V}_{\text{next}}(x) = \max\left\{\underline{V}(x), \langle a, x\rangle + b\right\}.$$

This guarantees our iterates are nondecreasing. Under the assumption that $V^\star$ can be extended to a convex function on X_δ, where $\delta > 0$, we can apply lemma 2.5.5 to obtain a uniform Lipschitz constant to our iterates.

Prior to explaining the exploratory algorithm in depth, we need to discuss one of its key features, that distinguishes it from simply solving a sufficiently large finite horizon problem: *cut sharing*. In a finite horizon problem, the value functions at each stage are different from each other, and an underestimator of the value function of a specific stage is not necessarily an underestimator of the value function at a different stage. In a stationary infinite horizon problem, the optimal value function is the same in all stages, allowing us to use the same underestimator across all stages. For that reason, we keep a set C_k of cuts accumulated up to the k-th step of the algorithm, and in our notation we use it as a superscript. The value function estimate given the accumulated cuts is

$$\underline{V}^{C_k}(x) = \max_{(a^i,b^i)\in C_k} \left\langle a^i, x\right\rangle + b^i,$$

and its image through the Wald-Bellman operator is

$$B\left(\underline{V}^{C_k}\right)(x) = \min_{u\in U(x)} c(u) + \beta \underline{V}^{C_k} \circ f(x,u).$$

The initial state visited during each forward step is always the same, $x_0^{C_k}$, so we can omit the superscript. The control $u_t^{C_k}$ minimizes $B\left(\underline{V}^{C_k}\right)\left(x_t^{C_k}\right)$. Finally, $x_{t+1}^{C_k} = f\left(x_t^{C_k}, u_t^{C_k}\right)$. Whenever strong duality holds, the cut generated at $x_t^{C_k}$ is equivalent to

$$\left\langle \partial_{x_t^{C_k}} B\left(\underline{V}^{C_k}\right), x - x_t^{C_k}\right\rangle + B\left(\underline{V}^{C_k}\right)\left(x_t^{C_k}\right).$$

Now that we have fixed the notation, the proof of convergence for the exploratory algorithm requires the following three main ideas and additional assumptions.

Backwards induction This is a technique commonly used in SDDP-type algorithm convergence proofs. The convergence of $\mathfrak{Q}_T^k$ to Q_T as $k \to \infty$ is used to ensure $\mathfrak{Q}_{T-1}^k \to Q_{T-1}$. In turn, the convergence of $\mathfrak{Q}_{T-1}^k$ to Q_{T-1} guarantees $\mathfrak{Q}_{T-2}^k \to Q_{T-2}$, and so on until the proof that $\mathfrak{Q}_0^k \to Q_0$ is attained. Recall that we assumed Q_T is the constant function everywhere zero, and thus actually $\mathfrak{Q}_T^k = Q_T$ holds for all iterations k.

In the infinite horizon setting, there is no last stage to serve as the basis of induction. However, we can compare $\underline{V}^{C_k}$ with a T-stage truncation of the original infinite horizon problem that is sufficiently close to the original problem, whose future cost function at stage t is V_t. That is to say, V_0 is close to the optimal value function $V^\star$ in L^∞-norm. Importantly, *the finite horizon approximation is not computed during execution*: it serves as an analytical tool to show convergence.

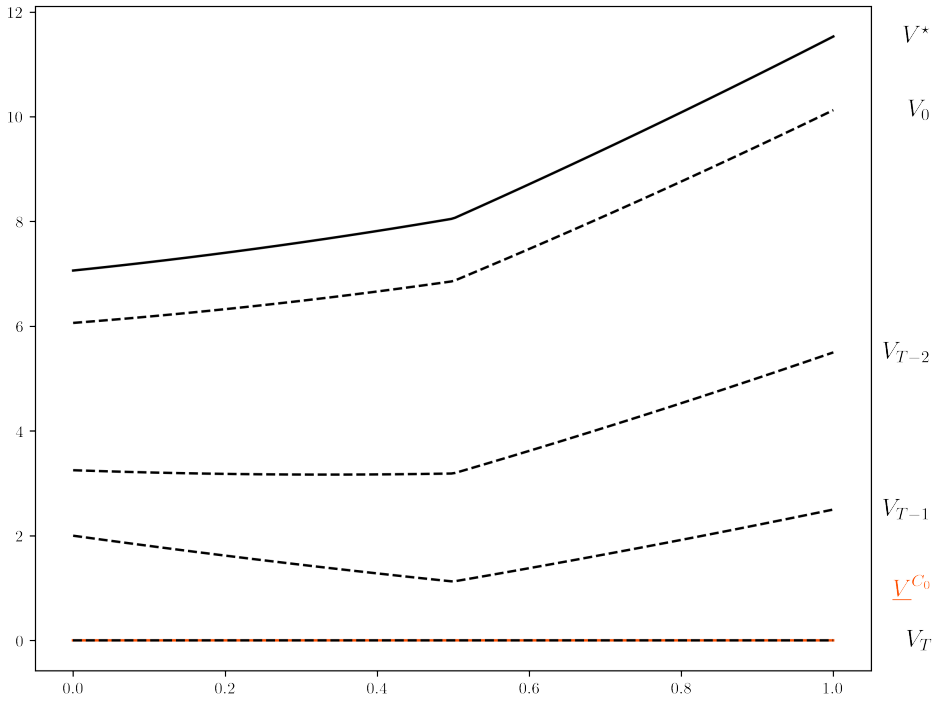

Figure 4.3: Initial approximation $\underline{V}^{C_0}$ (in orange, solid), optimal value function $V^\star$ (in black, solid), and finite horizon future cost functions V_t (in black, dashed).

Let the lower bound on the control cost be $0 \leq \underline{c} = \min_{u \in \bigcup_x U(x)} c(u)$, and denote by $\overline{c}$ the similarly defined upper bound. To our ends, we need to fix a sufficiently large $T(\varepsilon)$ (although we suppress the dependency on ε) such that

$$\sum_{t=T}^{\infty} \beta^{t-1} \overline{c} = \frac{\beta^T \overline{c}}{1-\beta} < \frac{\varepsilon}{2}.$$

Thus for any choice of initial estimate $\underline{V}^{C_0}$ such that $V^\star \geq \underline{V}^{C_0} \geq 0$ we guarantee that

$$\left\|V^\star - B^T\left(\underline{V}^{C_0}\right)\right\|_\infty \leq \beta^T \left\|V^\star - \underline{V}^{C_0}\right\|_\infty \leq \frac{\beta^T \overline{c}}{1-\beta}.$$

Furthermore, we can assume that, for $t > T$, V_t is the constant function equal to $\frac{c}{1-\beta}$, and otherwise $V_t = B^{T-t+1}(V_{T+1})$, that is, $V_T = B(V_{T+1})$, and $V_{T-1} = B(V_T) = B^2(V_{T+1})$, and so on. (The constant cut $\frac{c}{1-\beta}$ is the highest available without further assumptions or information about a particular problem instance.)

Monotone limit Because we do not remove cuts, the sequence of lower estimates $\left\{\underline{V}^{C_k}\right\}_{k\in\mathbb{N}}$ is nondecreasing. Lemma 4.1.7 guarantees the sequence $\{V_t\}_{t=0}^T$ is monotone and approaching $V^\star$ from below. When comparing our optimal value function estimate with the finite horizon future cost functions, our goal will be to show that, at the states explored by the algorithm, our estimate will eventually be above the future cost functions V_t. Actually, we try to show that a $\frac{\varepsilon}{2}$-relaxed condition holds,

$$\underline{V}^{C_{k+1}}(x_t^{C_k}) \geq V_t(x_t^{C_k}) - \frac{\varepsilon}{2}. \tag{4.9}$$

Local improvement The final step is to prove that condition 4.9 holds not only at $x_t^{C_k}$, but for every state in an open ball of radius $r > 0$ around it. (the radius r can depend on t, but not the state, however we suppress the dependency.) By assumption, the state variable belongs to a compact set, which has a finite packing number, *i.e.* the maximum cardinality of separated sets of a given size. In particular, the maximum cardinality of an r-separated set is finite, suggesting that the condition can only be violated at a finite number of $x_t^{C_k}$ for each t.

Local improvement requires that each V_t is Lipschitz with constant L_t, and that all functions $\underline{V}^{C_k}$ are Lipschitz with constant bounded above by ℓ. The first part can be verified as in section III of [3] (there the V_t are denoted by J_k). The work uses assumptions B defined in section II and modified in section IV for the stationary infinite horizon setting. The second part can be verified if the conditions of lemma 2.5.5 hold. By construction, the family $\left\{\underline{V}^{C_k}\right\}_{k\in\mathbb{N}} \cup \{V^\star\}$ is nondecreasing, and their elements are below $V^\star$. Also, as each $\underline{V}^{C_k}$ is a maximum of affine functions, they have readily available extensions to X_δ. The missing piece is whether $V^\star$ can be extended to X_δ, which *we assume to hold*. We consider this assumption to resemble, for example, assumption $H_1(6)$ of [14], used in the proof of convergence of SDDP-type algorithms. (An alternative would be to assume $V^\star$ is Lipschitz continuous directly, and build an extension to X_δ.)

Theorem 4.2.1 (Convergence of the exploratory algorithm)**.** *For any $\varepsilon > 0$, for sufficiently large iteration number k, the value function estimate at the state visited by the exploratory algorithm in the t-th stage satisfies*

$$\underline{V}^{C_{k+1}}(x_t^{C_k}) \geq V_t(x_t^{C_k}) - \frac{\varepsilon}{2}.$$

Proof. Fix T such that the contribution of stages larger than T to the cost of the infinite horizon problem is less than $\frac{\varepsilon}{2}$. Also, assume that the cut given by the constant function equal to $\frac{c}{1-\beta}$ is implied by C_k (*e.g.* by initializing C_0 with it, since we do not remove cuts). Note that this implies the theorem statement for $t > T$.

Now, choose $\varepsilon_T < \cdots < \varepsilon_t < \cdots < \varepsilon_1 \leq \frac{\varepsilon}{2}$. We are going to show that, for large enough k,

$$\underline{V}^{C_{k+1}}(x_t^{C_k}) \geq V_t(x_t^{C_k}) - \varepsilon_t,$$

which implies the inequality in this theorem's statement.

At the k-th step of the algorithm, let $\tau \in [1..T]$ be the largest index such that

$$\underline{V}^{C_k}(x_\tau^{C_k}) < V_\tau(x_\tau^{C_k}) - \varepsilon_\tau.$$

Denote the optimal objective function estimate starting from state $x_\tau^{C_k}$ by

$$v = B\left(\underline{V}^{C_k}\right)(x_\tau^{C_k}) = c(u_\tau^{C_k}) + \beta \underline{V}^{C_k}(x_{\tau+1}^{C_k}).$$

By assumption, $\underline{V}^{C_k}(x_{\tau+1}^{C_k}) \geq V_{\tau+1}(x_{\tau+1}^{C_k}) - \varepsilon_{\tau+1}$. This implies that

$$v \geq c(u_\tau^{C_k}) + \beta V_{\tau+1}(x_{\tau+1}^{C_k}) - \beta\varepsilon_{\tau+1}.$$

Also, $c(u_\tau^{C_k}) + \beta V_{\tau+1}(x_{\tau+1}^{C_k})$ is the cost of performing the control $u_\tau^{C_k}$ starting from state $x_\tau^{C_k}$ at stage τ, *i.e.*, it is underestimated by $V_\tau(x_\tau^{C_k})$,

$$v \geq c(u_\tau^{C_k}) + \beta V_{\tau+1}(x_{\tau+1}^{C_k}) \geq V_\tau(x_\tau^{C_k}) - \beta\varepsilon_{\tau+1}.$$

Then

$$\underline{V}^{C_k}(x_\tau^{C_k}) < V_\tau(x_\tau^{C_k}) - \varepsilon_\tau < V_\tau(x_\tau^{C_k}) - \varepsilon_{\tau+1} < V_\tau(x_\tau^{C_k}) - \beta\varepsilon_{\tau+1} \leq v,$$

resulting in an useful Benders cut that improves our estimate at $x_\tau^{C_k}$ by at least $\varepsilon_t - \beta\varepsilon_{t+1}$. We set $\hat{\varepsilon}_t = \varepsilon_t - \varepsilon_{t+1}$, which is an even lower bound on our improvement, but simplifies a later calculation. These inequalities and the strict improvement are illustrated in figure 4.4.

It is possible to use a backwards update scheme, similar to the DDP algorithm, where the new cut at $x_t^{C_k}$ is calculated using an estimate that contains more cuts (*e.g.* the cut at $x_{t+1}^{C_k}$). In that case, we have $k' \geq k$,

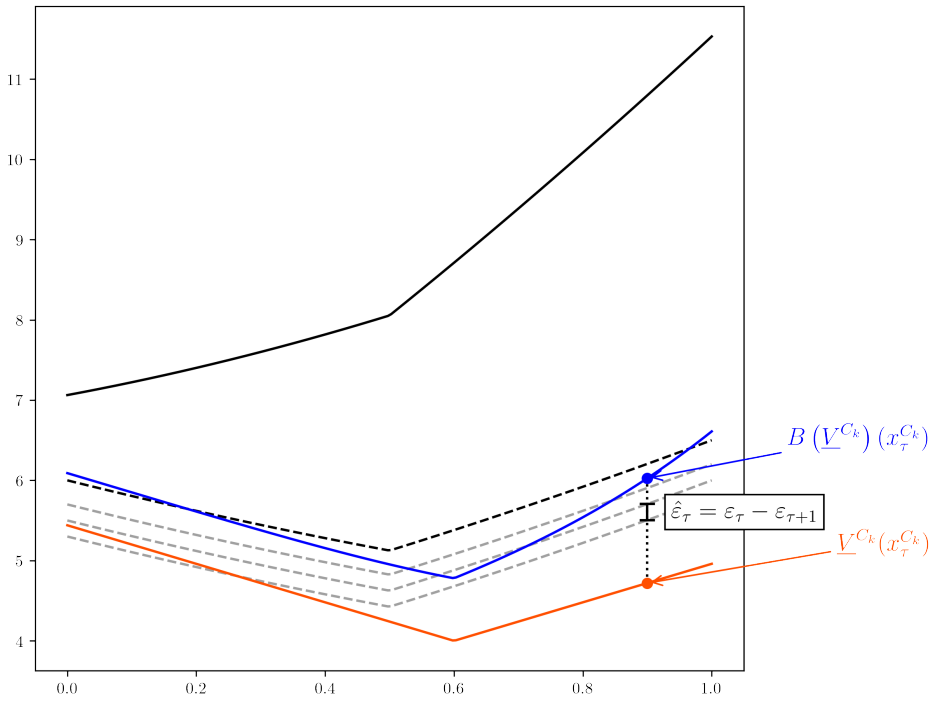

Figure 4.4: The approximation $\underline{V}^{C_k}$ (in orange, solid), and its image through the Wald-Bellman operator (in blue, solid). Note we only compute $B\left(\underline{V}^{C_k}\right)(x_\tau^{C_k})$, rather than $B\left(\underline{V}^{C_k}\right)$ in its entirety. The figure also depicts the optimal value function $V^\star$ (in black, solid); the future cost function V_τ (in black, dashed); and the three translations of V_τ by $-\beta\varepsilon_{\tau+1}$, $-\varepsilon_{\tau+1}$, and $-\varepsilon_\tau$ (in gray, dashed).

implying that $C_{k'} \supset C_k$ and thus $\underline{V}^{C_{k'}} \geq \underline{V}^{C_k}$. During the update step, we calculate $B\left(\underline{V}^{C_{k'}}\right)(x_\tau^{C_k})$, and monotonicity of the Wald-Bellman operator guarantees that

$$B\left(\underline{V}^{C_{k'}}\right)(x_\tau^{C_k}) \geq B\left(\underline{V}^{C_k}\right)(x_\tau^{C_k}) = v > \underline{V}^{C_k}(x_\tau^{C_k}),$$

and we have an improving Benders cut, as before.

Under the Lipschitz continuity conditions, the inequality $V_t(x) - \varepsilon \leq \underline{V}^{C_{k+1}}(x)$ can be guaranteed to hold for every $x \in \mathrm{B}_r(x_t^{C_k})$, where $r > 0$ can be solved for in the following equation,

$$V_t(x_t^{C_k}) - \varepsilon_t + L_t r = V_t(x_t^{C_k}) - \varepsilon_t + \hat{\varepsilon}_t - \ell r.$$

The term $V_t(x_t^{C_k}) - \varepsilon_t$ in the right-hand side arises from assuming $v = V_t(x_t^{C_k}) - \varepsilon_t + \hat{\varepsilon}_t$, the least increase that can happen of the estimate at $x_t^{C_k}$[2]. It is simple to see that, whenever L_t and ℓ are nonzero, $r = \frac{\hat{\varepsilon}}{L_t + \ell} > 0$. This is illustrated by figure 4.5. Recall that $\underline{V}^{C_{k+1}}(x_\tau^{C_k}) = B\left(\underline{V}^{C_k}\right)(x_\tau^{C_k})$, and thus figure 4.5 can be seen as a close-up of figure 4.4 around $B\left(\underline{V}^{C_k}\right)(x_\tau^{C_k})$ after our optimal value function estimate is updated.

As previously discussed, compactness ensures the maximum cardinality of a r-separated set is finite, and thus for each t the condition can only be violated at a finite number of $x_t^{C_k}$. That is, the set

$$\left\{k \in \mathbb{N} : \underline{V}^{C_k}(x_t^{C_k}) < V_t(x_t^{C_k}) - \varepsilon_t\right\}$$

is finite for each t. Also, if the condition is satisfied at a point for a given k, it is satisfied at that point for all larger values of k.

□

[2]Actually, this assumes v is lower than the least increase, as we set $\hat{\varepsilon}_t < \varepsilon_t - \beta\varepsilon_{t+1}$.

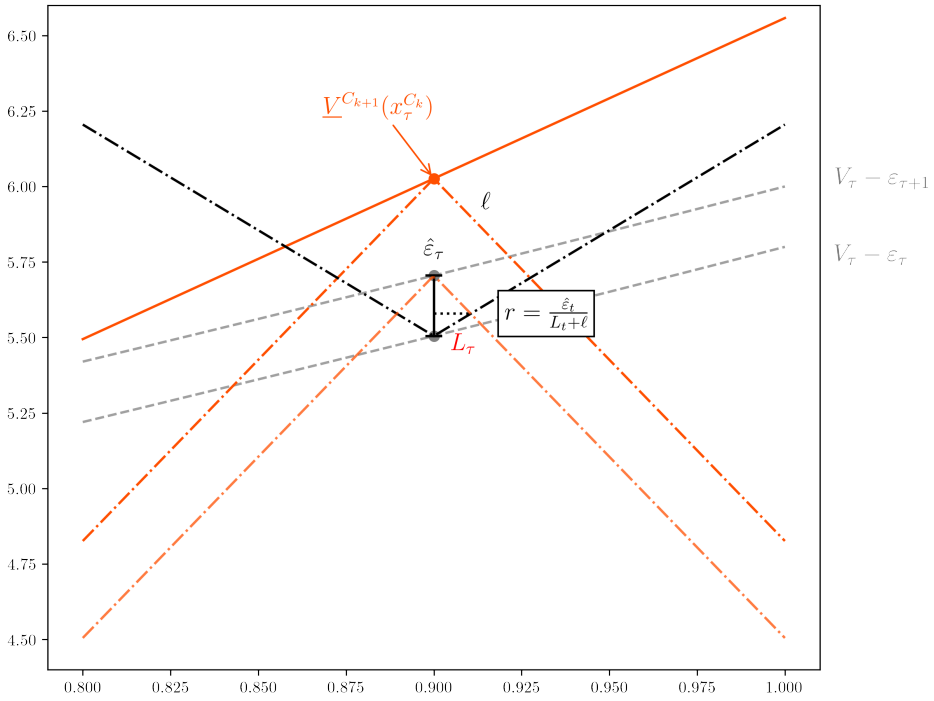

Figure 4.5: The approximation $\underline{V}^{C_{k+1}}$ (in orange, solid), after the update, and translations of V_τ (in gray, dashed). Lipschitz estimates of each function around x_τ are also shown (in their respective color, dot-dashed), as well as a translation of the Lipschitz estimate of $\underline{V}^{C_{k+1}}$ (in light orange, dot-dashed).

4.3 Examples

In this section, we go over examples of infinite horizon problems that are of practical and theoretical interest. In particular, the hydrothermal model discussed previously is modified to more closely resemble the Brazilian interconnected power system problem. Some questions regarding infinite horizon problems are posed, alongside relevant examples (sometimes indicating a negative answer). Finally, we conclude with a final discussion.

The models here presented were implemented using the *SDDP.jl* library (cf. [13] and [12]) in Julia 1.5.2, and solved with Gurobi.

4.3.1 Hydrothermal problem

The hydrothermal optimization models seen through this work have been motivated by the Brazilian interconnected power system. The multistage version in equation 3.11 was simplified for exposition. A subtle but important distinction of scope must be made. We convert the model from one-dimensional variables to *multidimensional variables*. For example, rather than representing the generation target g_t of a single thermal power plant (at stage t), g_t is a vector whose entries represent the generation targets of each plant in the system.

Related to this change in dimensionality, the country is divided into four *subsystems* or *regions*—Southeast, South, Northeast, and North—plus an additional *transshipment node*. Each subsystem can be thought of as an instance of problem 3.11, with its own (hydro and thermal) power plants, demand, and so on. However, some subsystems have the ability to *exchange* energy with each other, justifying the construction of a single model containing all subsystems. The energy from subsystem j transferred to subsystem k at stage t is denoted by $ex_{j\to k,t}$, and it is bounded above by $\overline{ex}_{j\to k,t}$. This process has a cost $b_{j\to k,t}$, and only a fraction $\ell_{j\to k,t}$ of the energy sent arrives at its destination (the rest is lost). In this notation, the index 5 denotes the transshipment node, which is not a region: it simply mediates the energy exchange between some subsystems. We avoid using indices to represent the subsystem each variable belongs to whenever there is no ambiguity. If necessary, we write *e.g.* $g_{j,t}$ to denote the vector of generation targets of the thermal power plants that belong to subsystem j.

When j cannot transfer energy to k, $\overline{ex}_{j\to k,t} = 0$.

Two minor details remain. First, the objective has a cost a_t penalizing the spill s_t during stage t. Second, the thermal power plants have a minimum generation target $\underline{g}_t$, which may be strictly positive for some particular

plants.

$$
\begin{aligned}
\min_{\substack{h,g,df,\\ s,r,ex}} \quad & \sum_{t=1}^{T} \beta^{t-1} \left(c_t^\top g_t + p_t^\top df_t + a_t^\top s_t + b_t^\top ex_t \right) \\
\text{s.t.} \quad & h_{j,t} + g_{j,t} + df_{j,t} - \sum_{k=1}^{5} ex_{j\to k,t} + \sum_{k=1}^{5} \ell_{k\to j,t} ex_{k\to j,t} = d_{j,t}, \;\; j \in [1..4], \\
& r_{j,t} + h_{j,t} + s_{j,t} = r_{j,t-1} + i_{j,t}, \quad j \in [1..4], \\
& \sum_{j=1}^{4} ex_{j\to 5,t} = \sum_{j=1}^{4} ex_{5\to j,t}, \\
& 0 \le h_t \le \overline{h}_t, \\
& \underline{g}_t \le g_t \le \overline{g}_t, \\
& 0 \le df_t \le \overline{df}_t, \\
& 0 \le r_t \le \overline{r}_t, \\
& 0 \le s_t, \\
& 0 \le ex_t \le \overline{ex}_t.
\end{aligned}
\tag{4.10}
$$

We consider two instances of this model, the Finite Horizon (FH) with $T = 120$ stages representing 120 months, and the Infinite Horizon (IH) (corresponding to $T = \infty$). The problem data—consisting of all constants and the distribution of the random inflow i—is 12-periodic, coinciding with a yearly cycle. The discount factor is chosen as $\beta = 0.9906$, amounting to a yearly discount rate of $\beta^{-12} = 1.12$.

Each model was trained for 2000 iterations using SDDP, then both were simulated on the same sample of 2000 scenarios. Figure 4.6 shows the average reservoir volume at the end of each stage. An *end-of-horizon effect* is particularly noticeable in the Southeast region: at the final stages of the FH horizon model, the stored volume plummets. Figure 4.7 illustrates the mean total deficit along each stage, while figure 4.8 also shades the area corresponding to one standard deviation above the mean. The IH model mostly produces lower peaks than the FH model at stages that have a large deficit. In our sample, at any given stage, 90% of scenarios had zero deficit, explaining the large standard deviations as a small number of scenarios with high deficit values.

For analyses of similar models: [28] compares risk neutral (average cost) and risk averse finite horizon models; and [27] features numerical experiments comparing finite and infinite, risk neutral and averse models.

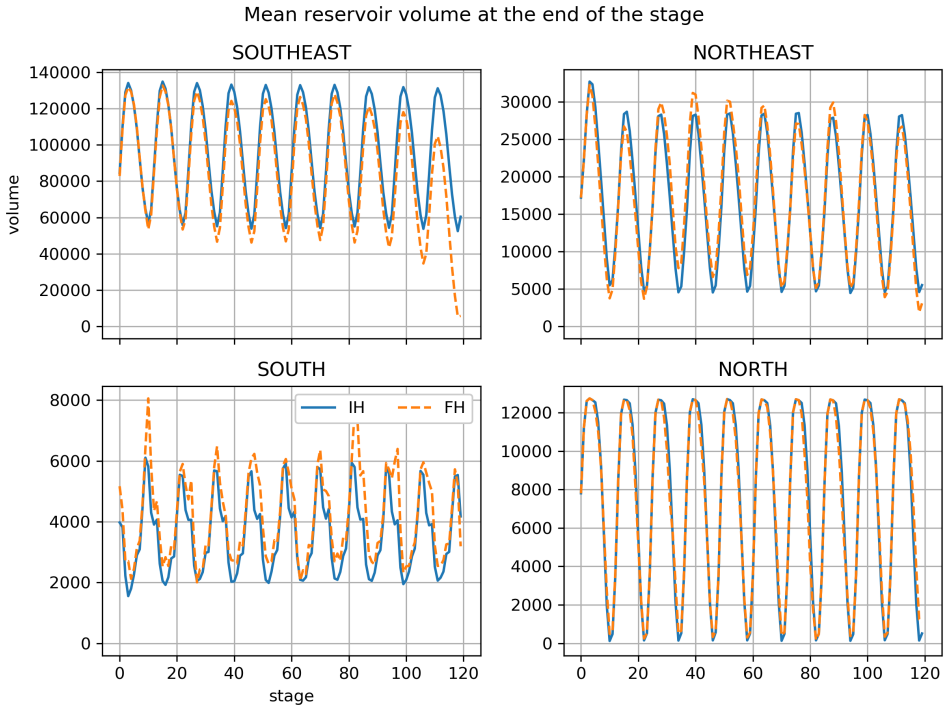

Figure 4.6: Mean reservoir volume at the end of each stage, divided by region, for models FH (in orange, dashed) and IH (in blue, solid).

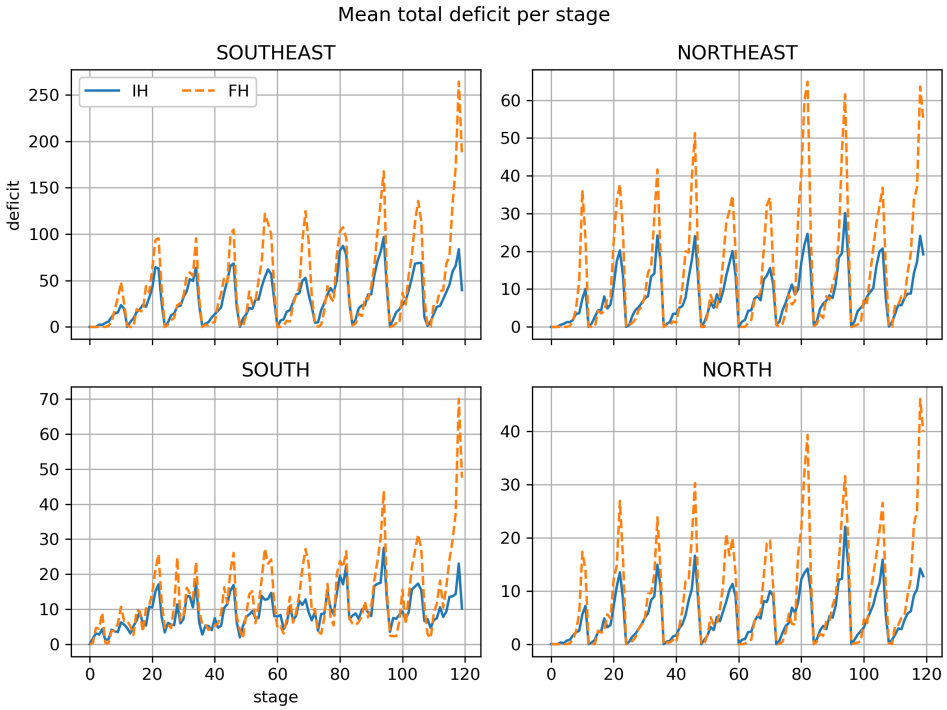

Figure 4.7: Mean total deficit, divided by region, for models FH (in orange, dashed) and IH (in blue, solid).

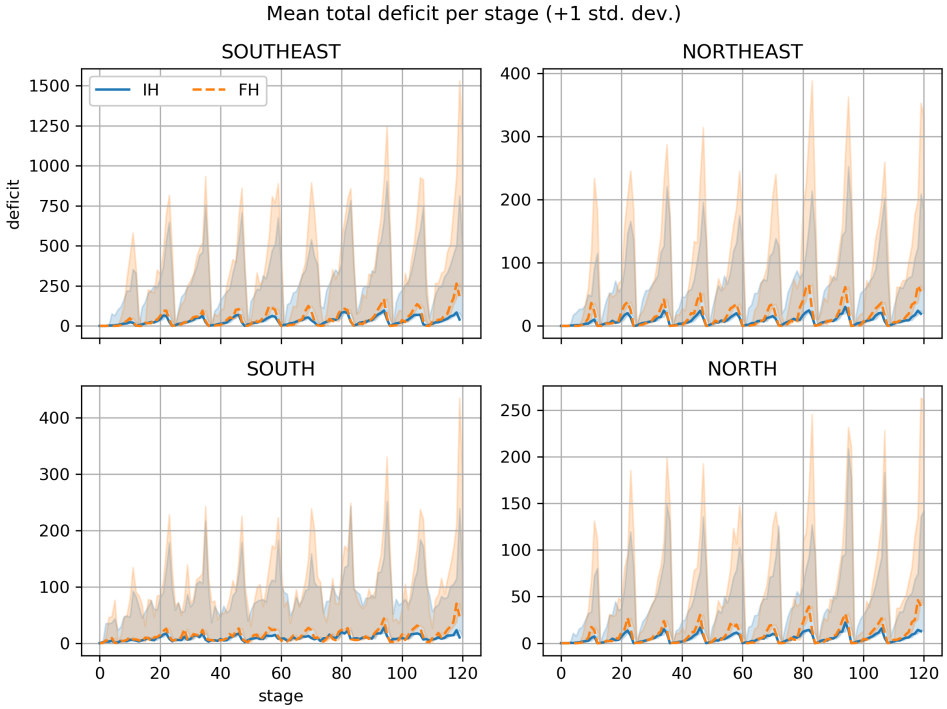

Figure 4.8: Mean total deficit, divided by region, for models FH (in orange, dashed) and IH (in blue, solid). The area between the mean and one standard deviation above it is shaded in the respective color.

4.3.2 Simple counterexamples to natural conjectures

In the hydrothermal example, the average value of the state variable exhibited periodical behaviour. It would be possible for the structure of a periodical infinite horizon problem to imply certain properties of its solution. The examples in the remainder of this section were pursued in an attempt to answer such questions. In order to formulate our questions precisely, we require the following terminology.

Definition 4.3.1. An *optimal trajectory* or *optimal orbit* is a sequence $\{x_t^\star\}_{t\in\mathbb{N}}$ of optimal solutions of an infinite horizon problem.

We begin with the two following questions.

Question 4.3.2. Does the period of the limit of an optimal trajectory match the period of the model?

Answer. No. See examples 4.3.4, 4.3.5, 4.3.9, 4.3.10, and 4.3.11. We highlight that lemma 4.3.6 and the discussion following it analyze example 4.3.5.

In particular, for stationary models, question 4.3.2 is equivalent to asking if the optimal trajectory converges to a fixed point.

One can also wonder:

Question 4.3.3. Is the optimal value function nonincreasing along an optimal trajectory?

Answer. No. See examples 4.3.5, 4.3.9, and 4.3.10.

We introduce a simple example that serves as a starting point for building more interesting problems.

Example 4.3.4 (Deterministic uncontrolled 1D symmetric linear model)**.**

$$V^\star(x_0) = \begin{cases} \min\limits_{x} & |x| + \frac{1}{2}V^\star(-x) \\ \text{s.t.} & x = x_0. \end{cases}$$

This model has a state variable $x \in \mathbb{R}$, and no control variables. The present cost is given by $c(x) = |x|$, and the transition function is $f(x) = -x$, a reflection through the origin. There is a single constraint, $x = x_0$. Note that this problem is stationary.

The optimal solution to the problem 4.3.4 lacks the two properties of interest from the preceding questions. First, with the exception of the trajectory that begins at $x_0 = 0$, all trajectories are 2-periodic. Second, the optimal value function along each trajectory is constant. However, in informal terms, the preceding example is a degenerate optimization problem. There are no decisions to be made: its behaviour is deterministic and a straightforward computation.

Model 4.3.4 can be modified to be less degenerate by adding a control variable u, and changing the transition function to $f(x, u) = -x + u$, which allows the controller to move between states. Additionally, we bound the control u by adding the constraint $|u| \leq 0.1$. Another modification to make the model more interesting is to make the present cost asymmetric with respect to the origin, by changing it to $c(x) = |x - 0.5|$.

Example 4.3.5 (Deterministic 1D linear model)**.**

$$V^\star(x_0) = \begin{cases} \min\limits_{x,u} & |x - 0.5| + \frac{1}{2}V^\star(-x + u) \\ \text{s.t.} & x = x_0, \\ & |u| \leq 0.1. \end{cases}$$

For an initial condition outside an interval around the origin, $x_0 \notin (-0.4, 0.5)$, the trajectory tends to the boundary of that interval, *i.e.* $\{-0.4, 0.5\}$. With exception of $x_0 = 0.05$, initial conditions inside the interval $(-0.4, 0.5)$ follow along 2-periodic orbits, alternating between the original state x_0 and another state. Figure 4.9 illustrates the behaviour of optimal trajectories $\{x_t^\star\}_{t\in\mathbb{N}}$ and of the optimal value function $\{V^\star(x_t^\star)\}_{t\in\mathbb{N}}$ along them.

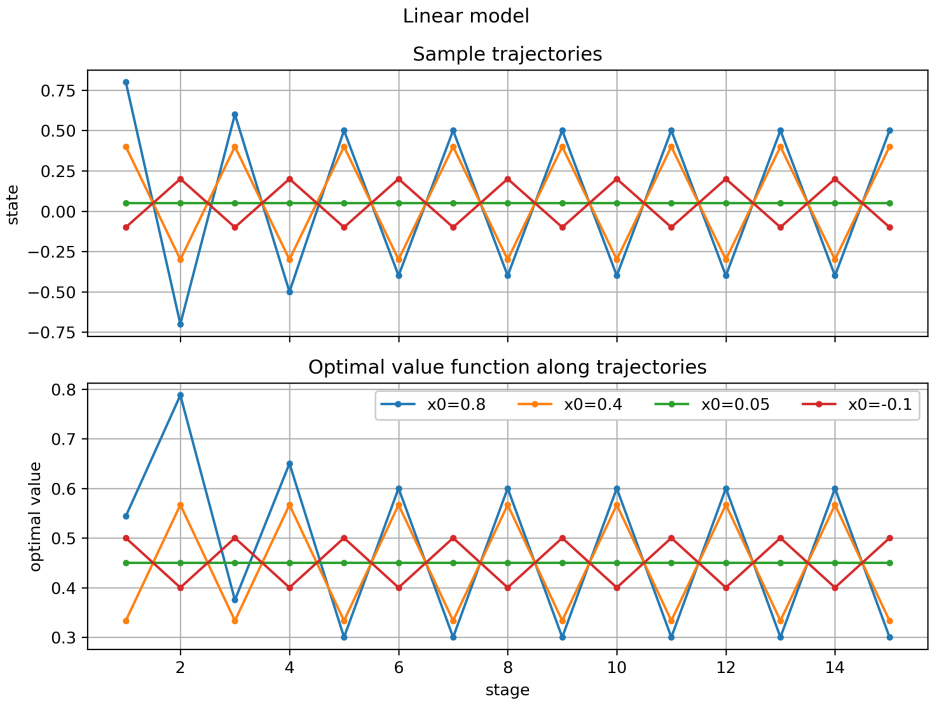

Figure 4.9: Simulation results for the deterministic 1D linear model. On the top, sample trajectories in different colors. On the bottom, the respective evaluation of the optimal value function.

This example showcases two properties. The first property is that not all initial conditions converge to the same limiting solution. In particular, there

are optimal solutions that do not converge to either the global minimum of the optimal value function or to a fixed point. Note that this model is stationary: the objective function, the transition function, and the constraints are the same at each stage. Hence the period of almost every optimal solution does not match the period of the model. The second property is that the optimal value function can indeed increase along an optimal trajectory.

We are going to verify analytically that the optimal value function of example 4.3.5 satisfies the aforementioned properties, but, for simplicity, we restrict the domain of the problem. As long as this domain restriction does not affect the optimization procedure, we obtain the same result. Prior to specifying what we mean and proving the lemma that shows the subsequent analysis is sufficient, we need some definitions.

Denote by $Z \subset X$ the restricted domain of interest, and assume it is convex, compact, and non-empty. Naturally, we want to look at states $x_0 \in Z$. We would like that each subsequent state also belong to Z, and we can achieve this by modifying the original infinite horizon problem. In order to keep our previous structure, we must avoid constraining the state directly, so we augment the original problem with a control variable z, and add constraints on z. Our modified Wald-Bellman operator is

$$B_Z(V)(x_0) = \begin{cases} \min\limits_{x,u,z} & c\left(u\right) + \beta V\left(x\right) \\ \text{s.t.} & x = f(x_0, u), \\ & u \in U\left(x_0\right), \\ & z \in Z, \\ & z = x. \end{cases}$$

To conform to our usual notation, the constraints on the control variables should be replaced by $(u, z) \in W(x_0)$, where the set-valued function W is given by $W(x_0) = \{(u, z) : u \in U(x_0), z \in Z(f(x_0, u))\}$, where Z is in turn described by $Z(x) = \{z : z \in Z, z = x\}$. Note we are only interested in states x_0 for which $Z(f(x_0, u))$ is non-empty.

Out of our remaining assumptions in remark 4.1.6, we only need to verify that

$$\{(x, u, z) : x \in X, u \in U(x), z \in Z(f(x, u))\}$$

is a convex compact set. However, this is an intersection of closed convex sets contained within the compact $X \times \mathcal{U} \times \mathcal{Z}$, where $\mathcal{Z} = \bigcup_{x \in X} Z(x)$, showing those properties hold.

We can finally assert the existence of $V_Z^\star = B_Z(V_Z^\star)$, the optimal value function for the modified Wald-Bellman operator. Our goal now is to show that $V_Z^\star|_Z = V^\star|_Z$ under the following assumption: for each $x_0 \in Z$, there exists $S(x_0) \subset Z$, open relative to Z, with $z^\star = f(x_0, u^\star) \in S(x_0)$, where $z^\star$ is

an optimal solution of the variable z for $V_Z^\star(x_0)$. We refer to this assumption by saying the constraint $z \in Z$ is inactive for every $x_0 \in Z$. Now we achieve this goal by iterating the original operator B on $V_Z^\star$.

Lemma 4.3.6. *Let $Z \subset X$ be defined as before. Assume that for every $x_0 \in Z$, the constraint $z \in Z$ is inactive in the modified problem. Then $V_Z^\star|_Z = V^\star|_Z$.*

Proof. Whenever $x_0 \in Z$, denote by

$$R(x_0) = \{u \in U(x_0) \,:\, f(x_0, u) \in S\};$$

importantly, the set $R(x_0)$ is open relative to $U(x_0)$ as the inverse image of $S(x_0)$ by a continuous function.

For every $u \in R(x_0)$ and any $V \in L^\infty(X)$ we can write

$$c(u) + \beta V \circ f(x_0, u) + \mathbb{I}\,[f(x_0, u) \in Z] = c(u) + \beta V \circ f(x_0, u).$$

Note that if we minimize over $u \in U(x_0)$, the left-hand and right-hand side are equivalent to the expressions inside B_Z and B respectively. Instead, we can minimize both sides of this equation over $u \in R(x_0)$. Whenever this minimum is attained, we can conclude it is also the minimum over $U(x_0)$, as $R(x_0)$ is open relative to $U(x_0)$ and this is a convex minimization problem (by assumption).

In particular, setting $V = V_Z^\star$, the right-hand side yields

$$\min_{u \in R(x_0)} c(u) + \beta V_Z^\star \circ f(x_0, u) = \min_{u \in U(x_0)} c(u) + \beta V_Z^\star \circ f(x_0, u) = B(V_Z^\star)(x_0),$$

where the first equality was justified by the previous paragraph. Similarly, for the left-hand side,

$$\min_{u \in R(x_0)} c(u) + \beta V_Z^\star \circ f(x_0, u) + \mathbb{I}\,[f(x_0, u) \in Z] = B_Z(V_Z^\star)(x_0) = V_Z^\star(x_0).$$

We have shown the base case: $V_Z^\star(x_0) = B(V_Z^\star)(x_0)$ holds for every $x_0 \in Z$. Now, assume that $V_Z^\star(x_0) = B^k(V_Z^\star)(x_0)$ for every $x_0 \in Z$ and some $k \in \mathbb{N}$. Then for every $u \in R(x_0)$, $f(x_0, u) \in Z$ and thus

$$c(u) + \beta B^k(V_Z^\star) \circ f(x_0, u) = c(u) + \beta V_Z^\star \circ f(x_0, u).$$

We know that the minimum over $u \in R(x_0)$ is attained on the right-hand side, and therefore also on the left-hand side. As before, we have

$$\min_{u \in R(x_0)} c(u) + \beta B^k(V_Z^\star) \circ f(x_0, u) = \min_{u \in U(x_0)} c(u) + \beta B^k(V_Z^\star) \circ f(x_0, u) = B^{k+1}(V_Z^\star)(x_0).$$

This completes the induction step and we may now take the limit as $k \to \infty$ to show that $V_Z^\star|_Z = V^\star|_Z$. □

We can finally proceed with our analysis of example 4.3.5, which is restricted to the interval $[-0.6, 0.6]$. In this interval, the optimal value function is given by

$$V^\star(x) = \max\left\{1.5x - 0.15, -\frac{2}{3}x + \frac{14}{15}, -x + 0.8\right\}.$$

Indeed, we can verify this by checking the Bellman iteration: let $x \in [-0.6, 0.6]$, then we can compute

$$B(V^\star)(x) = \min_{|u|\leq 0.1} |x - 0.5| + \frac{1}{2}V^\star(-x + u) = |x - 0.5| + \frac{1}{2}\min_{|u|\leq 0.1} V^\star(-x + u)$$

by dividing it into three cases.

Case 1. For $x \in [-0.6, -0.4]$, the solution is $u = x + 0.5$, as the global minimizer of $V^\star$ is 0.5. Thus $V^\star(-x + u) = V^\star(0.5) = 0.6$, and

$$B(V^\star)(x) = |x - 0.5| + 0.3 = -x + 0.8 = V^\star(x).$$

Case 2. For $x \in [-0.4, 0.5]$, we use that $-x \in [-0.5, 0.4] \subset [-0.6, 0.5]$, where $V^\star$ is strictly decreasing. The solution is $u = 0.1$, and $V^\star(-x + 0.1) = -\frac{2}{3}(-x + 0.1) + \frac{14}{15} = \frac{2}{3}x + \frac{13}{15}$, and

$$B(V^\star)(x) = |x - 0.5| + \frac{1}{3}x + \frac{13}{30} = -\frac{2}{3}x + \frac{14}{15} = V^\star(x).$$

Case 3. For $x \in [0.5, 0.6]$, it is similar to the previous case: the solution is $u = 0.1$, and $V^\star(-x + 0.1) = -(-x + 0.1) + 0.8 = x + 0.7$. Bellman's iteration results in

$$B(V^\star)(x) = |x - 0.5| + \frac{1}{2}x + 0.35 = 1.5x - 0.15 = V^\star(x).$$

The optimal solution for $x \in [-0.6, 0.6]$ is $u^\star(x) = \min\{0.1, x + 0.5\}$. The optimal transition function is

$$\phi^\star(x) = f(x, u^\star(x)) = -x + u^\star(x) = \min\{-x + 0.1, 0.5\}.$$

It allows us to verify the existence of multiple cycles of period 2 by looking at the orbits $\{\varphi^{\star n}(x_0)\}_{n\in\mathbb{N}}$ of specific points:

$n = 0$	1	2	3	4
0.6	−0.5	0.5	−0.4	0.5
0.4	−0.3	0.4	−0.3	0.4
0.05	0.05	0.05	0.05	0.05
−0.1	0.2	−0.1	0.2	−0.1

After the preceding example, one may conjecture that models with a present cost function that is strongly convex in both state *and* control variables would exhibit a single limit optimal trajectory. Framed as a question:

Question 4.3.7. Does the period of the limit of an optimal trajectory match the period of a model with *strongly convex immediate cost*?

Answer. No. See examples 4.3.9 and 4.3.10.

Furthermore, one may ask:

Question 4.3.8. Does a strongly convex immediate cost guarantee an unique limit that is independent of initial conditions?

Answer. No. See examples 4.3.9 and 4.3.10.

We modify the present cost to $c(x) = |x - 0.5|^2 + |u|^2$, and showcase some simulation results in figure 4.10. Note that the sample trajectories begin at different states than the linear example.

Example 4.3.9 (Deterministic 1D strongly convex model)**.**

$$V^\star(x_0) = \begin{cases} \min\limits_{x,u} & |x - 0.5|^2 + |u|^2 + \frac{1}{2}V^\star(-x + u) \\ \text{s.t.} & x = x_0, \\ & |u| \leq 0.1. \end{cases}$$

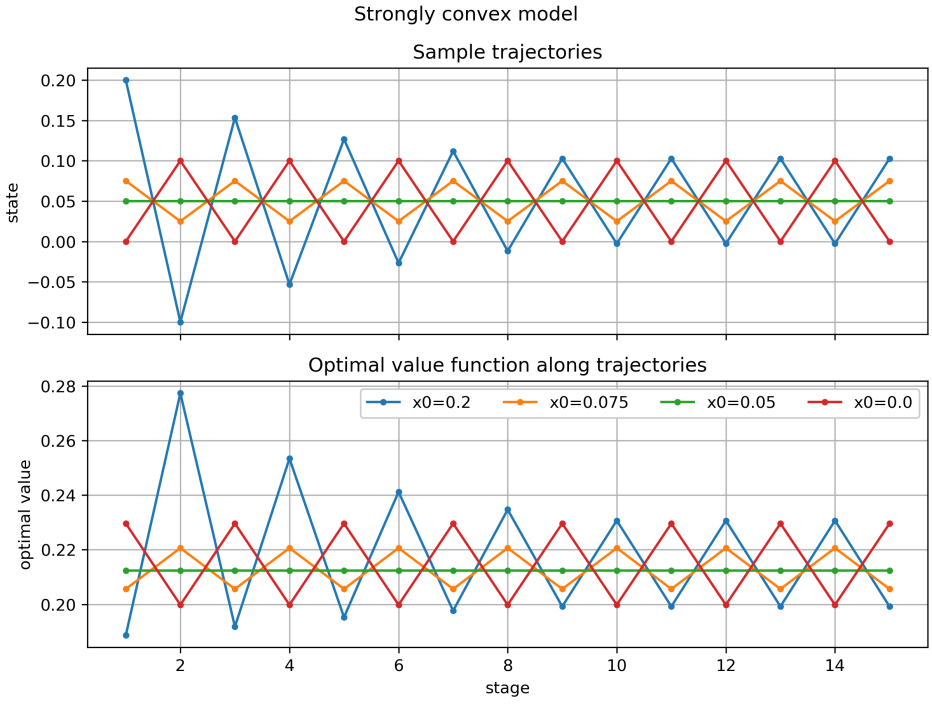

Figure 4.10: Simulation results for a deterministic 1D strongly convex model. On the top, sample trajectories in different colors. On the bottom, the respective evaluation of the optimal value function.

Another simple modification yields another strongly convex model whose trajectories still do not converge to an unique limit. Rather than evaluating the present cost at the initial state, $c(x_0, u)$, which we cannot influence, a possible modification is to consider a present cost as a function of the subsequent state, $c(f(x_0, u), u)$. As the transition function f is linear, this preceding composition is convex. Substituting their definitions,

$$c(f(x,u),u) = |f(x,u) - 0.5|^2 + |u|^2 = |-x + u - 0.5|^2 + |u|^2 = \\ x^2 - 2ux + 2u^2 + x - u + 0.25, \quad (4.11)$$

is a strongly convex function of x and u, whose Hessian $\left(\begin{smallmatrix} 4 & -2 \\ -2 & 2 \end{smallmatrix}\right)$ has eigenvalues $3 \pm \sqrt{5} > 0$.

Example 4.3.10 (Deterministic 1D strongly convex model with alternative cost)**.**

$$V^\star(x_0) = \begin{cases} \min\limits_{x,u} & |-x + u - 0.5|^2 + |u|^2 + \frac{1}{2} V^\star(-x + u) \\ \text{s.t.} & x = x_0, \\ & |u| \leq 0.1. \end{cases}$$

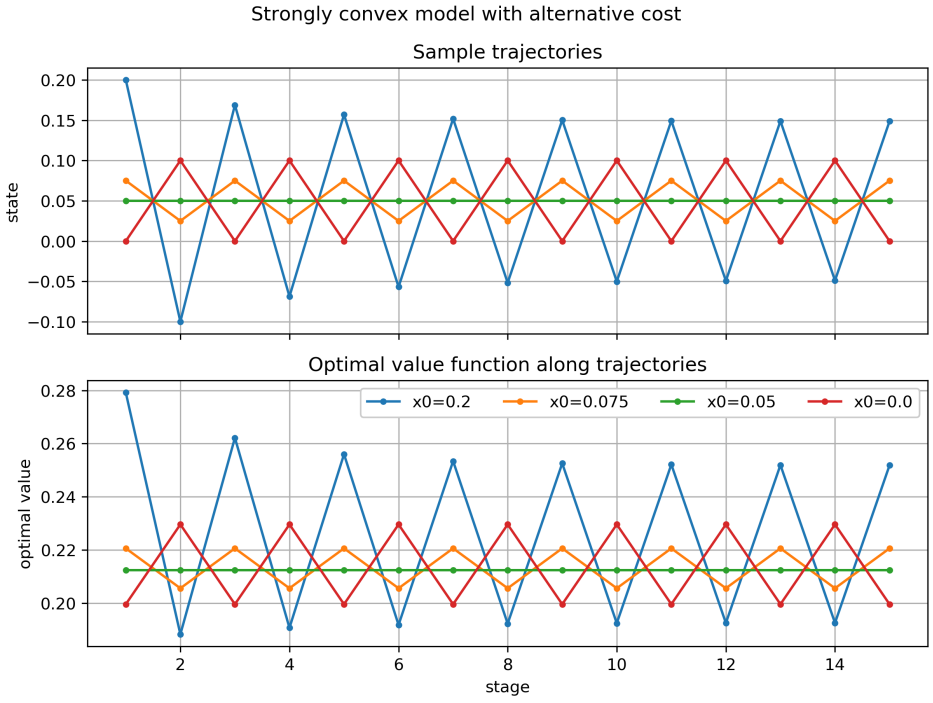

Figure 4.11: Simulation results for the alternative deterministic 1D strongly convex model. On the top, sample trajectories in different colors. On the bottom, the respective evaluation of the optimal value function.

We have then seen two examples of stationary infinite horizon models that suggest a strongly convex present cost is not enough to: guarantee stationary optimal trajectories; and an unique limit independent of initial conditions.

Finally, we can construct a two-dimensional analogue of the one-dimensional examples seen so far. In the linear model, the absolute value function in the present cost function is replaced with the ℓ_1-norm. The upper bound constraint on the absolute value of the control is replaced with an upper bound constraint on the ℓ_∞-norm of the control. The most noteworthy modification is, however, the change to the transition function. Rather than a reflection through the origin, given by $-I$, we choose to use a rotation of θ radians, given by R_θ.

Example 4.3.11 (Deterministic 2D linear model)**.**

$$V^\star(x_0) = \begin{cases} \min\limits_{x,u} & \|x - c_0\|_1 + 0.99V^\star(R_\theta x + u) \\ \text{s.t.} & x = x_0, \\ & \|u\|_\infty \leq \overline{u}. \end{cases}$$

The optimal trajectories of this model exhibit different behaviours for different values of $\overline{u}$. This is illustrated in figure 4.12 with trajectories starting from $x_0 = (0.75, -0.25)$ and $\theta = \frac{\pi}{36}$. A large value of $\overline{u}$ results in convergence to a fixed point. As $\overline{u}$ diminishes, the limiting fixed point changes, and eventually there is a phase transition: rather than converging to a limit, the optimal trajectory seems to approach a closed curve. It is unclear if this trajectory is at all periodic as previously defined in this work. If the answer is negative, perhaps it could be possible to define periodicity in a continuous sense and to characterize this trajectory as such.

I.e. 5 degrees.

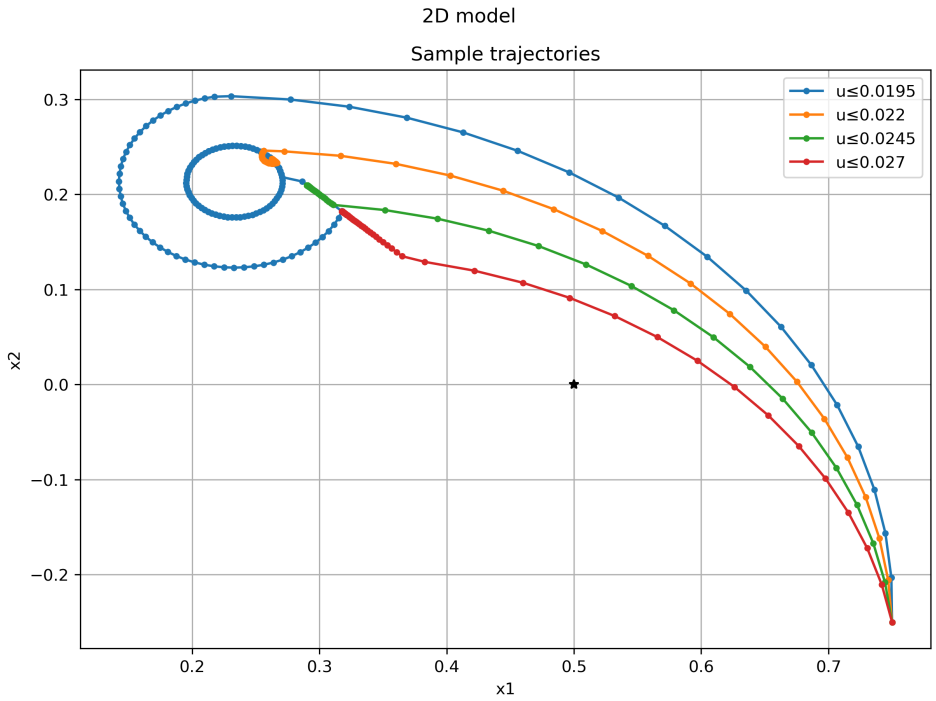

Figure 4.12: Optimal trajectories for the same initial condition but various value of the parameter $\overline{u}$ in different colors. The star symbol marks the global minimum of the present cost function.

4.4 Final remarks

In this section, we discuss possible improvements or lines of study.

The stochastic model for the hydrothermal problem was based on historical data. The realization at each stage corresponded with the rainfall of each region in the respective month of a particular year. In this manner, we aimed for a model of weather patterns that is consistent country-wide, however independent from each other across time. Moreover, model training and simulation were based on the same data, although the specific scenarios at each step may differ. Both aspects could be improved by developing an autoregressive-moving-average (ARMA) model. This particular kind of stochastic model can be incorporated into a multistage optimization model while preserving the stagewise independency property via the addition of auxiliary state variables. By fitting ARMA models to different data (perhaps subsets of the historical rainfall time series), the training results could be cross-validated on out-of-sample realizations.

Despite the counterexamples to the questions asked in this section, general properties of infinite horizon problems could be obtained under further regularity conditions. By their very nature, such characterizations would be useful to more detailed analysis and the development of alternative algorithms. Example 4.3.11 suggests the behaviour of optimal trajectories can change alongside parameters of the model. A deeper inquiry into models exhibiting this quality—perhaps under the optics of dynamical systems—could aid in identifying what conditions are necessary for certain properties to hold. Despite our focus on discrete-time problems, shifting to continuous-time infinite horizon might facilitate the analysis.

Among the model parameters, β stands out as the contraction property of the Wald-Bellman operator necessitates $\beta < 1$. Investigating the effects of the limiting operation $\beta \to 1$ could be interesting, *i.e.* moving from discounted to undiscounted problems. Setting $\beta = 1$ can be interpreted as being indifferent to when costs are incurred, as opposed to having a preference for delaying costs as much as possible. The main issue is that the objective value becomes unbounded unless costs are zero for infinitely many stages. A remedy could be to multiply to objective value by $1-\beta$, as multiplication by a positive constant does not alter the solution of an optimization problem. Under assumptions given in this chapter, the objective can be bounded below by 0 and above by $\frac{\overline{c}}{1-\beta}$, suggesting that $\lim_{\beta\to 1^-}(1-\beta)V_\beta^\star(x_0) \in [0, \overline{c}]$. A different but related approach would be to consider the optimal transition functions $\phi_\beta(x_0) = f(x_0, u_\beta^\star)$. Although unecessary for the purposes of this dissertation, they could prove useful if they were stable under the limit. That is to say, even for problems where $\lim_{\beta\to 1^-}(1-\beta)V_\beta^\star(x_0) = 0$ for every $x_0 \in X$, if $\lim_{\beta\to 1^-}\phi_\beta(x_0) = \phi_1(x_0)$ is well-defined, that might be sufficient as a solution of the undiscounted

problem, even if the limiting value function is everywhere zero.

Bibliography

[1] Richard Bellman. "On the theory of dynamic programming". In: *Proceedings of the National Academy of Sciences of the United States of America* 38.8 (1952), p. 716.

[2] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2019.

[3] Dimitri Bertsekas. "Convergence of discretization procedures in dynamic programming". In: *IEEE Transactions on Automatic Control* 20.3 (1975), pp. 415–419.

[4] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. 3rd ed. Vol. 1. Athena scientific, 2005.

[5] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume II*. 3rd ed. Vol. 2. Athena scientific, 2005.

[6] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA, 1997.

[7] John R. Birge. "Decomposition and partitioning methods for multistage stochastic linear programs". In: *Operations Research* 33.5 (1985), pp. 989–1007.

[8] John R. Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[9] John R. Birge and Gongyun Zhao. "Successive linear approximation solution of infinite-horizon dynamic stochastic programs". In: *SIAM Journal on Optimization* 18.4 (2008), pp. 1165–1186.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Vitor L. De Matos, Andy B. Philpott, and Erlon C. Finardi. "Improving the performance of stochastic dual dynamic programming". In: *Journal of Computational and Applied Mathematics* 290 (2015), pp. 196–208.

[12] O. Dowson. "The policy graph decomposition of multistage stochastic optimization problems". In: *Networks* 76 (1 2020), pp. 3–23. DOI: `https://doi.org/10.1002/net.21932`.

[13] O. Dowson and L. Kapelevich. "SDDP.jl: a Julia package for stochastic dual dynamic programming". In: *INFORMS Journal on Computing* 33 (1 2021), pp. 27–33. DOI: https://doi.org/10.1287/ijoc.2020.0987.

[14] P. Girardeau, V. Leclere, and A. B. Philpott. "On the convergence of decomposition methods for multistage stochastic convex programs". In: *Mathematics of Operations Research* 40.1 (2015), pp. 130–145.

[15] Vincent Guigues. "Convergence analysis of sampling-based decomposition methods for risk-averse multistage stochastic convex programs". In: *SIAM Journal on Optimization* 26.4 (2016), pp. 2468–2494.

[16] Vincent Guigues. "Dual dynamic programing with cut selection: Convergence proof and numerical experiments". In: *European Journal of Operational Research* 258.1 (2017), pp. 47–57.

[17] Alexey Izmailov and Mikhail Solodov. *Otimização, volume 1: condições de otimalidade, elementos de análise convexa e de dualidade*. Impa, 2005.

[18] James E. Kelley Jr. "The cutting-plane method for solving convex programs". In: *Journal of the society for Industrial and Applied Mathematics* 8.4 (1960), pp. 703–712.

[19] Lisa A. Korf. "Approximating infinite horizon stochastic optimal control in discrete time with constraints". In: *Annals of Operations Research* 142.1 (2006), pp. 165–186.

[20] Karsten Linowsky and Andrew B. Philpott. "On the convergence of sampling-based decomposition algorithms for multistage stochastic programs". In: *Journal of Optimization Theory and Applications* 125.2 (2005), pp. 349–366.

[21] Giacomo Nannicini, Emiliano Traversi, and Roberto Wolfler Calvo. "A Benders squared (B2) framework for infinite-horizon stochastic linear programs". In: *Mathematical Programming Computation* (2020), pp. 1–37.

[22] George L Nemhauser and Laurence A Wolsey. *Integer and combinatorial optimization*. Vol. 18. Wiley New York, 1988.

[23] Mario V. F. Pereira and Leontina M. V. G. Pinto. "Multi-stage stochastic optimization applied to energy planning". In: *Mathematical programming* 52.1 (1991), pp. 359–375.

[24] R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational analysis*. Vol. 317. Springer Science & Business Media, 2009.

[25] Alexander Shapiro and Yi Cheng. "Dual Bounds for Periodical Stochastic Programs". In: (2020).

[26] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[27] Alexander Shapiro and Lingquan Ding. "Periodical multistage stochastic programs". In: *SIAM Journal on Optimization* 30.3 (2020), pp. 2083–2102.

[28] Alexander Shapiro et al. "Risk neutral and risk averse stochastic dual dynamic programming method". In: *European Journal of Operational Research* 224.2 (2013), pp. 375–391.

[29] Richard M. Van Slyke and Roger 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.

[30] Joseph Warrington, Paul N. Beuchat, and John Lygeros. "Generalized dual dynamic programming for infinite horizon problems in continuous state and action spaces". In: *IEEE Transactions on Automatic Control* 64.12 (2019), pp. 5012–5023.

Appendix

A.1 Algorithms

We present well-known methods for solving the optimization problems presented in chapter 2.

A.1.1 General convex problems

As we are going to see, the solution process of an optimization problem with complicated structure can be reduced to solving multiple problems with a simpler structure (*e.g.* the Barrier method in definition A.1.4). The same can be said about the problems discussed in other chapters: a key topic in Finite horizon is how to decompose a multistage problem into "one-stage" problems, *i.e.* the kind of problem the current section explains how to solve.

A simple, well-known optimization algorithm is *gradient descent*. This method attempts to build a sequence $\{x_n\}_{n\in\mathbb{N}}$ such that the iterates $f(x_n)$ converge to the minimum of the function f. This sequence is called a *minimizing sequence*, and it takes the form

$$x_{n+1} = x_n + t_n \Delta x_n,$$

where, for gradient descent, $\Delta x_n = -\nabla f(x_n)$, and t_n is chosen at each iteration via a process known as *line search*. Gradient descent requires an unconstrained minimization problem

$$\min_x f(x),$$

an initial feasible solution x_0, and a differentiable objective f. Proofs of convergence typically further require a twice-differentiable objective f, strong convexity, and the existence of an optimal solution. Gradient descent belongs to a larger class of optimization methods: *descent methods*.

For this problem, feasibility means $f(x_0) < \infty$.

Definition A.1.1 (Descent method)**.** A *descent method* is an algorithm for constructing a sequence $\{x_n\}_{n\in\mathbb{N}}$ such that

$$f(x_{n+1}) < f(x_n),$$

except whenever $x_n = x^\star$ is an optimal solution. This sequence is produced iteratively by the equation

$$x_{n+1} = x_n + t_n \Delta x_n,$$

where t_n is called *step size* or *step length*, and Δx_n the *step* or *search direction*.

The main difference among descent methods is the process of choosing the search direction. The step sizes are chosen by *line search*, which can be *exact* or *inexact*. Exact line search requires solving the following minimization problem in one variable:

$$\min_{t \geq 0} f(x_n + t\Delta x_n).$$

A simple option for inexact line search is *backtracking*: begin with a unit step size $t_{n_0} = 1$, which is reduced to $t_{n_{k+1}} = \beta t_{n_k}$ where $\beta \in (0, 1)$ until $f(x_n + t_{n_k}\Delta x_n)$ decreases a sufficient amount when compared to a linear extrapolation of f at x_n. Despite line search being a constrained optimization problem, it is easier to solve as it is an optimization problem in a single variable. Section 9.2 of [10] describes general descent methods, alongside the two kinds of line search here mentioned. Moreover, section 9.3 covers gradient descent.

A typical stopping criterion for descent methods is of the form $\|\nabla f(x)\| < \varepsilon$. This is because when f is strongly convex with parameter m, $f(x)$ is at most $\frac{1}{2m}\|\nabla f(x)\|^2$ away from the true minimum. (Cf. subsection 9.1.2 of [10].) A generic descent method is outlined below.

Algorithm A.3: Descent method.

Data: an objective function f and an initial feasible point x.
Result: a suboptimal solution.

```
while stopping criterion not satisfied do
    calculate descent direction Δx      // Method dependent.
    calculate step size t               // Line search.
    set x ← x + tΔx
end
```

We now discuss a specific descent method that can be adapted to constrained optimization problems.

Definition A.1.2 (Newton's method)**.** *Newton's method* is a descent method where the search direction Δx_n is given by

$$\Delta x_n = -\nabla^2 f(x_n)^{-1} \nabla f(x_n).$$

A quantity of interest is the *Newton decrement* at x, given by

$$\lambda(x) = \sqrt{\nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x)}.$$

The Newton decrement has multiple interpretations, one of which is the directional derivative of f at x in the direction of Δx:

$$-\lambda(x)^2 = \nabla f(x)^\top \Delta x.$$

It can be used to define a stopping criterion, and used to assess the convergence rate of the method (cf. subsection 9.5.3 of [10]; a sharper estimate holds for self-concordant strictly convex functions, and is shown in subsection 9.6.3).

Optimizing a constrained optimization problem requires different methods. A common feature of such algorithms is the distinction between two stages during their execution. If no initial feasible solution is provided, the method either finds a feasible solution or shows no such solution can exist: this stage is called *phase I*. Once an initial feasible solution is available, the method attempts to find an optimal solution: this stage is called *phase II*. We now discuss how to modify Newton's method for equality constrained optimization problems.

Definition A.1.3 (Newton's method revisited)**.** The modification for a phase II Newton's method is straightforward. Assume the problem is of the form

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & Ax = b. \end{aligned}$$

Then the modification is to calculate search direction as the solution to the following system of equations

$$\begin{pmatrix} \nabla^2 f(x) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \mu \end{pmatrix} = - \begin{pmatrix} \nabla f(x) \\ 0 \end{pmatrix}.$$

Now, for the phase I method, assume $f(x) < \infty$, but not that x is feasible. Then we solve

$$\begin{pmatrix} \nabla^2 f(x) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \mu \end{pmatrix} = - \begin{pmatrix} \nabla f(x) \\ Ax - b \end{pmatrix}.$$

Note that when x is feasible, $Ax - b = 0$, and phase I reduces to phase II.

This system of equations arises from the Karush-Kuhn-Tucker (KKT) conditions, which under differentiability assumptions are satisfied by the optimal solutions of a primal-dual pair of optimization problems. Here, this system of equations can be interpreted as the KKT conditions for the problem of minimizing the second-order approximation of f at x subject to $Ax = b$. For this reason, μ can be thought of as a dual variable, and this problem as a primal-dual method. Despite their importance for optimization, we do not explicitly use the KKT conditions in the remainder of this dissertation, so we refer to [10]: subsection 5.5.3 covers the KKT conditions specifically, while chapter 10 presents algorithms for equality constrained optimization problems (including Newton's method in section 10.2).

A technique for optimizing a constrained optimization problem with inequality constraints is the *barrier method*.

Definition A.1.4 (Barrier method)**.** The main idea behind the *barrier method* is to approximate the inequality constrained optimization problem via a sequence of optimization problems without inequality constraints (but possibly with equality constraints). This approximation is done by first incorporating the inequality constraints into the objective via indicator functions,

$$\begin{array}{ll} \min\limits_{x} & f(x) + \sum_{i=1}^{m} \mathbb{I}[f_i(x) \leq 0] \\ \text{s.t.} & Ax = b, \end{array}$$

and then approximating each indicator function with

$$\frac{1}{t}\phi_i(x) = -\frac{1}{t}\log(-f_i(x)),$$

where $t > 0$ is a parameter. These functions are called *logarithmic barrier functions*. Denoting $\phi(x) = \sum_{i=1}^{m} \phi_i(x)$, the sequence of problems can be described by

$$\begin{array}{ll} \min\limits_{x} & tf(x) + \phi(x) \\ \text{s.t.} & Ax = b. \end{array} \tag{P(t)}$$

Algorithm A.4: Barrier method.

Data: an objective function f, the logarithmic barrier functions $\{\phi_i\}_{i=1}^{m}$, an initial feasible point x, and tolerance $\varepsilon > 0$.
Result: a suboptimal solution.

```
while m/t ≥ ε do
    calculate x* by solving P(t)        // Centering step.
    set t ← γt
end
```

The centering step has its name as the optimal solution $x^\star(t)$ of $P(t)$ is called a *central point*.

Confer chapter 11 of [10] for a discussion of methods for constrained optimization problems, including the barrier method.

A.1.2 Linear and mixed-integer problems

The solution techniques we have discussed so far can be applied to a broad range of convex optimization problems. By focusing on a smaller class of problems, their structure can be exploited to develop more efficient algorithms. We now look at an algorithm for solving linear programs, and then at a method for solving integer and mixed-integer programs. The latter is often implemented alongside linear approximations of the original problem via continuous relaxations of the integer.

Assume the LP is in standard form,

$$\begin{array}{rl} \min\limits_{x\in\mathbb{R}^n} & c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0. \end{array} \qquad \text{(2.5 revisited)}$$

We further assume that all redundant equality constraints have been removed, and thus that the $m \times n$ matrix A has full rank m, where $m < n$. The dual problem can then be written as

$$\begin{array}{rl} \min\limits_{\mu\in\mathbb{R}^m} & -\mu^\top b \\ \text{s.t.} & \mu^\top A \geq -c^\top. \end{array} \qquad (12)$$

Definition A.1.5 (Simplex method)**.** An overview of the simplex method is that it operates by examining a feasible solution with specific properties, showing that: the examined solution is optimal; that the problem is unbounded below; or finds another feasible solution with the same desired properties, which is then examined.

A linear constraint is called *active at* x when it holds with equality, *e.g.* the constraint

$$x_1 + x_2 \leq 1$$

is active at $(x_1, x_2) = (1, 0)$ and at $(x_1, x_2) = (0.5, 0.5)$, but not at $(x_1, x_2) = (0, 0)$. A vector $x \in \mathbb{R}^n$ is called a *basic solution* if it satisfies all equality constraints *and* if there are n linearly independent linear operators among the constraints active at x.

Note that a satisfied equality constraint is active.

Under our assumptions, $\operatorname{rank} A = m$, and thus there is a set of m indexes $I \subset [1..n]$ such that the columns of A indexed by I form a nonsingular $m \times m$ matrix, denoted by A_I. The matrix A_I is called a *basis*. We denote by $J = [1..n] \setminus I$ the complement of I. A vector with m linearly independent active constraints can be obtained by calculating $x^b = A_I^{-1} b$. Setting $x_i = x_i^b$ for each $i \in I$, we can get a basic solution by fixing the remaining variables as $x_j = 0$ for each $j \in J$: this results in $|J| = n - m$ active inequality constraints plus the $|I| = m$ active equality constraints; furthermore, they form a set of n linearly independent constraints. We can reorder the problem's variables such that this basic solution can be written as

We do not assume $x_i^b \geq 0$.

$$x = (x^b, x^n) = (A_I^{-1} b, 0).$$

The variables of x^b are called *basic variables*, while the variables of x^n are the *nonbasic variables*. When $x^b \geq 0$, x is feasible, and thus called a *basic primal feasible solution*, while A_I is a *primal feasible basis*.

Denote by A_J the $m \times (n - m)$ matrix such that, after reordering, $A = (A_I, A_J)$. Similarly, denote by $c = (c_I, c_J)$ the components of the objective

associated with the basic and nonbasic variables respectively. We define $\mu^\top = -c_I A_I^{-1}$, and thus

$$\mu^\top A + c^\top = \mu^\top (A_I, A_J) + (c_I, c_J) = (-c_I, -c_I A_I^{-1} A_J) + (c_I, c_J) = (0, -c_I A_I^{-1} A_J + c_J),$$

showing: that x and $\mu^\top$ are complementary; and that $\mu^\top$ is dual feasible if $-c_I A_I^{-1} A_J \geq -c_J$. Whenever this last inequality holds, A_I is called a *dual feasible basis*.

If a basis A_I is both primal and dual feasible, then x is an optimal solution to the primal problem, and μ is an optimal solution to the dual problem. To see this, note that

$$c^\top x = c_I^\top A_I^{-1} b = -\mu^\top b.$$

Now, if $x^\star$ is an optimal solution of the primal problem, then $c^\top x \geq c^\top x^\star$ by (primal) feasibility of x, and $c^\top x^\star \geq -\mu^\top b$ by weak duality and (dual) feasibility of μ. The analogous argument holds for an optimal solution of the dual.

Recall the negative sign from equation 12.

Two basis A_I and $A_{I'}$ are said to be *adjacent* if they differ by a single column, *i.e.*, I and I' differ by a single index. The basic solutions associated with adjacent basis are also said to be adjacent. When the simplex method does not show neither optimality nor unboundedness, then it moves from the current solution to an adjacent solution via a process similar to pivoting in Gaussian elimination. A primal basic feasible solution $x = (x^b, x^n)$ is called *degenerate* if $x_i^b = 0$ for some $i \in I$. If x is nondegenerate, then for each $j \in J$, there is at most one basis to be obtained by replacing a column of A_I with the j-th column of A. In other words, a nondegenerate primal basic feasible solution has at most $|J| = m$ adjacent solutions.

Denote by K the index set of one of those adjacent solutions, and by $k \in J$ the element of K not in I. Also, let A_K be the adjacent basis, and x' the adjacent solution. If, additionally, k is associated with a violated dual inequality, that is, $-\left(c_I A_I^{-1} A_J\right)_k < -c_k$, then x' is a strict improvement over x,

$$c^\top x' < c^\top x.$$

This strict improvement can be arbitrarily large if $A_I^{-1} a_k \leq 0$, where a_k is the k-th column of A. In this case, the primal problem is unbounded below and the dual problem is infeasible.

The algorithm below assumes all candidate solutions are nondegenerate. The degenerate case is discussed afterwards.

Algorithm A.5: Phase II primal simplex method.

Data: a primal feasible basis A_I.
Result: an optimal solution.
while A_I not dual feasible **do**
  choose $k \in J$ corresponding to a violated dual inequality
  if $A_I^{-1} a_k \leq 0$ **then**
    return optimal value is $-\infty$
  end
  calculate the adjacent primal feasible basis A_K
  set $A_I \leftarrow A_K$
end

Although rare in practice, it is possible that a degenerate solution, having multiple basis associated with an index $j \in J$, causes the algorithm to return to the same solution. This phenomenon is known as *cycling*, but it can be avoided via additional rules for basis selection: the lexicographic rule and Bland's rule. Both are described in section 3.4 of [6].

A phase I method can be devised by solving an auxiliary LP that uses a vector of *artificial variables* $x^a \in \mathbb{R}^m$,

$$\begin{aligned} \min_{(x,x^a)\in\mathbb{R}^{n+m}} \quad & \textstyle\sum_{j=1}^{m} x_j^a \\ \text{s.t.} \quad & Ax + Ix^a = b, \\ & x \geq 0, x^a \geq 0. \end{aligned} \tag{13}$$

The basic feasible solution $(x, x^a) = (0, b)$ is known, and so the phase II method can be applied to the preceding problem. Its solution yields a primal basic feasible solution of the original problem (when the optimal value is nonnegative); certificates the original problem is infeasible (when its optimal value is strictly negative); or contains basic artificial variables that can be removed via degenerate basis changes, or by removing redundant constraints from the original problem (and the corresponding artificial variables from the auxiliary problem).

We direct the reader to section 2.3 of part I of [22] for a presentation of the simplex method (but note it considers a maximization primal problem instead of minimization problem). Alternatively, confer chapter 3 of [6], and perhaps the preceding chapters for a review of auxiliary definitions.

We now discuss a generic method for solving integer and mixed-integer programs.

Definition A.1.6 (Branch-and-bound method)**.** There are two main ideas behind this method: relaxing the integer or mixed-integer problem by removing the integrality constraints; and progressively dividing the feasible set and its subdivisions until termination of the algorithm. There exist other branch-and-bound methods, but we do not discuss them here.

The ideas behind the method are essentially the same between integer and mixed-integer problems, and so we discuss only the wholly integer case for simplicity. The relaxation of the integer problem is a linear program whose optimal value bounds the original optimal value from below. When the optimal solution of this LP is integral, it coincides with the optimal solution of the original IP. This is unlikely to happen without further assumptions.

The set of all feasible solutions of the original IP can be too large to be searched for an optimal solution directly. An alternative is to divide it into smaller sets, and optimize over those instead. This procedure can be visualized by depicting it as a tree. A partial representation of the tree for a pure binary program is given in figure A.1.6. Each node has all constraints of its parent plus one additional constraint, and is thus a different optimization problem. Note that, despite this depiction, the method does not require a representation of the whole tree as an input. One can interpret the algorithm as constructing the tree iteratively, or, rather, constructing only the nodes that are necessary as they become relevant.

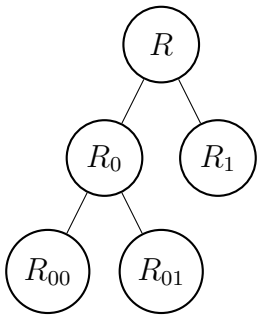

Figure 13: The root of the tree, R, represents the original problem. Its children, R_0 and R_1, represent the feasible solutions where $z_1 = 0$ and $z_1 = 1$ respectively. The children of R_0, R_{00} and R_{01}, represent the feasible solutions where $(z_1, z_2) = (0, 0)$ and $(z_1, z_2) = (0, 1)$, and so on.

Following through with the subdivision procedure would result into an enumeration of all feasible solutions. The key is to avoid unnecessary subdivisions by eliminating a node and all its children: this process is called *pruning*. Let N denote a node of the tree, and also the subproblem associated with it. We can prune N whenever any of the following criterions hold:

Infeasibility. N is infeasible, *i.e.* its feasible set is empty;

Optimality. An optimal *integer* solution of N is known;

Value dominance. The optimal value of N is bounded away from the optimal value of the original problem.

An example for the value dominance criterion is when both an optimal non-integer solution $x^\star_N$ of N and a feasible solution z of the original IP are known,

satisfying

$$c^\top z \leq c^\top x_N^\star.$$

Alternatively, if $\overline{p}$ is a known upper bound for the optimal value of the IP, and a feasible solution of the dual of N with value d_N is known, then N can be pruned if

$$\overline{p} \leq d_N.$$

Algorithm A.6: Generic branch-and-bound method.

Data: a set of problems $\mathcal{N}$, and upper and lower bounds.
Result: an optimal solution.

```
while 𝒩 not empty do
    select, remove, and solve a problem N from 𝒩
    if N can be pruned then
        Skip to next iteration        // Note that N was already
         removed.
    end
    divide N and add subdivisions to 𝒩
end
```

It remains to detail the *node selection* and *problem division* subroutines, which we only discuss briefly (see references after this definition for further details). Node selection can be done via *a priori* or *adaptative* rules. A common (essentially) a priori rule is *depth-first search with backtracking*, or *last in, first out* (LIFO). Intuitively, the set $\mathcal{N}$ can be thought of as a stack or pile, where the problem on the top is accessible. Figure 14 illustrates a possible depth-first search mid-execution. Computationally, it can be interesting to represent $\mathcal{N}$ in such a way its elements can be readily accessed. For example, whenever the upper or lower bounds are updated, one could verify if each node can be pruned by the value dominance criterion.

Problem division is done by adding linear constraints to each node. In practice, specific choices of coefficients are used, one of which is *variable dichotomy*. It generates two subproblems, one with the constraint $z_i \leq b$ and another with the constraint $z_i \geq b + 1$, where $b \in \mathbb{Z}$ and i is the index of an optimization variable. We now briefly discuss how the coefficient b and the index i can be chosen.

When a solution $x^\star$ to the linearly relaxed problem is known, an useful choice of b is $\lfloor x_i^\star \rfloor$, as it removes $x^\star$ from the feasible sets of its children (when $x_i^\star \notin \mathbb{Z}$). A benefit of variable dichotomy is that the kinds of constraints added at each division are upper and lower bounds for each variable. Thus, rather than accumulating more constraints, the method can simply update the bounds of existing constraints.

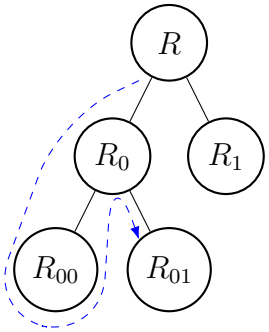

Figure 14: Initially, $\mathcal{N} = \{R\}$. The first iteration of the method removes and solves R, and then adds R_1 and R_0 to $\mathcal{N}$, in that order. The second iteration removes and solves R_0, and adds R_{01} and R_{00}. The third iteration removes and solves R_{00}, and determines it should be pruned. After the third iteration, $\mathcal{N} = \{R_1, R_{01}\}$, and the algorithm will remove and solve R_{01} in the next iteration.

Index selection, whether for variable dichotomy or not, is shown by empirical evidence to be important for the running time of the method, as frequently only a few variables need to be integer for the rest to turn out integer in the LP. Common methods for index selection include: *a priori* user-specified priorities (perhaps through heuristics based on interpretation of the model); and automated index selection based on estimations of how coercing a variable to be integer-valued degrades the optimal value (for minimization problems, how much the optimal value increases).

For a brief introduction to the branch-and-bound method, see section 11.2 of [6]. A more in-depth discussion can be found in sections 4.1 and 4.2 of part II of [22], covering a general branch-and-bound algorithm that uses LP relaxations, including the previously mentioned node selection and problem division submethods.