2015

Two-Stage Stochastic Mixed Integer Linear Optimization

Anahita Hassanzadeh
Lehigh University

Two-Stage Stochastic Mixed Integer Linear Optimization

by

Anahita Hassanzadeh

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial Engineering

Lehigh University

August 2015

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dr. Theodore K. Ralphs
Dissertation Advisor

Committee Members:

_____
Dr. Theodore K. Ralphs, Committee Chair

_____
Dr. Shabbir Ahmed

_____
Dr. Simge Küçükyavuz

_____
Dr. Aurélie C. Thiele

_____
Dr. Luis F. Zuluaga

Acknowledgements

First and foremost, I would like to thank my advisor, professor Ted Ralphs, whose ideas, inspirations, guidelines, and constant support made this work possible. I have the deepest appreciation for his enthusiasm and patience and especially for his perspectives on ways to approach challenging problems. I am very grateful for the countless hours he has spent mentoring me in my research and my career.

I would also like to thank my thesis committee members, professors Aurélie Thiele and Luis Zuluaga from the Department of Industrial and Systems Engineering at Lehigh University, as well as my external committee members professors Shabbir Ahmed and Simge Küçükyavuz, for their help and suggestions for enhancing this thesis.

My special thanks go to two very supportive faculty members of the Department of Industrial and Systems Engineering at Lehigh University, professor Katya Scheinberg and professor Frank Curtis. Dr. Scheinberg has been a true role model for me, to whom I owe the very valuable experience I gained during my internship in the second year of my studies. I appreciate the time and effort she has put in helping me with my career during my time at Lehigh and afterwards. My thanks also go to a great teacher, Dr. Curtis, whose office door has always been open to me and who has generously answered my questions many times. I would also like to thank professor Asgeir Tomasgard who provided me with the opportunity to visit the stochastic optimization research team at the Norwegian University of Science and Technology.

My years at Lehigh could never be as fulfilling without my amazing friends. Thank you Inkeri and Doug Coleman, Kiana Kouchakzadeh, Leyla Mohseninejad, Joyita Bhadra,

Violette Pradel, Tom Matarazzo, Burcu Unlutabak, Salman Reza and Jiadong Wang.

I also owe my profound gratitude to my family. I would like to thank my husband, Neil Dexter, for his continuous encouragement and support in the past four years. My heartfelt thanks also go to my parents, my brother, Babak, and my sister, Ainaz, to whom I owe every success of my life. Finally, my thanks go to the Gonglewskis for their love and support.

Contents

List of Tables

List of Figures

Abstract

The primary focus of this dissertation is on optimization problems that involve uncertainty unfolding over time. In many real-world decisions, the decision-maker has to make a decision in the face of uncertainty. After the outcome of the uncertainty is observed, she can correct her initial decision by taking some corrective actions at a later *time stage*. These problems are known as *stochastic optimization problems with recourse*. In the case that the number of time stages is limited to two, these problems are referred to as *two-stage stochastic optimization problems*. This class of optimization problems is the focus of this dissertation. The optimization problem that is solved before the realization of uncertainty is called the *first-stage* problem and the problem solved to make a corrective action on the initial decision is called the *second-stage* problem. The decisions made in the second-stage are affected by both the first-stage decisions and the realization of random variables. Consequently, the two-stage problem can be viewed as a parametric optimization problem that involves the so-called *value function* of the second-stage problem. The value function describes the change in optimal objective value as the right-hand side is varied and understanding it is crucial to developing solution methods for two-stage optimization problems.

In the first part of this dissertation, we study the value function of a MILP. We review the structural properties of the value function. We propose a discrete representation of the MILP value function. We show that the structure of the MILP value function arises from two other optimization problems that are constructed from its discrete and continuous components. We show that our representation can explain certain structural properties

of the MILP value function, such as the sets over which the value function is convex. We then provide a simplification of the *Jeroslow Formula* obtained by applying our results. Finally, we describe a cutting plane algorithm for its construction and determine the conditions under which the proposed algorithm is finite.

Traditionally, the solution methods developed for two-stage optimization problems consider the problem in which the second-stage problem involves only continuous variables. In recent years, however, two-stage problems with integer variables in the second-stage have been visited in several studies. These problems are important in practice and arise in several applications in supply chain, finance, forestry and disaster management, among others. The second part of this dissertation concerns the development and implementation of a solution method for the two-stage optimization problem where both the first and second stage involve mixed integer variables. We describe a generalization of the classical Benders' method for solving mixed integer two-stage stochastic linear optimization problems. We employ the strong dual functions encoded in the branch-and-bound trees resulting from solution of the second-stage problem. We show that these can be used effectively within a Benders' framework and describe a method for obtaining all required dual functions from a single, continuously refined branch-and-bound tree that is used to warm start the solution procedure for each subproblem.

Finally, we provide details on the implementation of our proposed algorithm. The implementation allows for construction of several approximations of the value function of the second-stage problem. We use different warm-starting strategies within our proposed algorithm to solve the second-stage problems, including solving all second-stage problems with a single tree. We provide computational results on applying these strategies to the stochastic server problems (SSLP) from the stochastic integer programming test problem library (SIPLIB).

Chapter 1

Introduction

We begin by considering the general optimization problem

$$\inf_{x \in U} f(x), \tag{1.1}$$

where the $f : \mathbb{R}^n \to \mathbb{R}$ is called the *objective function* and the set $U \subseteq \mathbb{R}^n$ is the *feasible region* of the problem. A vector $x \in U$ is referred to as a *feasible solution*, which expresses a *decision*, and $f(x)$ is its associated *solution value*, or the *cost* of the decision. If the set U is an empty set, the problem is called *infeasible*. Any $x^* \in U$ such that $f(x^*) = z_G$ is called an *optimal solution* and $f(x^*)$ is called the *optimal value*.

The optimization problem (1.1) assumes a single decision-maker with a single objective function making a single set of decisions at a single point of time where the descriptions of the objective function f and the feasible region U are fixed and known. Optimization problems in real-world, however, often involve multiple decision-makers with congruent or conflicting objective functions that make decisions in multiple points of time. The decisions made at a given point of time often affect future decisions and are affected by future uncertainty themselves. Several frameworks have been developed that extend (1.1) by adding one or more of these features to it. One such framework is *multi-objective optimization*, which extends the traditional framework to capture the decisions involving a single decision-maker trading off multiple objective functions. The setting of multi-

objective optimization problems is still restricted to centralized decision processes that are controlled by a single decision-maker. A class of optimization problems that generalizes this setting is *multi-level* optimization problems. This setting allows for the modeling of situations that involves several levels of decision-maker whose decisions are made sequentially with each decision potentially affecting the decisions made in higher or lower levels of the overall hierarchy. Multi-level optimization problems provide a representation of game-theoretic processes in which decision-makers make sequential decisions in multiple rounds. In this setting each player strives to optimize her individual, competing objective function by making a decision that takes into account the reactions of the other players. This setting assumes *perfect information*, meaning that the decision-maker is able to predict the reaction of other players to her decision.

Both multi-objective and multi-level optimization problems assume that at the time of decision-making, the decision-makers have complete information about the parameters of the optimization problems to be solved by other players lower in the hierarchy. In this sense, these problems are *deterministic*. In practice, however, one frequently faces problems where the actual cost or feasibility of a decision is only determined *after* the decision is made. There are several frameworks that allow for modeling uncertainty in optimization problems. Like the deterministic case, these frameworks allow for single or multiple decision makers who make decisions hierarchically or at a single point of time. We discuss these frameworks more formally next.

1.1 Optimization Under Uncertainty

We now consider a generalization of (1.1) formulated as

$$\begin{aligned} &\inf f_\omega(x) \\ &\text{s.t. } g_i(x,\omega) \leq 0, i = 1,\dots,m,\ \omega \in \Omega, \end{aligned} \tag{1.2}$$

where ω represents an uncertain element of the problem that comes from a set of outcomes, Ω. The uncertain element is said to be *realized* or *observed* when it actually happens and

its value is observed.

Chance-Constrained Optimization Problems. In problem (1.1), a feasible point must satisfy all the *constraints*, $g_i(x, \omega) \leq 0$ for $i = 1, \ldots, m$ and $\omega \in \Omega$. *Chance-constrained* optimization, on the other hand, allows for incorporating uncertainty by relaxing some or all of the constraints probabilistically. The goal in solving a chance-constrained optimization problem is to satisfy the constraints in "most" cases. Chance-constrained models assume that the uncertain terms in the problem can be modeled as random variables derived from a probability distribution that is known or can be estimated. That is, Ω is assumed to be the set of all possible outcomes of a probability space. The requirement in the constraints is to ensure that for a given decision $\hat{x}$, the probability that the constraint is violated is no more than a given value α_i where $0 \leq \alpha_i \leq 1$, $i = 1, \ldots, m$.

Let $\text{prob}\{g_i(x, \omega) \leq 0\}$ denote the probability that the i^{th} constraint holds. Then the chance-constrained problem can be written as

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } \text{prob}\{g_i(x, \omega) \leq 0\} \geq 1 - \alpha_i \quad \text{for } i = 1, \ldots, m. \end{aligned} \tag{1.3}$$

In the above case, the constraints are assumed to be independent of each other. Chance-constrained problems can also model the case where interactions exists between the constraints. In this case, the set of constraints can be written as

$$\text{prob}\{g_i(x, \omega) \leq 0 \text{ for } i = 1, \ldots, m\} \geq 1 - \alpha.$$

Chance-constraints are appealing from a modeling perspective, as they provide means to model real-world requirements of an optimization problem such as the desired service-level. However, determining the probability levels and information about the joint probability distribution with potentially a large number of random variables can be a challenge.

Robust Optimization Problems. Another framework that is used to model uncertainty is *robust optimization*. This framework captures the case where a single decision-maker aims to protect against the risk by a one-time decision that is made in the presence of uncertainty. In particular, the decision-maker is interested in minimizing the maximum cost that can incur as a result of the uncertainty. Here, the *minmax* goal is used to protect the decision against the *worst-case* outcome. The robust framework assumes the uncertainty belongs to an *uncertainty set*. That is, the set of outcomes where the uncertain element is coming from is this uncertainty set. This set can be a complicated convex or polyhedral set, or simply is the bounds of the uncertain parameters. This allows for modeling uncertainty whose specific probability distribution is not known. This is an appealing property where historical data is not available and probabilities are challenging to estimate.

The general model of a robust optimization problem is

$$\begin{aligned} &\min \max_{\gamma_0 \in \gamma_0} f(x, \gamma_0) \\ &\text{s.t. } g(x, \gamma) \leq 0, \ \gamma \in \Gamma, \end{aligned} \tag{1.4}$$

where γ denotes the uncertain elements and Γ_0 and Γ are given uncertainty sets and have special forms such that the problem can be solved with efficient algorithms. For example, these sets can be assumed to be convex or polyhedral.

Stochastic Optimization Problems. A critical difference between models of optimization problems that involve uncertainty origins from the way they immunize the decision against the outcome of uncertain elements. While in the robust optimization framework the goal is to minimize the maximum possible cost, in the stochastic optimization problem the expected cost of decisions is minimized. In this scheme, the random cost is replaced by its *expected value*, denoted by $\mathbb{E}$. In this case, the general problem (1.2) takes the form

$$\inf \ \mathbb{E}_{\omega \in \Omega} f(x, \omega). \tag{1.5}$$

When the probability space Ω is discrete and finite, ω represents which one of a finite number of explicitly enumerated scenarios is realized and p_ω represents the probability of such an outcome. Then, (1.5) can be written as

$$\inf \sum_{\omega \in \Omega} p_\omega f(x, \omega). \tag{1.6}$$

Similarly, if Ω represents a continuous space with the probability density function $\rho(\omega)$ defined for $\omega \in \Omega$, then (1.5) is equivalent to

$$\inf \int_\Omega f(x, \omega)\rho(\omega)d\omega. \tag{1.7}$$

There are studies that consider random variables that follow an infinite distribution. For a review of the assumptions and properties of this case we refer the reader to (Ruszczynski and Shapiro, 2003). A standard approach to solve stochastic optimization problems is by generating *scenarios*. A scenario is a realization of a future event consisting of an outcome of random variables. In the case of stochastic optimization, scenarios can be constructed by drawing samples from the known probability space(s) and observing their values, rendering the outcome space where the uncertainties come from finite.

Both the chance-constrained and robust optimization frameworks consider decisions that are made at a single point of time. In practice, however, the decision-maker is often able to react to the new situation as new information becomes available. While the decision-maker makes the initial decision in the face of the unknown, she can take *recourse* actions at one or more points of time in future. This framework is known as *stochastic optimization with recourse*. This framework resembles multi-level optimization problems in that decisions are made dynamically and they change according to the realization of uncertainty. We can also view these problems as multi-level problems in which decision makers cooperate and their objectives align with one another and therefore, can be interpreted as a problem with a single decision-maker. Stochastic optimization problems, however, extend the multi-level framework by allowing uncertainty to be explicitly mod-

eled. Like chance-constrained optimization, stochastic optimization assumes a known probability distribution where the random variables can be drawn from a probability space.

A classical example of stochastic optimization problems with recourse is the unit commitment problem in which the utility company has to decide *ahead of time* whether to start up/shut down power generators to meet an unknown future demand load. In this case, the distribution of demand maybe known or can be estimated from statistical data. Other examples arise in areas such as disaster planning, where decisions on dispatching storm/wildfire protections equipment are made before the weather conditions are observed. In supply chain management, for example, decisions on the quantity of order for retail goods are made with non-deterministic demand and holding costs.

Studying stochastic optimization problems with recourse is the focus of this thesis. We mentioned earlier that in the setting of stochastic optimization, the decision-maker has to commit to certain actions before the unknown terms are observed (the problems is, of course, deterministic if she can postpone making commitments after the observation). Stochastic optimization with recourse also assumes there is a *possibility of recourse* that allows the decision-maker to correct her earlier decisions in one or multiple times in future. This decision-making process can be illustrated with:

$$
\begin{array}{cl}
x_0 \in U^0 & \text{initial decision} \\
\omega_1 \in \Omega^1 & \text{observation} \\
x_1 \in U^1(x_0) & \text{recourse decision} \\
\omega_2 \in \Omega^2 & \text{observation} \\
\vdots & \\
\omega_N \in \Omega^N & \text{observation} \\
x_N \in U^N(x_0, \ldots, x_{N-1}) & \text{recourse decision.}
\end{array}
$$

We refer to the point of time when a decision is made as a *time stage*. In the above

illustration, the initial stage x_0 is followed by N recourse actions taken in N future time stages. The number N is usually assumed to be finite. Typically, a stochastic optimization problem that involves a large number of time stages is very challenging to solve. A simpler variation that is often considered is where the first observation is followed by only one corrective decision. The resulting recourse problems are important in theory and practice and are referred to as *two-stage stochastic optimization problems*. Next, we introduce two fundamental concepts in this class of problems.

In this thesis, we focus on two-stage stochastic optimization problems which involves random variables that follow a discrete distribution. We consider the case where all or some of the variables in the first or second-stage problems have to be integer. We also focus on the case where the objective function and constraints in both stages are linear functions. These problems lie at the intersection of stochastic linear optimization with recourse and discrete linear optimization. We review these problems in more depth in the upcoming Section 1.3. In the remainder of this section, we provide an overview of some of the fundamental concepts in discrete optimization that are necessary for this work. We then review the principle of Benders' decomposition applied to optimization problems and in particular, to two-stage linear optimization problems.

1.2 Discrete Optimization

A *mixed integer linear optimization problem* (MILP) is a variation of the optimization problem (1.1) with a linear objective function and a polyhedral feasible region where a subset of the variables are *integer variables*. Consider the nominal MILP instance

$$\inf_{x \in S} c^\top x, \tag{MILP}$$

where $c \in \mathbb{R}^n$ is the objective function vector and $S = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = \tilde{b}\}$ is the feasible region, described by $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{R}^m$, and a scalar r indicating the number of integer variables. We refer to this problem as the *primal* problem. Throughout the thesis, we assume $rank(A, d) = rank(A) = m$. The variables of the instance are indexed

on the set $N = \{1, \ldots, n\}$, with $I = \{1, \ldots, r\}$ denoting the index set for the integer variables and $C = \{r+1, \ldots, n\}$ denoting the index set for the continuous variables. For any $D \subseteq N$ and a vector v indexed on N, we denote by v_D the sub-vector consisting of the corresponding components of v. Similarly, for a matrix M, we denote by M_D the sub-matrix constructed by columns of M that correspond to indices in D.

In many applications of integer optimization, we are interested in a parametric version of (MILP) that allows for analyzing the changes in the optimal value of the problem when the parameters of the problem are perturbed. In the case of two-stage stochastic optimization problems, it will be clear soon that we are interested in the change in the optimal value as the right-hand side of a given MILP instance is modified. In what comes next, we introduce a function that is designed to provide such information.

Value Function

The *value function* of a MILP is a function $z : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}$ that describes the change in the optimal solution value of a MILP as the right-hand side is varied. In the case of (MILP), we have

$$z(b) = \inf_{x \in S(b)} c^\top x \quad \forall b \in B, \tag{1.8}$$

where for $b \in \mathbb{R}^m$, $S(b) = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = b\}$. The set of right-hand sides, B, is the set of real vectors such that the corresponding feasible region is non-empty. That is, $B = \{b \in \mathbb{R}^m \mid S(b) \neq \emptyset\}$. By convention, we let $z(b) = \infty$ if $S(b) = \emptyset$ and $z(b) = -\infty$ when the infimum is not attained at any feasible point. To simplify the presentation, we assume that $z(0) = 0$.

A special case of a MILP is where none of the variables have integrality restrictions, e.g., $r = 0$. The resulting problem is known as a *linear optimization problem* (LP) and is defined as

$$\inf_{x \in S_{LP}} c^\top x, \tag{1.9}$$

where the feasible region S_{LP} is defined as

$$S_{LP} = \{x \in \mathbb{R}^n_+ \mid Ax = \tilde{b}\}. \tag{1.10}$$

We will discuss in Chapter 2 that the structure of the MILP value function (1.8) is closely related to that of a certain LP. We define the LP value function as

$$z_{LP}(b) = \inf_{x \in S_{LP(b)}} c^\top x \quad \forall b \in B, \tag{1.11}$$

where for $b \in \mathbb{R}^m$, $S_{LP}(b) = \{x \in \mathbb{R}^n_+ \mid Ax = b\}$. Let us look at an example first.

Example 1.1. Consider

$$\begin{aligned} z_{LP}(b) = \inf\ & 4x_1 + 6x_2 + 7x_3 \\ \text{s.t. } & x_1 + 2x_2 - 7x_3 = b \\ & x_1, x_2, x_3 \in \mathbb{R}_+. \end{aligned} \tag{1.12}$$

The function z is plotted in Figure 1.1. We have

$$z_{LP}(b) = \begin{cases} -b & \text{if } b < 0 \\ 3b & \text{if } b \geq 0, \end{cases}$$

From the figure we can see that the value function is a piecewise convex and continuous function. This structure arises from the properties of the so-called *dual functions* to the LP value function. A dual function is simply a function that bounds the value function from below.

Definition 1.1. A function $f : \mathbb{R}^{m_2} \to \mathbb{R} \cup \{\pm\infty\}$ is said to be *dual* to the value function z if

$$f(b) \leq z(b) \quad \forall b \in \mathbb{R}^{m_2}. \tag{1.13}$$

f is *strong* at $\hat{b} \in \mathbb{R}^{m_2}$ if $f(\hat{b}) = z(\hat{b})$.

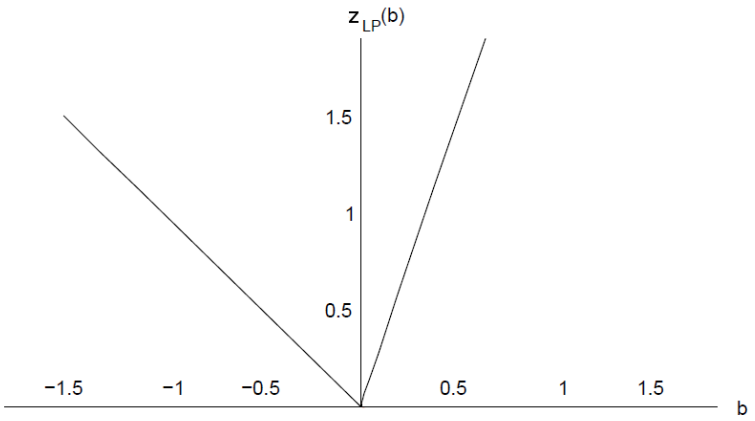


Figure 1.1: The LP value function (1.12)

The properties of dual functions, as well as methods for constructing them are studied in the theory of duality. The goal is to construct dual functions that approximate the value function closely. In this respect, the value function is the best dual function. However, an explicit construction of the value function, as we will see later, can be very challenging. In practice, we often need a method to generate a dual function that provides the best bound for a given instance. This is done by solving an optimization problem for a given $\hat{b}$, called the *dual problem*

$$\sup\{f(\hat{b}) \mid f(b) \leq z(b) \quad \forall b \in \mathbb{R}^m, f : \mathbb{R}^{m_2} \to \mathbb{R} \cup \{\pm\infty\}\}. \tag{1.14}$$

We call a dual function *optimal* if it is optimal for the problem (1.14). Although the definition of a dual problem allows for a wide range of dual functions to be selected, the search space in (1.14) is often in practice restricted to special families of functions which can be constructed with tractable methods.

Consider the case of an LP. Let us restrict the dual functions to be linear. For a given $\hat{b} \in \mathbb{R}^m$, the dual problem (1.14) can be written as

$$\sup\{\hat{b}^\top \nu \mid \hat{b}^\top \nu \leq z_{LP}(\hat{b}) \quad \forall \nu \in \mathbb{R}^m\}, \tag{1.15}$$

where z_{LP} is defined in (1.11). The problem (1.16) is equivalent to

$$\sup_{\nu \in \mathcal{D}_{LP}} \hat{b}^\top \nu, \tag{1.16}$$

where $\mathcal{D}_{LP} = \{\nu \in \mathbb{R}^m \mid A^\top \nu \leq c\}$. The latter problem is another LP which is referred to as the dual to (1.9). The primal-dual relationship between the problems (1.9) and (1.16) is important to study the properties of the LP value function and construction of dual functions to it. The following result is key to LP duality.

Theorem 1.1. *(LP weak and strong duality by Bazaraa et al. (1990)) For a given $\hat{b} \in \mathbb{R}^m$ with finite $z_{LP}(\hat{b})$, we have $b^\top \nu \leq z_{LP}(\hat{b})$ for any $\nu \in \mathbb{R}^m$ that is a feasible solution to (1.16). Furthermore, there always exists an optimal solution ν^* to the dual problem such that $f(b) = b^\top \nu^*$, $b \in \mathbb{R}^m$ is a strong dual function to the value function w.r.t. $\hat{b}$.*

The first and second parts of the Theorem 1.1 are respectively known as the weak and strong duality theorems. As a consequence of the weak duality theorem, we have a dual function for the LP value function $f(b) = b^\top \nu$, where ν is a feasible solution to the dual problem (1.16). This in fact holds even when the instance of the LP problem is not finite. The only case that the weak duality theorem does not hold is when both the dual and the primal problems are infeasible.

Strong and weak duality theorems reveal important structural properties of the LP value function. To explain this relation, we first need some definitions.

Definition 1.2. A set $C \subseteq \mathbb{R}^n$ is a called a *cone* if for any $x \in C$ and $\lambda \geq 0$ we have $\lambda x \in C$.

Furthermore, a polyhedron that can be described in the form $S = \{x \in \mathbb{R}^n : Ax \geq 0\}$ for some $A \in \mathbb{R}^{m \times n}$ is called a *polyhedral cone*.

Definition 1.3. The *epigraph* of the function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\text{epi } f = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} : t \geq f(x)\}.$$

Definition 1.4. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *polyhedral* if its epigraph are polyhedral.

Let us define $\mathcal{K}_{LP}$ to be the polyhedral cone that is the positive linear span of A, i.e., $\mathcal{K}_{LP} = \{\lambda_1 A^1 \ldots + \lambda_n A^n : \lambda_1, \ldots, \lambda_{n-r} \geq 0\}$, where A^j is the j^{th} column of A. This cone is the set of right-hand sides over which z_{LP} is finite and plays an important role in the structure of the LP value function. We assume the cone $\mathcal{K}_{LP}$ is non-empty, therefore $\mathcal{D}_{LP} \neq \emptyset$. Then, we can write the LP value function as

$$z_{LP}(b) = \sup_{\nu \in \mathcal{D}_{LP}} \hat{b}^\top \nu. \tag{1.17}$$

Since $\mathcal{D}_{LP} \neq 0$, from the Minkowski-Weyl theorem, we have that there exists a non-empty set $\{\nu^i\}_{i \in K}$, the set of a finite number of extreme points of $\mathcal{D}_{LP}$ indexed by set K. Furthermore, when $\mathcal{D}_{LP}$ is unbounded, there exists a non-empty set of a finite number of extreme directions $\{d^j\}_{j \in L}$ be indexed by set L. If the LP with right-hand side $\hat{b}$ has a finite optimum, then

$$z_{LP}(\hat{b}) = \sup_{\nu \in \mathcal{D}_{LP}} \hat{b}^\top \nu = \sup_{i \in K} \hat{b}^\top \nu^i. \tag{1.18}$$

Otherwise, for some $j \in L$, we have $\hat{b}^\top d^j > 0$ and $z_{LP}(\hat{b}) = +\infty$.

The convexity of z_{LP} follows from the representation (1.18), since z_{LP} is the maximum of a finite number of affine functions and is hence a convex polyhedral function (Bazaraa et al., 1990; Blair and Jeroslow, 1977). For such a function, we can derive dual functions using its subgradient information. We discuss this next. The first following result guarantees the existence of subgradients for z_{LP}, while the second result provide means to generate them.

Proposition 1.1. *(Ruszczynski and Shapiro, 2003) If $f : \mathbb{R}^m \to \mathbb{R}$ is a convex function and $x \in \mathbb{R}^m$, then $\partial f(x)$ is non-empty and bounded.*

Definition 1.5. A real function f is said to be *differentiable* at a point if its derivative exists at that point.

Intuitively, for a function to be differentiable at a point x_0 of its domain, the right and left limit used in the usual differentiability definition along *any* path through x_0 should

be the same. For the scope of our work, it suffices to note that if f is differentiable in a neighborhood of x_0 then

$$f'(x_0, p) = \nabla f(x_0)^\top p.$$

Definition 1.6. A vector $g \in \mathbb{R}^n$ is a subgradient of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point x_0 if

$$f(x) - f(x_0) \geq g^\top (x - x_0) \ \ \forall x \in \mathbb{R}^n.$$

The above inequality is called the *subgradient inequality*. The set of all subgradients of f at x_0 is called the *subdifferential*. We denote this set by $\partial f(x_0)$.

Definition 1.7. The function f is called *subdifferentiable* at x_0 if $\partial f(x_0) \neq \emptyset$.

If a function is differentiable at a point, then its subdifferential is a singleton consisting of the gradient of the function at that point.

Proposition 1.2. *(Blair and Jeroslow, 1977; Bazaraa et al., 1990)*

- *z_{LP} is convex, continuous and sub-differentiable on $\mathcal{K}_{LP}$.*
- *At a given $\hat{b} \in \mathcal{K}_{LP}$, we have*

$$\partial z_{LP}(\hat{b}) = cl(conv(\{\nu_1, \ldots, \nu_k, d_1, \ldots, d_l\})),$$

  where $\nu_1, \ldots, \nu_k, d_1, \ldots, d_l$ respectively denote the extreme points and directions that are optimal to the dual LP (1.16) with $b = \hat{b}$.

- *If z_{LP} is differentiable at $\hat{b} \in \mathbb{R}^m$, then the gradient of z_{LP} at $\hat{b}$ is the unique $\nu^* \in \mathcal{D}_{LP}$ such that $z_{LP}(\hat{b}) = \hat{b}^\top \nu^*$.*

Let us apply the above proposition to an example.

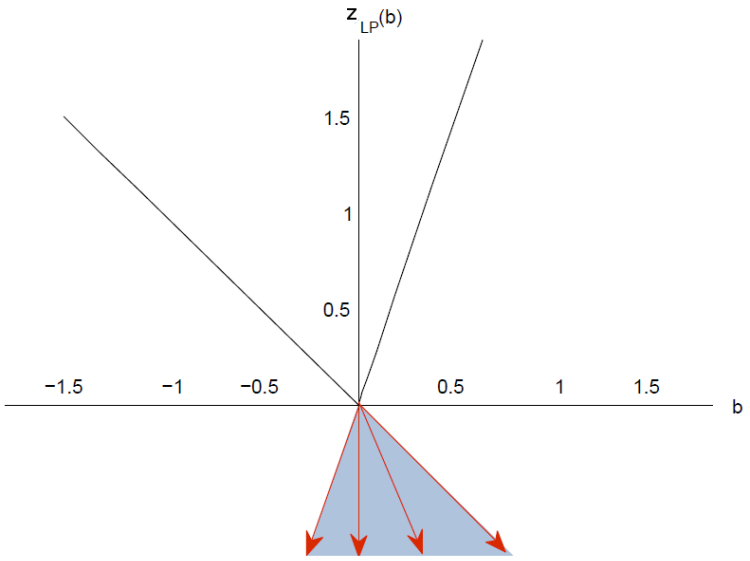


Figure 1.2: The LP value function (1.12) and the set of its subgradients at zero.

Example 1.2. The dual problem of an instance of (1.12) when b is fixed to $\hat{b}$ is

$$\begin{aligned} \sup\ & \hat{b}\nu \\ \text{s.t.}\ & \nu \leq 4 \\ & 2\nu \leq 6 \\ & -7\ \nu \leq 7 \end{aligned} \tag{1.19}$$

The extreme points of the feasible region of the dual problem above are -1 and 3. The function z_{LP} is plotted in Figure 1.2. As one can observe, this function is differentiable everywhere except at $b = 0$. The gradient of the function at a given point $\hat{b} \in (-\infty, 0)$ is -1, an extreme point of the feasible region to (1.19). Similarly, the gradient of the LP value function over $(0, \infty)$ is 3. At the origin, the function is subdifferentiable and we have

$$\partial z_{LP}(0) = cl(conv(\{-1, 3\})).$$

Proposition 1.2 enables us to derive polyhedral dual functions for the value function of an LP by solving the dual LP instance at a given right-hand side. For example, if the dual problem of (1.12) with b fixed at -1 is solved, from the optimal solution we derive

the dual function $f(b) = -2b$. If b is fixed to 4, the optimal dual solution is 0.5 and we obtain $f(b) = 3b$, another dual function. In the case where the right-hand side b is fixed to zero, we can obtain a dual function from the subgradient inequalities

$$z_{LP}(b) - z_{LP}(0) \geq g(b - 0) = gb,$$

where $g \in cl(conv(\{-1, 3\}))$ and $z_{LP}(0) = 0$ because the feasible region of the dual problem (1.19) is non-empty.

We mentioned earlier that construction of dual functions to the second-stage value function is key in classical methods of solving two-stage optimization problems. In the case of a linear second-stage problem, dual functions can be derived by obtaining subgradients of the value function as shown in Proposition (1.2). However, such linear functions do not give valid dual functions for the case where the second-stage problem contains integer variables. We show this with an example next and discuss further technical details in Chapter 2 and Chapter 3.

Example 1.3. Consider the MILP value function defined by adding integer variables to (1.12).

$$\begin{aligned} z(b) = \inf\ & 4x_1 + x_2 + 4x_3 + 6x_4 + 7x_5 \\ \text{s.t. } & 2x_1 - 2x_2 + x_3 + 2x_4 - 7x_5 = b \\ & x_1, x_2 \in \mathbb{Z}_+, x_3, x_4, x_5 \in \mathbb{R}_+. \end{aligned} \tag{1.20}$$

Figure 1.3 shows this non-convex and non-concave function, along with the LP value function in Example 1.1. The function $f(b) = 2.5b$ is plotted in red. As one can observe in the figure, this function is a dual function for z_{LP} (in dashed blue), but not for the MILP value function z.

What we illustrated in the previous example in fact can be generalized: the general dual functions to the MILP value function cannot be linear functions. However, there are classes of functions that result in dual functions for the MILP value functions. We discuss these classes and review their construction methods in Chapter 2.

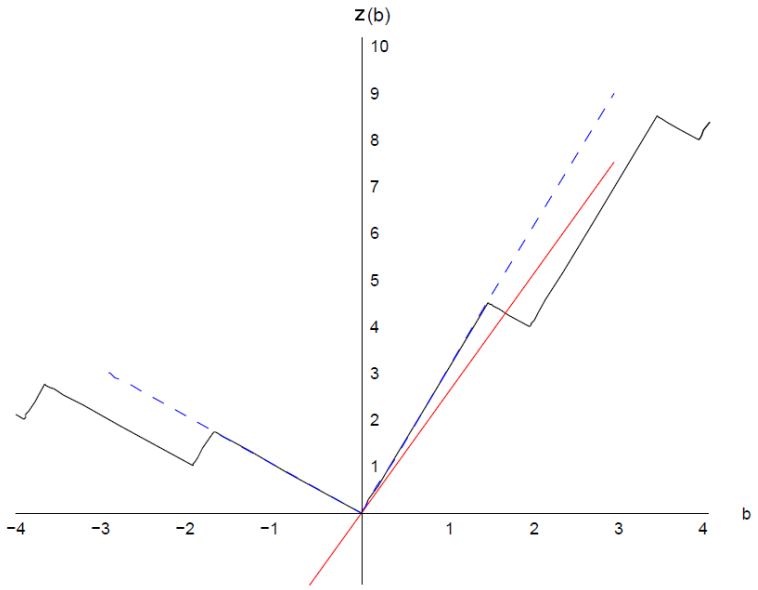


Figure 1.3: The MILP value function (1.20).

Benders' Principle

Consider the linear optimization problem (1.9) where the variables are partitioned into two groups. The problem can be written equivalently as

$$\begin{aligned} z_{LP} = \inf\ & c_1^\top x_1 + c_2^\top x_2 \\ \text{s.t. } & A_1 x_1 + A_2 x_2 = \tilde{b} \\ & x_1 \in \mathbb{R}^p_+, x_2 \in \mathbb{R}^{n-p}_+. \end{aligned} \tag{1.21}$$

Suppose that the variables x_1 are "complicating" variables, in the sense that the problem becomes easy to solve if these variables are fixed. The idea of *Benders' decomposition* is to partition the problem into two problems, one called the *master problem* which contains the complicating variables x_1, the second called the *subproblem* that contains the variables x_2. We formulate these problems next. Let us first rewrite (1.21) as

$$\inf_{x_1 \in \mathbb{R}^p_+} c_1^\top x_1 + z'(\tilde{b} - A_1 x_1), \tag{1.22}$$

where z' is the LP value function

$$\begin{aligned} z'(\tilde{b} - A_1 x_1) &= \inf c_2^\top x_2 \\ \text{s.t. } A_2 x_2 &= \tilde{b} - A_1 x_1 \\ x_2 &\in \mathbb{R}_+^{n-p}. \end{aligned} \tag{1.23}$$

From the formulation of the dual problem of an LP in (1.16), we have that the feasible region of the dual of (1.23) does not depend on x_1. Assuming that this feasible region is non-empty, we have a representation of (1.23) in terms of its dual extreme points and directions as

$$z'(\tilde{b} - A_1 x_1) = \inf \theta \tag{1.24}$$

$$\text{s.t. } 0 \geq (\tilde{b} - A_1 x_1)^\top d^j \quad \forall j \in L \tag{1.25}$$

$$\theta \geq (\tilde{b} - A_1 x_1)^\top \nu^i \quad \forall i \in K, \tag{1.26}$$

where ν_i with $i \in K$ and d_j with $j \in L$ are respectively the dual extreme points and directions defined in (1.18). Therefore, the original problem (1.21) can be equivalently written as

$$\begin{aligned} z_{LP} &= \inf c_1^\top x_1 + \theta \\ \text{s.t. } 0 &\geq (\tilde{b} - A_1 x_1)^\top d^j \quad \forall j \in L \qquad (1.27) \\ \theta &\geq (\tilde{b} - A_1 x_1)^\top \nu^i \quad \forall i \in K, \qquad (1.28) \\ x_1 &\in \mathbb{R}_+^p. \end{aligned}$$

Note that if $\tilde{b}$ is replaced with the parameter b, the right-hand side of each constraint in (1.28) is a dual function to the LP value function z'. That is,

$$(b - A_1 x_1)^\top \nu_i \leq z'(b - A_1 x_1). \quad \forall i \in K \tag{1.29}$$

Since the size of the sets L and K are typically large, it is not practical to include

all the constraints (1.27) and (1.28) in the problem. Instead, Benders' decomposition dynamically generates a subset of these sets in the master problem in such a way that the optimal solution is still obtained. The master problem is formulated as

$$
\begin{aligned}
z_{LP} = \inf\ & c_1^\top x_1 + \theta \\
\text{s.t. } & 0 \geq (\tilde{b} - A_1 x_1)^\top d^j \quad \forall j \in L' \subseteq L && (1.30)\\
& \theta \geq (\tilde{b} - A_1 x_1)^\top \nu^i \quad \forall i \in K' \subseteq K, && (1.31)\\
& x_1 \in \mathbb{R}^p_+.
\end{aligned}
$$

Benders' decomposition algorithm starts with $L' = \emptyset$ and $K' = \emptyset$ to obtain an initial solution $(\hat{x}_1, \hat{\theta})$ from the master problem. We then solve the resulting subproblem (1.23) by fixing x_1 to $\hat{x}_1$ and obtaining a new dual extreme point or direction to form a constraint in the form of (1.30)–(1.31). For a given $\hat{x}_1$, if the dual of (1.23) is unbounded, then (1.23) is infeasible and we can obtain an extreme direction to form a constraint in the form of (1.30). Constraints of this type are called *Benders' feasibility cuts*. When the dual problem w.r.t. x_1 has a finite optimum and x_1 is not feasible, we can generate a constraint in the form of (1.31) which is violated by x_1, which are known as a *Benders' feasibility cut*. These constraints are appended to the master problem, which is then resolved. The method terminates when the solution to the master problem satisfied $\hat{\theta} = z'(\tilde{b} - A_1\hat{x}_1)$. That is, the approximation of the value function of the subproblem obtained from the master problem in the final iteration should coincide with the exact value function of the subproblem at the right-hand side $\tilde{b} - A_1\hat{x}_1$.

Benders' method can be applied to MILPs. In this case, the variables are normally partitioned into the sets of integer and continuous variables. The integer variables are considered the complicating variables. The assumption is that by fixing the integer variables, the remaining problem is an LP that can be solved efficiently. Like the LP case, in each iteration of the Benders' algorithm, a new dual feasible solution to the subproblem is obtained that is used to construct an optimality or feasibility constraint. Due to the integrality restriction on the complicating variables, the master problem is, however, an

integer optimization problem. In what comes next, we discuss the application of Benders' decomposition to two-stage stochastic optimization problems.

1.3 Two-stage Stochastic Linear Optimization

Two-stage stochastic optimization problems involve decisions that are made in two points of time. The decisions made in the absence of information about the uncertainty are called the *first-stage decisions*, while the decisions made after the uncertainty is realized with the goal of correcting the initial decisions are called the *second-stage* decisions. We consider the following formulation of the two-stage stochastic mixed integer linear problem

$$\min_{x \in S_1} \Psi(x), \tag{SP}$$

where $S_1 = \{x \in \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1 - r_1} \mid Ax = b\}$ is the *first-stage feasible region* defined by $A \in \mathbb{Q}^{m_1 \times n_1}$ and $b \in \mathbb{Q}^{m_1}$. The objective function Ψ is defined by

$$\Psi(x) = c^\top x + \Xi(x), \tag{1.32}$$

where $c \in \mathbb{R}^{n_1}$ reflects the immediate cost of implementation of the first-stage solution and Ξ is a risk measure reflecting the additional cost incurred as a result of uncertainty about the future. As is conventional for stochastic optimization problems, we take Ξ to be the expected cost of the *recourse problem*, henceforth referred to as the *second-stage problem*. The second-stage problem is a MILP parameterized on both the value of the first-stage solution and a random variable ω. Formally, the function Ξ is defined by

$$\Xi(x) = \mathbb{E}_{\omega \in \Omega}[z(h_\omega - T_\omega x)], \tag{1.33}$$

for $x \in S_1$, where $T_\omega \in \mathbb{Q}^{m_2 \times n_1}$ and $h_\omega \in \mathbb{Q}^{m_2}$ represent the realized values of the stochastic inputs to the second stage for scenario $\omega \in \Omega$. The function z is the *second-stage value function*, which encodes the cost of the recourse decision for a given first-stage solution x and realization ω. This value function is defined earlier in (1.8). We rewrite it

in the context of the stochastic problem. For any $b \in \mathbb{R}^{m_2}$, we have

$$z(b) = \inf\{q^\top y \mid y \in S_2(b)\}, \tag{RV}$$

where $S_2(b) = \{y \in \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2-r_2} \mid Wy = b\}$. For a given $\hat{b} \in \mathbb{R}^m$, $S_2(\hat{b})$ is the *second-stage feasible region* with respect to $\hat{b}$, defined by constraint matrix $W \in \mathbb{Q}^{m_2 \times n_2}$, and the second-stage objective function $q \in \mathbb{R}^{n_2}$, which represents the cost of recourse action. In general, we may have a stochastic matrix W_ω and a stochastic vector q_ω. In this case, one has to work with $|\Omega|$ individual second-stage value functions. Although our results on the MILP value function in Chapter 2 and the method we propose to solve the two-stage problem in Chapter 3 remain valid in this case, we assume a fixed W and q to be able to work with a single value function in the second-stage problem. We next illustrate that the two-stage problem (SP) has a desirable structure which can be exploited in the Benders' framework to solve these problems.

The structure of the function Ψ is closely related to the structure of ϕ. The following example illustrates this.

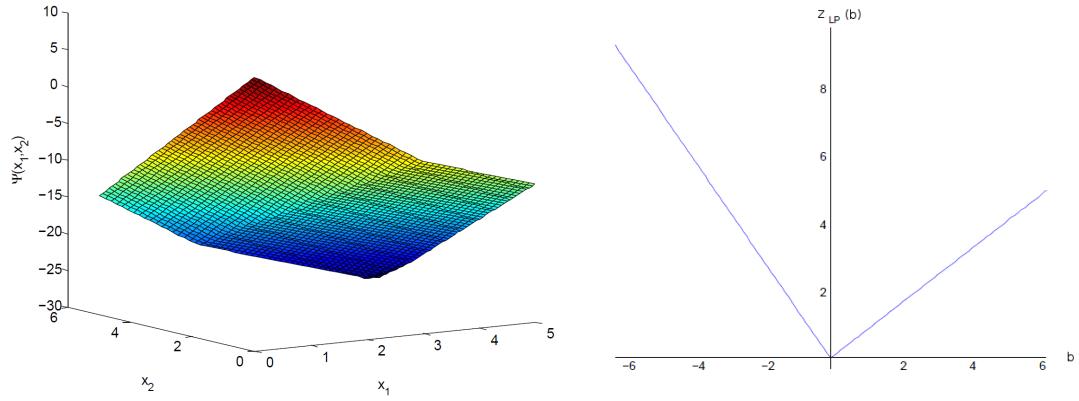
Example 1.4. Consider the following instance of a continuous two-stage problem ($r_1 = r_2 = 0$).

$$\begin{aligned} \min \quad & \Psi(x) = -3x_1 - 3.8x_2 + \sum_{\omega \in \Omega} 0.5z(h_\omega - 2x_1 - 0.5x_2), \\ \text{s.t.} \quad & x_1 \leq 5,\ x_2 \leq 5, \\ & x \in \mathbb{R}_+^2, \end{aligned} \tag{1.34}$$

where

$$\begin{aligned} z_{LP}(b) = \min \quad & 6y_1 + 4y_2 + 3y_3 + 4y_4 + 5y_5 + 7y_6 \\ \text{s.t.} \quad & 2y_1 + 5y_2 - 2y_3 - 2y_4 + 5y_5 + 5y_6 = b, \\ & y \in \mathbb{R}_+^6, \end{aligned} \tag{1.35}$$

where $\Omega = \{1, 2\}$, $h_1 = 6, h_2 = 12$. Figures 1.4a and 1.4b show the form of the objective function Ψ and second-stage value function z_{LP}, respectively, for Example 1.4. Both these functions are convex and can be approximated from below by subgradient inequalities in



(a) The objective function Ψ (b) The value function z

Figure 1.4: The objective function Ψ and second-stage value function z in Example 1.4.

the form of (1.29). Note the similarities in shape of the two functions. The structure of Ψ clearly derives from that of z_{LP}. □

In the next example, we illustrate the form of the value function in the pure integer case for which $r_2 = n_2$.

Example 1.5. Figure 1.5 shows two value functions resulting from the addition of integrality constraints to the problem of Example 1.4 for all variables in the second stage. The points plotted in blue (closed circles) are the finite values of the value function of the resulting recourse problem, while the function in red (dashed lines and open circles) is the value function of the recourse problem when the single linear constraint is relaxed to the inequality $2y_1 + 5y_2 - 2y_3 - 2y_4 + 5y_5 + 5y_6 \leq b$. □

In the pure integer case, the discrete nature of the problem is evident in the structure of the value function, which is only finite on a discrete set of points. In the inequality form, the value function remains constant over a countable number of regions of the domain. The discrete structure of the value function in this special case has been exploited in the development of several solution methods relying on combinatorial enumeration schemes (Ahmed et al., 2004; Kong et al., 2006; Trapp et al., 2013; Schultz et al., 1998). The structure above changes substantially when we have both continuous and integer variables in the second-stage problem. Let us modify the previous example such that we

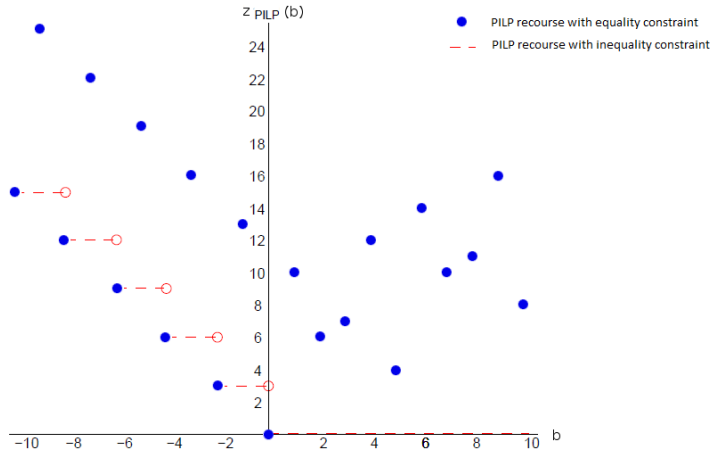


Figure 1.5: The value functions of two pure integer variations of (1.35)

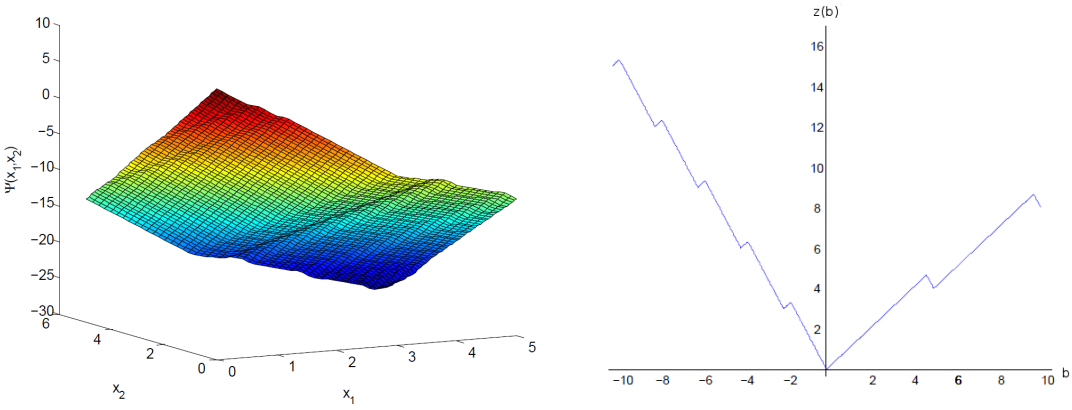
have a MILP in the second-stage.

Example 1.6. Consider the mixed integer variation of Example 1.4 where

$$\begin{aligned} z(b) = \min\ & 6y_1 + 4y_2 + 3y_3 + 4y_4 + 5y_5 + 7y_6 \\ \text{s.t. } & 2y_1 + 5y_2 - 2y_3 - 2y_4 + 5y_5 + 5y_6 = b \\ & y_1, y_2, y_3 \in \mathbb{Z}_+, y_4, y_5, y_6 \in \mathbb{R}_+. \end{aligned} \tag{1.36}$$

Figures 1.6a and 1.6b respectively show the objective function and second-stage value function, respectively, for this mixed integer variation of the problem from Example 1.4. □

We discussed earlier z_{LP} defined in (1.11) is a convex function in the continuous case. As a result, Benders' method can be applied straightforwardly in the same fashion we described in Section 1.2. This is true even with the introduction of integrality constraints in the first stage ($r_1 > 0$), as shown by Van Slyke and Wets (1969). Thus, when $r_2 = 0$, (SP) can be solved in principle using little modification in the Benders' method. When $r_2 > 0$, z is non-convex and discontinuous in general. As Example 1.6 should make clear, solution methods for the more general non-convex case must either exploit special structure, such as in Example 1.5, or rely on a more general class of functions for approximating the value function from below. In this thesis, we take the latter approach. To start, we need to



(a) The objective function Ψ in Example 1.6. (b) The value function z in Example 1.6.

Figure 1.6: The objective and second-stage value functions for Example 1.6.

describe the Benders' method with general lower bounding functions for the second-stage value function.

We assume ω is drawn from a given discrete and finite probability space $(\Omega, \mathcal{A}, P)$ so that ω represents which one of a finite number of explicitly enumerated scenarios is realized and p_ω represents the probability of such realization. Then, (1.33) is equivalent to

$$\Xi(x) = \sum_{\omega \in \Omega} p_\omega z(h_\omega - T_\omega x). \tag{1.37}$$

Therefore, the expectation in (SP) can be expressed as the sum of a finite number of terms, which allows for reformulation of (SP) as a large-scale MILP, the so-called *deterministic equivalent* problem:

$$\begin{aligned} \min\ & c^\top x + \sum_{\omega \in \Omega} p_\omega q^\top y_\omega \\ \text{s.t. } & Ax = b \\ & T_\omega x + W y_\omega = h_\omega \quad \forall \omega \in \Omega \\ & x \in \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1 - r_1}, y \in \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2}. \end{aligned} \tag{DE}$$

Figure 1.7 illustrates the structure of the constraints of (DE). From this figure, one can observe that the variable x acts as a "linking" or "complicating" variable. That is, if x is

$$\begin{pmatrix} A & & & \\ T_1 & W & & \\ & & \ddots & \\ T_{|\Omega|} & & & W \end{pmatrix} \begin{pmatrix} x \\ y_1 \\ \vdots \\ y_{|\Omega|} \end{pmatrix} = \begin{pmatrix} \tilde{b} \\ h_1 \\ \vdots \\ h_{|\Omega|} \end{pmatrix}$$

Figure 1.7: The constraints of (DE)

fixed, the constraint matrix can be separated into $|\Omega|$ individual blocks, each consisting of the the matrix W. Consider (DE) with a fixed $\hat{x}$. The resulting problem is

$$\begin{aligned} &\min \{c^\top \hat{x} + \sum_{\omega \in \Omega} p_\omega q^\top y_\omega\} \\ &\text{s.t. } W y_\omega = h_\omega - T_\omega \hat{x} \quad \forall \omega \in \Omega \\ &y \in \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2}. \end{aligned} \tag{1.38}$$

Clearly, (SP) can be decomposed into $|\Omega|$ independent subproblems. Naturally, the majority of the methods developed to solve (SP) are decomposition based methods that take advantage of the underlying block structure of the problem. As we discussed earlier, Benders' decomposition is designed in such a way to take advantage of the fact that the resulting problem after fixing the first-stage variables is relatively easy to solve. In this case, the remaining problem (1.38) is itself a separable problem. Due to the shape of the building blocks forming (DE)'s constraints, this method is also known as the *L-shaped method*. We will provide technical details of this method in Chapter 3. Here, we briefly explain the outline of the method when applied to (SP) and discuss its connection with the structure of the value function of the second-stage.

Consider (SP). We first begin by rewriting the problem (SP) as

$$\begin{aligned} &\min c^\top x + \theta \\ &\text{s.t. } \theta \geq \sum_{\omega \in \Omega} p_\omega z(h_\omega - T_\omega x) \\ &x \in S_1. \end{aligned} \tag{1.39}$$

Like the classical Benders' method, the idea is to approximate the right-hand side of the first set of constraints in (1.39) by a set $\mathcal{F}_\omega$ of dual functions for each scenario and to iteratively strengthen the approximation yielded by these dual functions through the generation of additional such functions. To form the master problem, we therefore replace the value function with such an approximation to obtain

$$\min\{c^\top x + \theta \mid \theta \geq \sum_{\omega \in \Omega} p_\omega \max_{f \in \mathcal{F}_\omega} f(h_\omega - T_\omega x)\}, \tag{1.40}$$

where $\mathcal{F}_\omega$ represents the set of all dual functions associated with scenario ω that have been generated so far. In iteration k, a strong dual function f_ω^k is produced for each scenario with respect to a proposed first-stage solution $x^k \in S_1$, the solution to the master problem in the previous iteration, by solving the dual problem

$$\max_f\{f(h_\omega - T_\omega x^k) \mid f(Wy) \leq q^\top y\}. \tag{1.41}$$

The collection of dual functions is then enlarged appropriately. With this approximation, the master problem after k iterations of the algorithm is

$$\begin{aligned} \min\ & c^\top x + \theta \\ \text{s.t. } & \theta \geq \sum_{\omega \in \Omega} p_\omega \max_{i=1,\dots,k} f_\omega^i(h_\omega - T_\omega x) \\ & x \in S_1. \end{aligned} \tag{1.42}$$

The algorithm terminates at iteration K if the solution to the master problem (x^*, θ^*) satisfies $\theta^* = \max_{i=1,\dots,K} f_\omega^i(h_\omega - T_\omega x^*)$, that is, the approximation of the expected recourse at the final iteration is exact.

It should be clear by now that applying Benders' decomposition to the general (SP) requires constructing dual functions for MILPs. Later in Section 2.6, we discuss that in practice, we are interested in finding dual functions that can be constructed as a by-product of integer optimization algorithms to solve scenario subproblems. We study the derivation of such dual functions and provide details about incorporating them into the

Benders' method in Sections 2.6, 3.3 and 3.4.

1.4 Contributions

In Chapter 2, we provide an in-depth review of the structure and properties of the general MILP value function. We extend previous results by demonstrating that the MILP value function has an underlying *discrete structure* similar to the value function of a *pure integer optimization problem* (PILP), even in the general case. This discrete structure emerges from separating the function into discrete and continuous parts, which in turn enables a representation of the function in terms of two discrete sets. We discuss the role of integer and continuous variables in the structure of the value function and in defining the discrete set. We provide a new representation of the value function using the discrete set and demonstrate that this representation can be applied to represent the value function of an LP and a PILP. Furthermore, we show the correspondence between the discrete set and the regions over which the MILP value function is continuous and convex. We show that the representation we provide can be constructed and propose an algorithm for doing so.

Using our earlier discrete representation of the MILP value function, we propose a deterministic reformulation of the two-stage problem in Chapter 3. We then describe a generalization of the classical Benders' method for solving two-stage mixed integer optimization problems and demonstrate that the algorithm is convergent if strong dual functions encoded in the branch-and- bound trees that are used to solve the second-stage subproblems are employed to approximate the second-stage value function. We demonstrate that it is possible to solve all second-stage subproblems with a single branch-and-bound tree and to refine the approximation using this tree. Finally, we show that this procedure allows us to conclude that there exists a single branch-and-bound tree that encodes the full value function.

In Chapter 4, we propose three warm-starting strategies to apply to the Generalized Benders' method we propose. We illustrate that each strategy leads to a different approx-

imation of the second-stage value function within the Benders' method. We use cut-pool management techniques to keep the size of the approximation manageable. Finally, we apply the algorithm to the problems in stochastic server location test set and analyze the performance of the algorithm under different warm-starting techniques.

1.5 Outline of Thesis

Chapter 2 includes the result of our work on the value function of a mixed integer optimization problem. Section 2.1 provides a review of duality in integer optimization. The discrete structure of the value function is examined in Section 2.2. Sections 2.3 and 2.4 respectively contain our results on the structural properties of the value function and a simplification of the Jeroslow formula that we propose by using our discrete representation. The proposed algorithm for construction is stated in Section 2.5. Finally, in Section 2.6 we review the upper and lower bounding methods to approximate the value function.

Chapter 3 contains our contributions in solving two-stage stochastic mixed integer optimization. we review the structural properties and solution methods of the continuous two-stage stochastic problems in Section 3.1. Section 3.2 included the literature review on the algorithms for the two-stage integer optimization problems. In Section 3.3, we provide a new formulation for this problem and discuss the implication of warm-starting the constructions approximating functions for the second-stage value function. Section 3.4 contains details and convergence results of the proposed algorithm to solve the two-stage mixed integer optimization problem.

We review MILP sensitivity analysis as well as the techniques to warm-start MILPs in Section 4.1. We overview current warm-starting techniques implemented in the MILP solver, SYMPHONY in Section 4.2. Section 4.3 contains the implementational details of the generalized Benders' algorithm, as well as alternative methods to construct approximations of the second-stage value function and several bunching and warm-starting strategies that can be used in the algorithm. Finally, we report our computational results

obtained by applying the algorithm to problems from the literature and SIPLIB.

Finally, Chapter 5 includes the summary of this work and remarks for future research.

Chapter 2

The Value Function of a Mixed Integer Linear Optimization Problem

Understanding and exploiting the structure of the value function of an optimization problem is a critical element of solution methods for a variety of important classes of multi-stage and multi-level optimization problems. Previous findings on the value function of a PILP have resulted in finite algorithms for constructing it, which have in turn enabled the development of solution methods for two-stage stochastic pure integer optimization problems (Schultz et al., 1998; Kong et al., 2006) and certain special cases of bilevel optimization problems (Bard, 1998). Studies of the value function of a general MILP, however, have not yet led to algorithmic advances. The goal of this chapter is to overview the previous work and provide new results on the structure and construction methods of the general MILP value function.

We start this section by reviewing the fundamental concepts that are necessary for the remainder of the chapter. We review MILP duality and the known results about the structure of the MILP value functions. In Section 2.2, we extend previous results by demonstrating that the MILP value function has an underlying *discrete structure*

similar to the PILP value function, even in the general case. We demonstrate that discrete structure emerges from separating the function into discrete and continuous parts, which in turn enables a representation of the function in terms of two discrete sets. In Section 2.3, we show how this discrete structure can explain certain structural properties of the MILP value function and use our representation to characterize regions over which the value function is convex and continuous.

We review lower and upper bounding approximation methods for the MILP value function in Section 2.6. Using our discrete representation, we develop an exact algorithm to construct the value function. We show this and the proof of finiteness of the algorithm in Section 2.5.

In the final section of this chapter, we show how that our discrete representation can explain several previously known properties of two well-known special cases of the MILP value function: the value function of a MILP with a single constraint and the value function of a PILP.

2.1 Overview

Recall that we defined a mixed integer optimization problem in (MILP) with

$$z = \inf_{x \in S} c^\top x, \tag{MILP}$$

where $c \in \mathbb{R}^n$ is the objective function vector and $S = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = b\}$ is the feasible region, described by $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{R}^m$, and a scalar r indicating the number of integer variables. We also defined the value function of a MILP in (1.8) in

$$z(b) = \inf_{x \in S(b)} c^\top x \quad \forall b \in B, \tag{2.1}$$

where for $b \in \mathbb{R}^m$, $S(b) = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = \hat{b}\}$ and $B = \{b \in \mathbb{R}^m \mid S(b) \neq \emptyset\}$. We assumed by convention that $z(0) = 0$. Let us introduce a few further notation that will be used widely in this chapter to find the discrete structure of z. We introduce

the discrete analogue to $S(b)$ and B by letting $S_I(b) = \{x_I \in \mathbb{Z}^r_+ : A_I x_I = b\}$ and $B_I = \{b \in \mathbb{R}^m : S_I(b) \neq \emptyset\}$. Finally, we let $S_I = \cup_{b \in B} S_I(b)$.

In Chapter 1, we defined a dual function in Definition 1.1 and showed that it follows from the LP duality theory that linear functions can be strong dual functions for the LP value functions. In what comes next, we overview major results in the duality theory for integer optimization problems. Mainly, we introduce certain classes of functions that can be used as dual functions for the MILP value function and discuss several methods for their construction.

The definition of dual functions in (1.13) is rather broad and does not impose a particular structure on the dual function. When dual functions are used within solution methods, such as the Benders' method we discussed in the previous chapter, then it is desirable for the dual function to be computable in practice. We saw earlier in Chapter 1 that linear functions are not dual to the MILP value function in general. The next natural class of functions to consider is convex functions.

The Subadditive Dual Let us once more consider the MILP value function (1.20). In Figure 2.1, the best piecewise linear convex function that is dual to the MILP is plotted along with the original value function. As one can observe in the figure, this function is strong only at the lower break points of the value function. The weak approximation provided by this convex function elsewhere is not a surprise, given that the MILP value function is non-convex and can clearly be best approximated by a non-convex function.

It turns out that the optimal convex dual function is in fact the value function of the LP relaxation of the MILP. In the case of the MILP (1.20), this problem is

$$\begin{aligned} z_{LP}(b) = \inf\ & 4x_1 + x_2 + 4x_3 + 6x_4 + 7x_5 \\ \text{s.t. } & 2x_1 - 2x_2 + x_3 + 2x_4 - 7x_5 = b \\ & x_1, x_2, x_3, x_4, x_5 \in \mathbb{R}_+. \end{aligned} \tag{2.2}$$

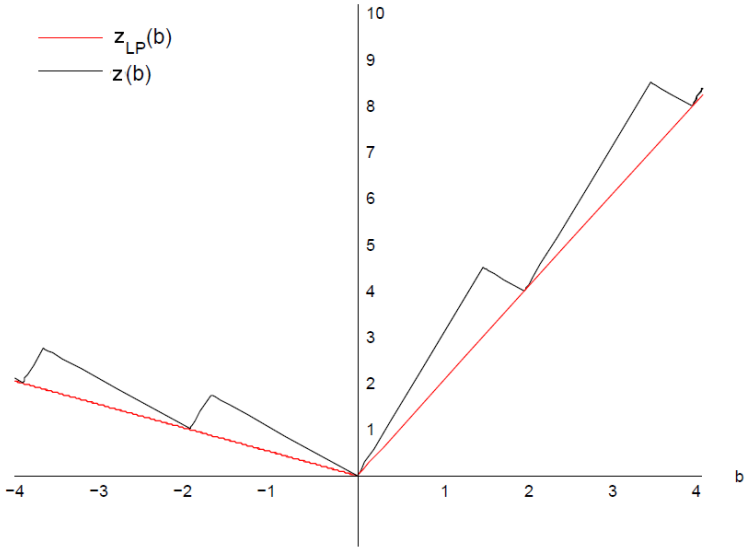


Figure 2.1: The value functions (1.20) and (2.2).

which is equivalent to

$$\begin{aligned} z_{LP}(b) = \sup\ & b\nu \\ \text{s.t.}\ & -0.5 \leq \nu \leq 2 \\ & \nu \in \mathbb{R}. \end{aligned} \tag{2.3}$$

(2.3) can be explicitly written as

$$z_{LP}(b) = \begin{cases} 2b & \text{if } b \geq 0 \\ -0.5b & \text{if } b < 0 \end{cases}$$

which is precisely the convex function plotted in Figure 2.1.

Searching for candidate classes of functions, Johnson (1973) first proposed the idea of restricting to the class of *subadditive* functions.

Definition 2.1. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *subadditive* on $\mathbb{R}^n$ if $f(x_1 + x_2) \leq f(x_1) + f(x_2)$ for all $x_1, x_2 \in \mathbb{R}^n$ such that $x_1 + x_2 \in \mathbb{R}^n$.

The strong motivation for considering this class is that the value function itself is a subadditive function on B. Therefore, there is always a dual function to the dual problem (1.14) that is subadditive: the value function.

Proposition 2.1. *The value function* (1.8) *is subadditive on* B.

Proof. Consider $b_1, b_2 \in B$. Let $z(b_1) = c^\top x_1$ and $z(b_2) = c^\top x_2$ for some $x_1 \in S(b_1)$ and $x_2 \in S(b_2)$. We have $(b_1 + b_2) \in S(b_1 + b_2)$, therefore $z(b_1 + b_2) \leq c^\top b_1 + c^\top b_2 = z(b_1) + z(b_2)$. □

Johnson showed that for a feasible MILP, we have

$$\begin{array}{ll} \inf c^\top x & \sup F(\hat{b}) \\ Ax = \hat{b} & = \quad F(A^j) \leq c_j \quad \forall j \in I \\ x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ & \bar{F}(A^j) \leq c_j \quad \forall j \in C \\ & F(0) = 0, F \text{ subadditive.} \end{array} \tag{2.4}$$

where A^j is the j^{th} column of A and the function $\bar{F}$ is defined as

$$\bar{F}(b) = lim \sup_{\delta \to 0^+} \frac{F(\delta b)}{\delta} \quad \forall b \in \mathbb{R}^m. \tag{2.5}$$

The second problem is known as the *subadditive dual problem.*

Definition 2.2. The *directional derivative* of a function $f : \mathbb{R}^n \to \mathbb{R}$ in the direction p at point x_0 is given by

$$f'(x_0, p) = \lim_{h \to 0} \frac{f(x_0 + hp) - f(x_0)}{h}.$$

In the case the limit exists, the function is called directionally differentiable at x_0 in the direction p.

The directional derivative gives the rate of change of the function, moving through x_0 in a given direction p and provides a useful characterization when the function is not continuously differentiable at x_0. The function $\bar{F}$ is the *upper d-directional derivative* of F at zero, first introduced by Gomory and Johnson (1972a,b). Intuitively, $\bar{F}$ provides an upper bound to F near zero and ensures that a function that is feasible to (2.4) has

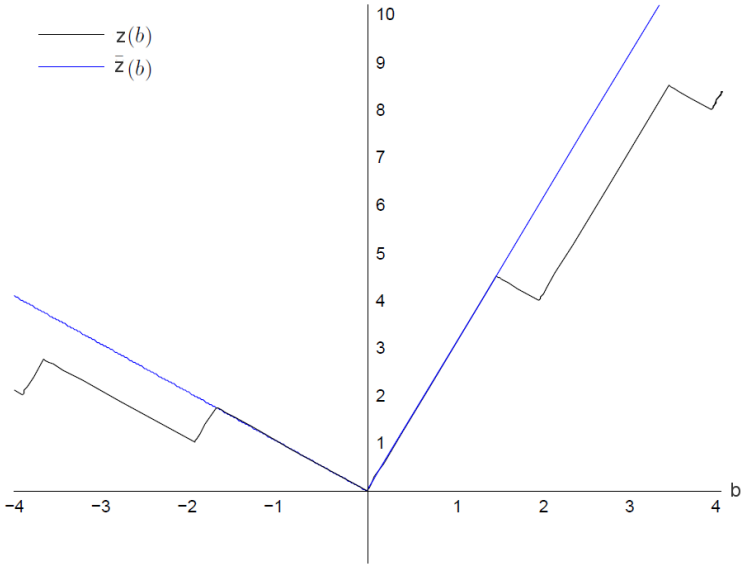


Figure 2.2: The upper d-directional derivative of (1.20).

gradients that do not exceed the gradients of the value function near zero. This was formally shown in a subsequent paper by Johnson (1974).

Proposition 2.2. *If F is a subadditive function with $F(0) = 0$, then for any $b \in \mathbb{R}^m$ with $\bar{F}(b) \leq \infty$ and any $\lambda \geq 0$, we have $F(\lambda b) \leq \lambda \bar{F}(b)$.*

Example 2.1. Consider the MILP value function (1.20). This function and its upper d-directional derivative $\bar{z}$ are plotted in Figure 2.2, where $\bar{z}$ is defined as

$$\bar{z}(b) = \begin{cases} 3b & \text{if } \ b \geq 0 \\ -b & \text{if } \ b < 0 \end{cases}$$

The function $\bar{z}$ is an upper bounding function to z near zero. One may note that this function is identical to the value function of the LP (1.1), which consists of only the continuous variables of the MILP. We show that this is not a coincidence and provide further details on it in Section 2.2.

Subadditve dual functions are extremely important to the theory of duality for MILPs. Subadditive dual functions not only provide lower bounds for the MILP instance with the

right-hand side $\hat{b}$, but also they allow to carry over several properties of the LP duality to the MILP case such as strong and weak duality and complementary slackness. We state these results next.

Theorem 2.1. *(weak duality by Jeroslow (1978, 1979)) If F is a feasible solution to the subadditive dual problem (2.4) and $\hat{x}$ is a feasible solution to (MILP) with its right-hand side fixed at $\hat{b} \in B$, then $F(\hat{b}) \leq c^\top \hat{x}$.*

Proof. Consider an arbitrary $\hat{b} \in B$ and $\hat{x}$ such that $\hat{x} \in S(\hat{b})$. From the subadditivity of F we have

$$F(\hat{b}) = F(A\hat{x}) \leq F(\sum_{j \in I} A^j \hat{x}_j) + F(\sum_{j \in C} A^j \hat{x}_j). \tag{2.6}$$

Since $F(0) = 0$ and $x_j \in \mathbb{Z}_+$ for all $j \in I$ and F is subadditive, we have

$$F(\sum_{j \in I} A^j \hat{x}_j) \leq \sum_{j \in I} F(A^j)\hat{x}_j. \tag{2.7}$$

Similarly, since $\bar{F}(0) = 0$, and $F(A^j x_j) \leq \bar{F}(A^j)x_j$ for $x_j \in \mathbb{R}_+$, $j \in C$, and F is subadditive we have

$$F(\sum_{j \in C} A^j \hat{x}_j) \leq \sum_{j \in C} \bar{F}(A^j)\hat{x}_j. \tag{2.8}$$

Together, we have

$$F(b) \leq \sum_{j \in I} F(A^j)\hat{x}_j + \sum_{j \in C} \bar{F}(A^j)\hat{x}_j \leq c\hat{x}, \tag{2.9}$$

where the last inequality holds since we have $F(A^j) \leq c_j, j \in I$ and $\bar{F}(A^j) \leq c_j, j \in C$ by feasibility of F for the subadditive dual problem and $x_j \geq 0$ by its feasibility for the primal MILP. □

The proof of the strong duality and complementary slackness require some extra machinery which we will not provide here for the sake of space and refer to the original texts in (Jeroslow, 1978, 1979; Johnson, 1974). The next result addresses the necessary and sufficient conditions on the infeasibility and unboundedness of the primal and dual problems. These results are analogous to those in LP duality.

Proposition 2.3. *The following statements hold for the primal problem* (MILP) *and its subadditve dual problem* (2.4).

i (MILP) *is unbounded if and only if* $b \in B$ *and* $z(0) \leq 0$.

ii *The dual problem is infeasible if and only if* $z(0) < 0$.

iii *If the primal problem (respectively, the dual) is unbounded, then the dual problem (respectively, the primal) is infeasible.*

iv *If the primal problem (respectively, the dual) is infeasible, then the dual problem (respectively, the primal) is infeasible or unbounded.*

Next, we state the strong duality theorem for MILPs, which is due to Jeroslow (1978, 1979).

Theorem 2.2. *(strong duality) If either the primal problem* (MILP) *or the dual problem* (2.4) *has an optimal value, then there exists an optimal feasible solution* x^* *to the primal problem and an optimal dual function* F^* *to the dual problem for which* $c^\top x^* = F^*(b)$.

Finally, we arrive at the complementary slackness result. Complementary slackness is a significant result as it provides a certificate of optimality for the primal-dual pair (MILP) and (2.4).

Theorem 2.3. *(Jeroslow, 1978, 1979; Bachem and Schrader, 1980) Let* x^* *be a feasible solution to* (MILP) *with its right-hand side fixed at a given* $\hat{b}$ *and* F^* *be an optimal solution the subadditive dual problem* (2.4). *Then,* x^* *and* F^* *are optimal if and only if*

$$\begin{aligned} x_j^*(c_j - F^*(A^j)) = 0, \quad \forall j \in I \\ x_j^*(c_j - \bar{F}^*(A^j)) = 0, \quad \forall j \in C. \end{aligned} \tag{2.10}$$

Although the stated results provide the theoretical fundamentals and answer several important questions about duality of integer optimization problems, we still need to find methods to compute and encode dual functions for MILPs. We review some methods to construct feasible, and in some cases optimal, dual functions in section 2.5.

The Chvátal Representation Blair and Jeroslow (1982) first showed that the value function of a PILP is a *Gomory function* that can be derived by taking the maximum of finitely many subadditive functions. In a subsequent work, they extended their earlier results in (Blair and Jeroslow, 1984) and identified a subclass of Gomory functions called *Chvátal functions* to which the general MILP value function belongs.

Definition 2.3. The *Gomory* functions are the smallest class $\mathscr{G}^m$ of functions such that

i For any $\alpha \in \mathbb{Q}^m$, $f(b) = \alpha b$ is in $\mathscr{G}^m$.

ii If $f_1, f_2 \in \mathscr{G}^m$ and $\alpha, \beta \in \mathbb{Q}^m_+$, then $\alpha f_1 + \beta f_2 \in \mathscr{G}^m$.

iii If $f \in \mathscr{G}^m$, then $\lceil f \rceil \in \mathscr{G}^m$.

iv If $f_1, f_2 \in \mathscr{G}^m$, then $\max\{f_1, f_2\} \in \mathscr{G}^m$.

The Chávatal functions are the smallest class satisfying i–iii.

The following two results show the connection between Gomory and Chvátal functions and that both of the classes have the subadditivity property.

Proposition 2.4. *(Blair and Jeroslow, 1982) If g is a Gomory function, then there is a finite number of Chvátal functions $f_1, \ldots, f_N$ such that*

$$g = \max\{f_1, \ldots, f_N\}. \tag{2.11}$$

Proposition 2.5. *(Blair and Jeroslow, 1982) Any Chvátal or Gomory function is subadditive.*

The main results that address the connection between the PILP and MILP value functions and Gomory and Chvátal functions are Theorems 2.4 and 2.6. First, we state the lemmas needed for the main results. The first lemma below shows that the convex hull of S can be represented by subadditive functions and this representation is finite for the case of a PILP.

Lemma 2.1. *(Blair, 1978; Bachem and Schrader, 1980; Wolsey, 1981b) For any $b \in B$ we have*

$$conv(S(b)) = \{x \in \mathbb{R}^n_+ \mid \sum_{j \in I} F(A^j)x_j + \sum_{j \in C} \bar{F}(A^j)x_j \geq F(b), F \text{ subadditive}, F(0) = 0\}. \tag{2.12}$$

In the case of a PILP, there exist finitely many subadditive functions $F_i, i = 1, \ldots, k$ such that

$$conv(S(b)) = \{x \in \mathbb{R}^n_+ \mid \sum_{j \in N} F_i(A^j)x_j \geq F_i(b), i = 1, \ldots, k\}. \tag{2.13}$$

Knowing that subadditive functions can represent the convex hull of solutions to PILPs and that Chvátal functions are subadditive, Schrijver (1980) showed that for PILPs, Chvátal functions can in fact be used to represent the convex hull of solutions. We state this lemma first and use it to arrive at Theorem 2.4, which shows that every value function of a PILP can be represented as a Gomory function.

Lemma 2.2. *(Schrijver, 1980) The subadditive functions in (2.13) can be taken to be Chvátal function.*

This lemma is used to represent the value function of a PILP as a Gomory function.

Theorem 2.4. *(Blair and Jeroslow, 1982) There always exists a Gomory function g such that $g(b) = z(b)$ for all $b \in B$, where z is the value function of a PILP with $z(0) = 0$.*

It is worth noting that for any Gomory function g, there is always a PILP such that g coincides with the value function of the PILP. We state this formally next.

Theorem 2.5. *(Blair and Jeroslow, 1986) Consider (MILP) with $C = \emptyset$. Then, for any Gomory function g there are A and c such that $g(b)$ is the optimal objective value to the problem for all vectors $b \in B$.*

Subsequently, Blair and Jeroslow (1984) extended their results from the PILP to the MILP case by showing that every MILP value function is a minimum of finitely many Gomory functions.

Theorem 2.6. *For a MILP, there is a finite number of Gomory functions $g_1, \dots, g_N$ such that for all b such that $z(b) < \infty$, we have*

$$z = \min\{g_1, \dots, g_N\}. \tag{2.14}$$

Although the efforts made to characterize the value function of a MILP resulted in very significant contributions, the results remained mainly theoretical. However, a closed form representation was not achieved until a decade later in a subsequent work of Blair (1995). The so-called *Jeroslow Formula* represents the MILP value function as collection of Gomory functions with linear correction terms. We investigate this representation extensively and study its relationship with the representation we propose in Section 2.4.

2.2 A Discrete Representation

The value function (1.8) can be written as

$$z(b) = \inf_{(x_I, x_C) \in S(b)} c_I^\top x_I + c_C^\top x_C \quad \forall b \in B. \tag{MVF}$$

Recall that for any $D \subseteq N$ and a vector v indexed on N, we established the denotation v_D to represent the sub-vector consisting of the corresponding components of v. Therefore, here we have x_C and x_I respectively corresponding to continuous and integer variables.

To understand the MILP value function, it is important to first understand the structure of the value function of the LP arising from (MILP) by fixing the values of the integer variables. We call this problem the *continuous restriction* (CR) w.r.t a given $\hat{x_I} \in S_I$. Its value function is given by

$$\begin{aligned} \bar{z}(b; \hat{x_I}) = c_I^\top \hat{x_I} + \inf\, & c_C^\top x_C \\ \text{s.t. } & A_C x_C = b - A_I \hat{x_I} \\ & x_C \in \mathbb{R}_+^{n-r}. \end{aligned} \tag{CR}$$

In addition to the notations we introduced for (2.1), for a given $\hat{x_I} \in S_I$, we let $S(b, \hat{x_I}) =$

$\{x_C \in \mathbb{R}^{n-r}_+ : A_C x_C = b - A_I \hat{x_I}\}$. As before, we let $\bar{z}(b; \hat{x_I}) = \infty$ if $S(b, \hat{x_I}) = \emptyset$ for a given $b \in B$ and $\bar{z}(b; \hat{x_I}) = -\infty$ if the function value is unbounded. As we will show formally in Proposition 2.9, it is evident that for any $\hat{x_I} \in S_I$, $\bar{z}(\cdot; \hat{x_I})$ bounds the value function from above, which is the reason for the notation.

Example 2.2. Consider the MILP value function defined by

$$\begin{aligned} z(b) = \inf\ & 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 \\ \text{s.t. } & 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 = b \\ & x_1, x_2, x_3 \in \mathbb{Z}_+,\ x_4, x_5 \in \mathbb{R}_+. \end{aligned} \tag{2.15}$$

Figure 2.3 shows this non-convex, non-concave piecewise polyhedral function. □

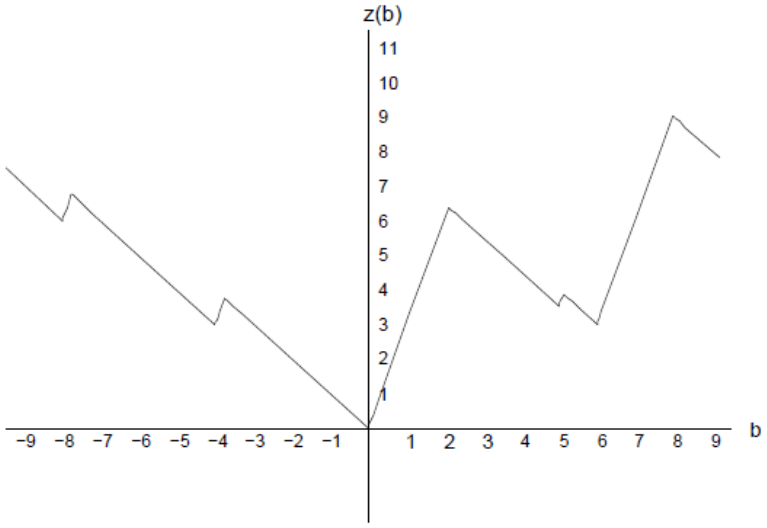


Figure 2.3: MILP Value Function of (2.15).

Although the MILP with which the value function in Example 2.2 is associated has only a single constraint, the structure of the function is already quite complex. Nevertheless, the function does have an obvious regularity to it. We investigate this in the remainder of the section.

Earlier in Section 1.2, we introduced this value function in the general form and reviewed its convexity and differentiability properties. When $\hat{x_I} = 0$ in (CR), the resulting

function is in fact the value function of a general LP, since A_C is itself an arbitrary matrix. In the remainder of the section, we consider this important special case and define

$$\begin{aligned} z_C(b) = \ & \inf c_C^\top x_C \\ & \text{s.t. } A_C x_C = b \\ & x_C \in \mathbb{R}_+^{n-r}. \end{aligned} \tag{LVF}$$

We let $\mathcal{K}$ be the polyhedral cone that is the positive linear span of A_C, i.e., $\mathcal{K} = \{\lambda_1 A^{r+1} \ldots + \lambda_{n-r} A^n : \lambda_1, \ldots, \lambda_{n-r} \geq 0\}$. As we discuss later, this cone is the set of right-hand sides for which z_C is finite and plays an important role in the structure of both the LP and MILP value functions. The following example illustrates a continuous restriction.

Example 2.3. Consider the MILP

$$\begin{aligned} & \inf \ 2x_1 + 6x_2 + 7x_3 + 5x_4 \\ & \text{s.t. } x_1 + 2x_2 - 7x_3 + x_4 = b \\ & x_1 \in \mathbb{Z}_+, x_2, x_3, x_4 \in \mathbb{R}_+. \end{aligned} \tag{2.16}$$

The value functions of the continuous restriction w.r.t. $x_1 = 0$ and $x_1 = 1$ are plotted in Figure 2.4. □

Note that in the example just given, $\bar{z}(\cdot; 1)$ is simply a translation of z_C. As we will explore in more detail later, this is true in general, so that for $\hat{x_I} \in S_I$, we have

$$\bar{z}(b; \hat{x_I}) = c_I^\top \hat{x_I} + z_C(b - A_I \hat{x_I}) \ \forall b \in B.$$

Thus, the following results can easily be generalized to the continuous restriction functions w.r.t. points other than the origin.

We shall now more formally analyze the structure of z_C. We first present a representation due to Blair and Jeroslow (1977), who characterized the LP value function in terms of its epigraph. Let $\mathcal{L} = \text{epi}(z_C)$.

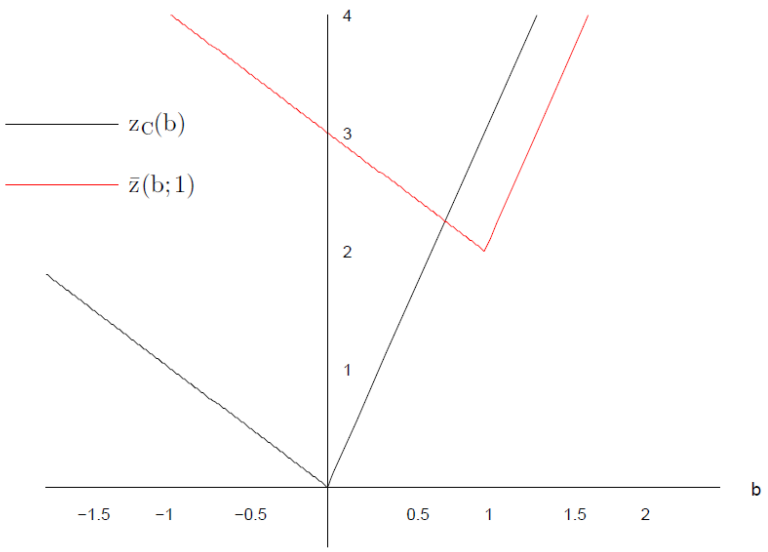


Figure 2.4: The value function of the continuous restriction of (2.16) and a translation.

Proposition 2.6. *(Blair and Jeroslow, 1977) The value function of z_C is a convex polyhedral function and its epigraph $\mathcal{L}$ is the convex cone*

$$cone\{(A^{r+1}, c_{r+1}), (A^{r+2}, c_{r+2}), \ldots, (A^n, c_n), (0, 1)\}.$$

The above description of the LP value function in terms of a cone is not computationally convenient for reasons that will become clear. We can derive a more direct characterization of the LP value function by considering the structure of the dual of (LVF) for a fixed right-hand side $\hat{b} \in \mathbb{R}^m$. In particular, this dual problem is

$$\sup_{\nu \in S_D} \hat{b}^\top \nu, \tag{2.17}$$

where $S_D = \{\nu \in \mathbb{R}^m : A_C^\top \nu \leq c_C\}$. Note that our earlier assumption that $z(0) = 0$ implies $S_D \neq \emptyset$. From strong duality, we have that $z_C(\hat{b}) = \sup_{\nu \in S_D} \hat{b}^\top \nu$ when $S_D \neq \emptyset$. If the LP with right-hand side $\hat{b}$ has a finite optimum, then

$$z_C(\hat{b}) = \sup_{\nu \in S_D} \hat{b}^\top \nu = \sup_{i \in K} \hat{b}^\top \nu^i, \tag{2.18}$$

where$\{\nu^i\}_{i\in K}$ is the set of extreme points of S_D indexed by set K. When S_D is unbounded, let its set of extreme directions $\{d^j\}_{j\in L}$ be indexed by set L. In the case that the solution to the problem is unbounded, for some $j \in L$, we have $\hat{b}^\top d^j > 0$ and $z_C(\hat{b}) = +\infty$. We can therefore obtain a representation of the cone $\mathcal{L}$ as

$$\{(b, z) \in \mathbb{R}^{m+1} : b^\top \nu^i \leq z,\ b^\top d^j \leq 0, i \in K, j \in L\}.$$

Let $\mathcal{E}$ be the set of index sets of the nonsingular square sub-matrices of A_C corresponding to dual feasible bases. That is, $E \in \mathcal{E}$ if and only if $\exists i \in K$ such that $A_E^\top \nu^i = c_E$. Abusing notation slightly, we denote this (unique) ν^i by ν_E in order to be consistent with the literature. The cone $\mathcal{L}$ has an extreme point if and only if there exist $m+1$ linearly independent vectors in the set $\{(\nu^i,\ -1) : i \in K\} \cup \{(d^j,\ 0) : j \in L\}$. It is easy to show that in this case, the origin is the single extreme point of $\mathcal{L}$ and all dual extreme points are optimal at the origin, i.e., $\nu_E^\top 0 = c_E^\top A_E^{-1} 0 = z_C(0) = 0$ for all $E \in \mathcal{E}$. Conversely, when $\mathcal{L}$ has an extreme point, it must be the single point at which all the inequalities in the description of $\mathcal{L}$ are binding.

We stated that z_C is continuous and convex in Proposition 1.2. From the same result we also have that if z_C is differentiable at $\hat{b} \in K$, then the gradient of z_C at $\hat{b}$ is the unique $\nu \in S_D$ such that $z_C(\hat{b}) = \hat{b}^\top \nu$. The next result shows the relationship between points of nondifferentiability of this value function and the primal and dual optimal solutions at such points.

Consider a right-hand side $b \in B$ for which the optimal solution to the corresponding LP is non-degenerate. Let the (unique) optimal basis and optimal dual solution be A_E and ν_E, respectively, for some $E \in \mathcal{E}$. As a result of the unchanged reduced costs, under a small enough perturbation in b, A_E and ν_E remain the optimal basis and dual solution to the new problem. Hence, the function is affine in a neighborhood of b and differentiability of the LP value function at b follows. On the other hand, whenever the value function is non-differentiable, the problem has multiple optimal dual solutions and every optimal basic solution to the primal problem is degenerate. These observations result in the

following characterization of the differentiability of the LP value function.

Proposition 2.7. *(Bazaraa et al., 1990) If $\hat{b} \in \text{int}(\mathcal{K})$ is a point of non-differentiability of z_C, then there exist $\nu^1, \nu^2, \ldots, \nu^s \in S_D$ with $s > 1$ such that $z_C(\hat{b}) = \hat{b}^\top \nu^1 = \hat{b}^\top \nu^2 = \ldots = \hat{b}^\top \nu^s$ and every optimal basic solution to the associated LP with right-hand side $\hat{b}$ is degenerate.*

Example 2.4. In (2.16), we have

$$z_C(b) = \sup\{\nu b : -1 \leq \nu \leq 3, \nu \in \mathbb{R}\} = \begin{cases} 3b & \text{if } \ b \geq 0 \\ -b & \text{if } \ b < 0 \end{cases}$$

Then, $\mathcal{E} = \{\{1\}, \{2\}, \{3\}\}$ with $A_{\{1\}} = 2$, $A_{\{2\}} = -7$, and $A_{\{3\}} = 1$. The corresponding basic feasible solutions to the dual problem are 3, -1, and 5 respectively. If the value function is differentiable at $\hat{b} \in \mathbb{R}$, then its gradient at $\hat{b}$ is either -1 or 3. These extreme points describe the facets of the convex cone $\mathcal{L} = cone\{(2,6), (-7,7), (1,5), (0,1)\} = \{(b,z) \in \mathbb{R}^2 : z \geq 3b, \ z \geq -b\}$. Note that we can conclude that fixing x_1 to 0 in (2.16) does not affect its value function. Finally, note that $\mathcal{K} = \mathbb{R}$, i.e., $z_C(b) < \infty$ for all $b \in \mathbb{R}$. □

We have so far examined the LP value function arising from restricting the integer variables to a fixed value and discussed that such a value function inherits the structure of a general LP value function. The LP value function, though it arises from a continuous optimization problem, has a discrete representation in terms of the extreme points and extreme directions of its dual. In the next section, we study the effect of the addition of integer variables.

Our goal in the rest of this section is to derive a discrete representation of a general MILP value function building from the results of the previous section. We observe that the MILP value function is the minimum of a countable number of translations of z_C and thus retains the same local structure as that of the continuous restriction (CR). By characterizing the set of points at which these translations occur, we arrive at Theorem 2.7, our discrete characterization. From the MILP value function (2.15) and its

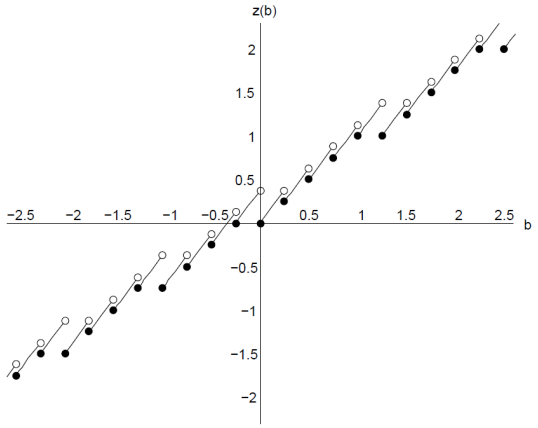


Figure 2.5: Value Function (2.19).

continuous restriction w.r.t $\hat{x} = 0$, plotted respectively in Figures 2.3 and 2.4, we can observe that when integer variables are added to the continuous restriction, many desirable properties of the LP value function, such as convexity and continuity, may be lost. The value function in this particular example remains continuous, but as a result of the added integer variables, the function becomes piecewise linear and additional points of non-differentiability are introduced. In general, however, even continuity may be lost in some cases. Let us consider another example.

Example 2.5. Consider

$$\begin{aligned} z(b) = \inf\ & x_1 - \frac{3}{4}x_2 + \frac{3}{4}x_3 + \frac{5}{2}x_4 \\ \text{s.t. } & \frac{5}{4}x_1 - x_2 + \frac{1}{2}x_3 + \frac{1}{3}x_4 = b \\ & x_1, x_2 \in \mathbb{Z}_+, x_3, x_4 \in \mathbb{R}_+. \end{aligned} \tag{2.19}$$

Figure 2.5 shows this value function. As in (2.15), the value function is piecewise linear; however, in this case, it is also discontinuous. More specifically, it is a lower semi-continuous function[1] . The next result formalizes these properties. □

Proposition 2.8. *(Nemhauser and Wolsey, 1988; Bank et al., 1983) The MILP value function* (MVF) *is lower semi-continuous, subadditive, and piecewise polyhedral over B.*

[1] A function $f : \mathbb{R}^n \to \mathbb{R} \cup \pm\infty$ is called *lower semi-continuous* at a point of its domain $\hat{x}$ if $\liminf_{x \to \hat{x}} f(x) \geq f(\hat{x})$.

Characterizing a piecewise polyhedral function amounts to determining its points of discontinuity and non-differentiability. In the case of the MILP value function, these points are determined by properties of the continuous restriction that has already been introduced and a second problem, called the *integer restriction*, obtained by fixing the continuous variables to zero. This problem is defined as follows.

$$\begin{aligned} z_I(b) = \ & \inf c_I^\top x_I \\ & s.t.\ A_I x_I = b \\ & x_I \in \mathbb{Z}^r_+. \end{aligned} \tag{IR}$$

The role of the integer restriction in characterizing the value function will become clear shortly, but we first need to introduce some additional concepts. Recalling that the continuous restriction for any $\hat{x_I} \in S_I$ can be expressed as $\bar{z}(b; \hat{x_I}) = c_I^\top \hat{x_I} + z_C(b - A_I \hat{x_I})$, we obtain the following representation of (MVF) in terms of the continuous restriction:

$$z(b) = \inf_{x_I \in S_I} c_I^\top x_I + z_C(b - A_I x_I) = \inf_{x_I \in S_I} \bar{z}(b; x_I) = \inf_{\hat{b} \in B_I} z(\hat{b}) + z_C(b - \hat{b}) \ \ \forall b \in B. \tag{2.20}$$

This shows that the MILP value function can be represented as a countable collection of value functions of continuous restriction functions arising from translations of the LP value function z_C. Describing the value function consists essentially of characterizing the minimal set of points at which such translations must be located to yield the entire function. The points at which translations may potentially be located can be thought of as corresponding to vectors $x_I \in S_I$, as in the first two equation in (2.20), though more than one member of S_I may specify the same location. Equivalently, we can also consider describing the function simply by specifying its value at points in B_I, as in the third equation above, which makes the correspondence one-to-one. Despite being finite under the assumption that B_I is finite, this characterization is nevertheless still quite impractical, as both S_I and B_I may be very large. As one might guess, it is not necessary to consider all members of B_I in order to obtain a complete representation. Later in this section, we characterize the subset of B_I necessary to guarantee a complete description.

This characterization provides a key insight that leads eventually to our algorithm for construction.

Before moving on, we provide some examples that illustrate how the structure of z_C influences the structure of (MVF). First, we examine the significance of the domain of z_C in the structure and the continuity of the MILP value function with the following example.

Example 2.6. Consider again the value function (2.19). Its continuous restriction w.r.t $\hat{x_I} = 0$ is

$$\begin{aligned} z_C(b) = \inf \ & \frac{3}{4}x_1 + \frac{5}{2}x_2 \\ \text{s.t.} \ & \frac{1}{2}x_1 + \frac{1}{3}x_2 = b \\ & x_1, x_2 \in \mathbb{R}_+. \end{aligned}$$

Equivalently,

$$z_C(b) = \sup\{\nu b : \nu \leq \frac{3}{2}, \nu \in \mathbb{R}\}. \tag{2.21}$$

Here, the positive linear span of $\{\frac{1}{2}, \frac{1}{3}\}$ is $\mathcal{K} = \mathbb{R}_+$. We also have $z_C(b) = \frac{3}{2}b$ for all $b \in \mathcal{K}$. The gradient of $z_C(b)$ at any $b \in \mathbb{R}_+ \backslash \{0\}$ is $\frac{3}{2}$, which is the extreme point of the feasible region of (2.21). Note that for $b \in \mathbb{R}_-$, $z_C(b) = +\infty$ because the continuous restriction w.r.t the origin is infeasible whenever $b \in \mathbb{R}_-$ and its corresponding dual problem is therefore unbounded. However, in the modification of this problem in (2.19), we have $B = \mathbb{R}$, while $\mathcal{K}$ remains $\mathbb{R}_+$. This is because the additional integer variables result in translations of $\mathcal{K}$ into $\mathbb{R}_-$. These translations result in the discontinuity of the value function observed in (2.19). □

The next result shows that the continuous restriction with respect to any fixed $\hat{x_I} \in S_I$ bounds the value function from above, as it is a restriction of the value function by definition.

Proposition 2.9. *For any $\hat{x_I} \in S_I$, $\bar{z}(\cdot; \hat{x_I})$ bounds z from above.*

Proof. For $\hat{x_I} \in S_I$ we have

$$\bar{z}(b; \hat{x_I}) = c_I^\top \hat{x_I} + z_C(b - A\hat{x_I}) \geq \inf_{x_I \in S_I} c_I^\top x_I + z_C(b - A_I x_I) = z(b).$$

□

The second result shows that the continuous restriction with respect to the origin coincides with the value function z over the intersection of $\mathcal{K}$ and some open ball centered at the origin. We denote an open ball with radius $\epsilon > 0$ centered at a point d by $\mathcal{N}_\epsilon(d)$.

Proposition 2.10. *There exists $\epsilon > 0$ such that $z(b) = z_C(b)$ for all $b \in \mathcal{N}_\epsilon(0) \cap \mathcal{K}$.*

Proof. At the origin, we have $z(0) = 0$ with a corresponding optimal solution to the MILP being $(x_I^*, x_C^*) = (0, 0)$. For a given $\hat{b} \in \mathbb{R}$, as long as there exists an optimal solution $\hat{x_I}$ to the MILP with right-hand side b such that $\hat{x_I} = 0$, we must have $z(\hat{b}) = z_C(\hat{b})$. Therefore, assume to the contrary. Then for every $\epsilon > 0$, $\exists \tilde{b} \in \mathcal{N}_\epsilon(0) \cap \mathcal{K}, \tilde{b} \neq 0$ such that $z_C(\tilde{b}) > z(\tilde{b})$. Consider an arbitrary $\epsilon > 0$ and an arbitrary $\tilde{b} \in \mathcal{N}_\epsilon(0) \cap \mathcal{K}, \tilde{b} \neq 0$ such that $z_C(\tilde{b}) > z(\tilde{b})$. Then if $\tilde{x}$ is a corresponding optimal solution to the MILP with right-hand side $\tilde{b}$, we must have $\hat{x_I} \neq 0$. Let E and $\hat{E}$ denote the set of column indices of sub-matrices of A_C corresponding to optimal bases of the continuous restrictions at 0 and $\hat{x_I}$, respectively (note that both must exist).

Case i. $E = \hat{E}$. We have

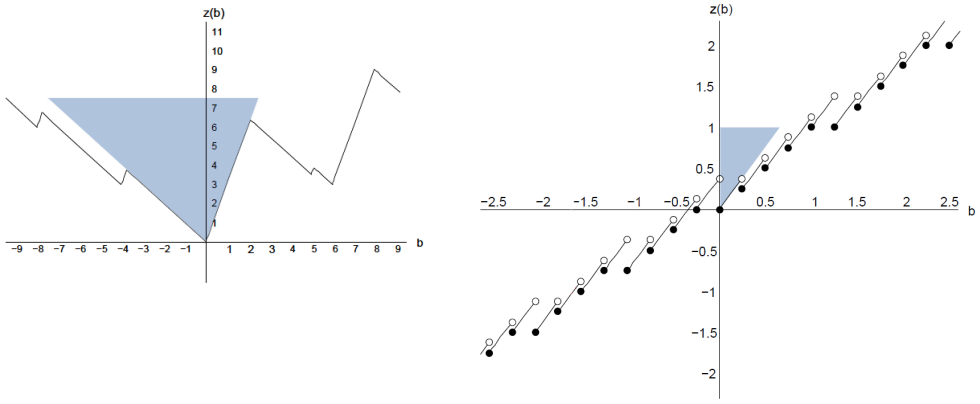
$$\begin{aligned} z_C(\hat{b}) > z(\hat{b}) &\Rightarrow c_E^\top A_E^{-1}\hat{b} > c_I^\top \hat{x_I} + c_{\hat{E}}^\top A_{\hat{E}}^{-1}\hat{b} - c_{\hat{E}}^\top A_{\hat{E}}^{-1} A_I \hat{x_I} \\ &\Rightarrow 0 > c_I^\top \hat{x_I} - c_{\hat{E}}^\top A_{\hat{E}}^{-1} A_I \hat{x_I}. \end{aligned}$$

However, the last inequality implies that at the origin, $(\hat{x_I}, A_{\hat{E}}^{-1} A_I \hat{x_I})$ provides an improved solution so that $z(0) < 0$, which is a contradiction.

Case ii. $A_E \neq A_{\hat{E}}$. We have $z_C(\hat{b}) = c_E^\top A_E^{-1}\hat{b} > c_{\hat{E}}^\top A_{\hat{E}}^{-1}\hat{b}$, which is a contradiction of the fact that z_C is the value function of the continuous restriction at 0. □

Example 2.7. Figure 2.6a shows that the epigraph of the value function of (2.15) coincides with the cone $\text{epi}(z_C) = \textit{cone}\{(2, 6), (-7, 7), (0, 1)\}$ on $\mathcal{N}_{2.125}(0)$. Similarly, Fig-

ure 2.6b demonstrates that the epigraph of the discontinuous value function (2.19) coincides with $\text{epi}(z_C) = cone\{(\frac{1}{2}, \frac{3}{4}), (\frac{1}{3}, \frac{5}{2}), (0, 1)\}$ on $\mathcal{N}_{0.25}(0) \cap \mathcal{K} = [0, 0.25) \subseteq \mathbb{R}^+$. □



(a) (b)

Figure 2.6: The MILP value function and the epigraph of the (CR) value function at the origin.

The characterization of the value function we proposed in (2.20) is finite as long as the set S_I is finite. However, there are cases where the set

$$B_I = \{b \in B : S_I(b) \neq \emptyset\}$$

is finite, while S_I remains infinite. Clearly in such cases, there is a finite representation of the value function that (2.20) does not provide. We can address this issue by representing the value function in terms of the set B_I rather than the set S_I, but even then, the representation is not minimal, as it is clear that not all members of B_I are necessary to the description. We next study the properties of the minimal subset of B_I that can fully characterize the value function of a MILP.

From the previous examples, we can observe that when the MILP has only a single constraint and the value function is thus piecewise linear, the points necessary to describing the function are the *lower break points*. To generalize the notion of lower break points to higher dimension, we need some additional machinery.

In Figure 2.6, the lower break points are also local minima of the MILP value function

and one may be tempted to conjecture that knowledge of the local minima is enough to characterize the value function. Unfortunately, it is easy to find cases for which the value function has no local minima and yet still has the nonconvex structure characteristic of a general MILP value function. Consider the following example.

Example 2.8. Consider

$$\begin{aligned} z(b) = \inf \ & -2x_1 + 6x_2 - 7x_3 \\ \text{s.t. } & x_1 - 2x_2 + 7x_3 = b \\ & x_1 \in \mathbb{Z}_+, x_2, x_3 \in \mathbb{R}_+. \end{aligned} \tag{2.22}$$

As illustrated in Figure 2.7, the extreme point of the epigraph of the continuous restriction of the problem does not correspond to a local minimum. In fact the value function does not have any local minima. □

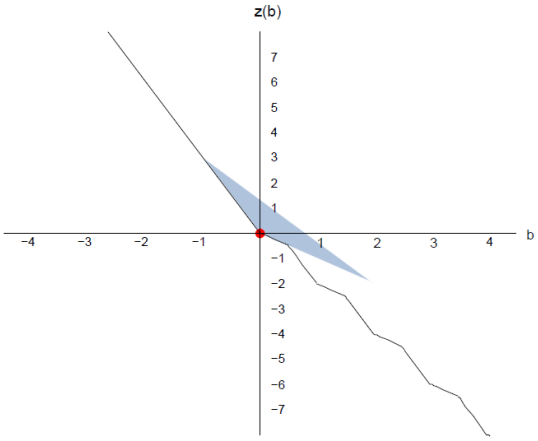


Figure 2.7: MILP value function (2.22) with no local minimum.

In the previous examples, the epigraph of z_C was also always a pointed cone. As a result, the MILP value function had lower break points that corresponded to the extreme points of $\text{epi}(\bar{z}(\cdot; x_I))$ for certain $x_I \in S_I$. However, the cone $\text{epi}(z_C)$ may not have an extreme point in general. When it fails to have one, in the single-dimensional case, the MILP value function will be linear and will have no break points. Consider the following example.

Example 2.9. Consider

$$\begin{aligned} z(b) = \inf\ & 2x_1 + 6x_2 - 7x_3 \\ \text{s.t. } & x_1 - 6x_2 + 7x_3 = b \\ & x_1 \in \mathbb{Z}_+, x_2, x_3 \in \mathbb{R}_+. \end{aligned} \tag{2.23}$$

In this example, the value function (2.23) coincides with the value function of the continuous restriction w.r.t the origin. This function is plotted in Figure 2.8. □

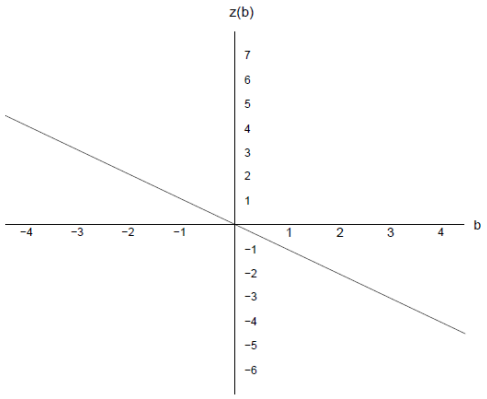


Figure 2.8: Linear and convex MILP and CR value functions to (2.23).

In this last example, the epigraph of the value function contains a line that passes through the origin. This property can be generalized to any dimension. If epi(z_C) is not a pointed cone, then for any given $\hat{x_I} \in S_I$, the boundary of the epigraph of $\bar{z}(\cdot; \hat{x_I})$ contains a line that passes through $(A_I\hat{x_I}, \bar{z}(A_I\hat{x_I}; \hat{x_I}))$. The boundary of the resulting MILP value function therefore contains parallel lines that result from translations of $\bar{z}$. Clearly, to characterize such a value function, one would need to have, for each such line, a point $\hat{b}$ such that $(\hat{b}, z(\hat{b}))$ is on the line and the value function of the continuous restriction, z_C. The case for which epi(z_C) is not a pointed cone is an edge case and its consideration would complicate the presentation substantially. For the remainder of this section, we therefore assume the more common case in which epi(z_C) is a pointed cone.

To generalize the set of lower break points to higher dimensions, we introduce the notion of *points of strict local convexity* of the MILP value function. We denote the set

of these points by B_{SLC}.

Definition 2.4. A point $\hat{b} \in B_I$ is a *point of strict local convexity* of the function $f : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}$ if for some $\epsilon > 0$ and $g \in \mathbb{R}^m$ such that $|g| > 0$, we have

$$f(b) > f(\hat{b}) + g^\top (b - \hat{b}) \text{ for all } b \in \mathcal{N}_\epsilon(\hat{b}),\ b \neq \hat{b}.$$

This definition requires the existence of a hyperplane that is tangent to the function f at the point $\hat{b} \in B_I$, while lying strictly below f in some neighborhood of $\hat{b}$. For the continuous restriction with respect to $\hat{x_I} \in S_I$, this can happen only at the extreme point of the epigraph of the function, if such a point exists. Note that at such a point, we must have $\bar{z}(\hat{b}; \hat{x_I}) = c_I^\top \hat{x_I}$. Furthermore, if $\hat{x_I} \in \arg\inf_{x_I \in S_I} \bar{z}(\hat{b}; x_I)$, then we will also have $\bar{z}(\hat{b}; \hat{x_I}) = z_I(\hat{b})$.

Proposition 2.11. *For a given $\hat{x_I} \in S_I$, $\hat{b} \in A_I \hat{x_I} + \mathcal{K}$ is a point of strict local convexity of $\bar{z}(\cdot; \hat{x_I})$ if and only if $(\hat{b}, \bar{z}(\hat{b}; \hat{x_I}))$ is the extreme point of* $\mathrm{epi}(\bar{z}(\cdot; \hat{x_I}))$.

Proof. Let $\hat{x_I} \in S_I$ and $\hat{b} \in A_I \hat{x_I} + \mathcal{K}$ be given as in the statement of the theorem. We use the following property in the proof. Let the function H_t be defined by

$$H_t(b) = \begin{cases} c_I^\top \hat{x_I} + (b - A_I \hat{x_I})^\top \eta^t & \text{for } b \in \mathcal{K}, \\ +\infty & \text{otherwise,} \end{cases}$$

where $\eta^t \in \{\nu^i\}_{i \in K} \cup \{d^j\}_{j \in L}$. Then, we have

$$\bar{z}(b; \hat{x_I}) = \sup_{t \in K \cup L} H_t(b)$$

Moreover,

$$\partial \bar{z}(\hat{b}; \hat{x_I}) = \mathrm{conv}(\{\nabla H_1, \ldots, \nabla H_p\}_{p \in P}) = \mathrm{conv}(\{\eta^1, \ldots, \eta^p\}_{p \in P}) \neq \emptyset,$$

where $P \subseteq K \cup L$ and $|P| > 1$ and finally, we have that

$$\bar{z}(\hat{b}; \hat{x_I}) = H_1(\hat{b}) = \cdots = H_p(\hat{b}) \text{ for } p \in P. \tag{2.24}$$

($\Rightarrow$) Let ϵ and g be the radius of the ball and a corresponding subgradient showing the strict local convexity of $\bar{z}(\cdot; \hat{x_I})$ at $\hat{b}$. If $\bar{z}(\cdot; \hat{x_I})$ is differentiable at $\hat{b}$, then $\exists \nu \in \mathbb{R}^m$ such that $\partial \bar{z}(\cdot; \hat{x_I}) = \{\nu\}$, and therefore $g = \nu$. Then we trivially have that $\hat{b}$ cannot be a point of strict local convexity of $\bar{z}(\cdot; \hat{x_I})$, as there always exists ϵ' with $0 < \epsilon' < \epsilon$ such that on $\mathcal{N}_{\epsilon'}(\hat{b})$, we have $\bar{z}(b; \hat{x_I}) = \bar{z}(\hat{b}; \hat{x_I}) + \nu^\top (b - \hat{b})$. Therefore, $\bar{z}(\cdot; \hat{x_I})$ cannot be differentiable at $\hat{b}$.

Since $\bar{z}(\cdot; \hat{x_I})$ is not differentiable at $\hat{b}$, there are $H_1, \ldots, H_p, p \in P$, as defined above. In the case that $p > m$, from the discussion on the LP value function, $\hat{b}$ has to be the extreme point of $\text{epi}(\bar{z}(\cdot; \hat{x_I}))$. Next, we show that $\hat{b}$ cannot be the extreme point of $\text{epi}(\bar{z}(\cdot; \hat{x_I}))$ if $p \leq m$.

When $1 < p \leq m$, equation (2.24) must still hold. Let

$$R = \{(b, \bar{z}(b; \hat{x_I})) \in (A_I \hat{x_I} + \mathcal{K}) \times \mathbb{R} : \bar{z}(b; \hat{x_I}) = H_1(b) = \cdots = H_p(b) \text{ for } p \in P\}$$

Then there exists $\tilde{b} \in N_\epsilon(\hat{b})$ such that $(\tilde{b}, z(\tilde{b})) \in R$ and $\tilde{b} \neq \hat{b}$. We have

$$\bar{z}(\tilde{b}; \hat{x_I}) - \bar{z}(\hat{b}; \hat{x_I}) = (\tilde{b} - \hat{b})^\top \eta^t, t \in P.$$

Then we can conclude that for $g \in \partial \bar{z}(\hat{b}; \hat{x_I}) = \text{conv}(\{\eta^1, \ldots, \eta^p\})$, the function $\bar{z}(\hat{b}; \hat{x_I}) + g^\top (\tilde{b} - \hat{b})$ also coincides with $\bar{z}(\tilde{b}; \hat{x_I})$ as follows. Choose $0 \leq \lambda^t \leq 1, t \in P$ such that $g = \sum_{t \in P} \lambda^t \eta^t, \sum_{t \in P} \lambda^t = 1$. From the equations

$$\lambda^t (\bar{z}(\tilde{b}; \hat{x_I}) - \bar{z}(\hat{b}; \hat{x_I})) = \lambda^t (\tilde{b} - \hat{b})^\top \eta^t, t \in P$$

we have a contradiction to $\hat{b}$ being the point of strict local convexity of $\bar{z}(\cdot; \hat{x_I})$, since

$$\bar{z}(\tilde{b}; \hat{x_I}) - \bar{z}(\hat{b}; \hat{x_I}) = \sum_{t=1}^{p} \lambda^t (\tilde{b} - \hat{b})^\top \eta^t = g^\top (\tilde{b} - \hat{b}).$$

($\Leftarrow$) Since $(\hat{b}, z(\hat{b}))$ is the extreme point of $\text{epi}(\bar{z}(\cdot; \hat{x_I}))$, then $\partial \bar{z}(\hat{b}; \hat{x_I}) = \text{conv}(\{\eta^1, \dots, \eta^p\})$, where $p \in P$ and we must have that $|P| > m$. Choose $g \in \text{int}(\text{conv}(\{\eta^1, \dots, \eta^p\}))$. For an arbitrary $\tilde{b} \in A_I \hat{x_I} + \mathcal{K}$, $\tilde{b} \neq \hat{b}$ there exists $\tilde{\eta} \in \{\eta^1, \dots, \eta^p\}$ such that $\bar{z}(\tilde{b}; \hat{x_I}) = (\tilde{b} - A_I \hat{x_I})^\top \tilde{\eta}$. Then, from the monotonicity of the subgradient of a convex function we have $(\tilde{\eta}^\top - g^\top)(\tilde{b} - \hat{b}) > 0$. Therefore,

$$\bar{z}(\tilde{b}; \hat{x_I}) = \bar{z}(\hat{b}; \hat{x_I}) + \tilde{\eta}^\top (\tilde{b} - \hat{b}) > \bar{z}(\hat{b}; \hat{x_I}) + g^\top (b - \hat{b}) \quad \forall \tilde{b} \in A_I \hat{x_I} + \mathcal{K},\ \tilde{b} \neq \hat{b}. \tag{2.25}$$

That is, $\hat{b}$ is a point of strict local convexity of $\bar{z}(\cdot; \hat{x_I})$. □

Example 2.10. Consider the MILP in Example 2.8. The blue shaded region in Figure 2.9 is $\text{epi}(\bar{z}(\cdot; 1))$. The point $(1, -2)$ is the extreme point of the cone $\text{epi}(\bar{z}(b; 1))$ and $\hat{b} = 1$ is a point of strict local convexity of the value function. □

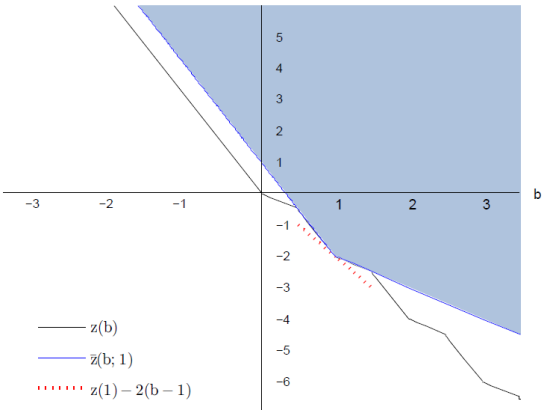


Figure 2.9: The value function of the MILP in (2.22)

Next, we discuss the points of strict local convexity of the MILP value function.

Proposition 2.12. *If $\hat{b}$ is a point of strict local convexity, then there exists $\hat{x_I} \in S_I$ such that*

– $\hat{b} = A_I \hat{x_I}$;

– $(\hat{b}, \bar{z}(\hat{b}; \hat{x_I}))$ *is the extreme point of* $\text{epi}(\bar{z}(\cdot; \hat{x_I}))$; *and*

– $\bar{z}(\hat{b}; \hat{x_I}) = c_I^\top \hat{x_I} = z_I(\hat{b}) = z(\hat{b})$.

Proof. Let $\hat{b}$ be a point of strict local convexity. If there exists $\hat{x_I} \in S_I(\hat{b})$ such that $\hat{x_I} \in \arg\inf_{x_I \in S_I} \bar{z}(\hat{b}; x_I)$, then we have that $c_I^\top x_I = z(\hat{b})$. The remainder of the statement is trivial in this case. Consider the case where such x_I does not exist. That is, for any $(x_I, x_C) \in S(\hat{b})$ such that $c_I^\top x_I + c_C^\top x_C = z(\hat{b})$, we have $y > 0$. Let one such point be $(\hat{x_I}, \hat{x_C})$. Consider $\epsilon > 0$ used to show $\hat{b}$ is a point of strict local convexity. If $\bar{z}(\cdot; \hat{x_I})$ coincides with z on $\mathcal{N}_\epsilon(\hat{b})$, then from from Proposition 2.11 it follows that $\hat{b}$ cannot be a point of strict local convexity. On the other hand, if z is constructed by multiple translations of $\bar{z}$ over $\mathcal{N}_\epsilon(\hat{b})$, since it attains the minimum of these functions, there cannot be a supporting hyperplane to z at $\hat{b}$, therefore $\hat{b}$ cannot be a point of strict local convexity. □

We note that the reverse direction of Proposition 2.12 does not hold. In particular, it is possible that for some $\hat{x_I} \in S_I$ we have $z(A_I \hat{x_I}) = c_I^\top \hat{x_I}$, but that $A_I \hat{x_I}$ is not a point of strict local convexity. For instance, in example 2.2, for $\hat{x_I} = (1, 0, 1)$ we have that $A_I \hat{x_I} = 2$ and that $\bar{z}(2; \hat{x_I}) = z(2) = 6$. Nevertheless, 2 is not a point of strict local convexity.

Points of strict local convexity may lie on the boundary of B_I. The next example illustrate a case where this happens.

Example 2.11. Consider the MILP value function

$$\begin{aligned} z(b) = \inf \ & -x_1 + 3x_2 \\ \text{s.t. } & x_1 - 3x_2 = b \\ & x_1 \in \mathbb{Z}_+,\ x_2 \in \mathbb{R}_+. \end{aligned} \tag{2.26}$$

shown in Figure 2.10. If we artificially impose the additional restriction that $b \in [0, 2]$ for

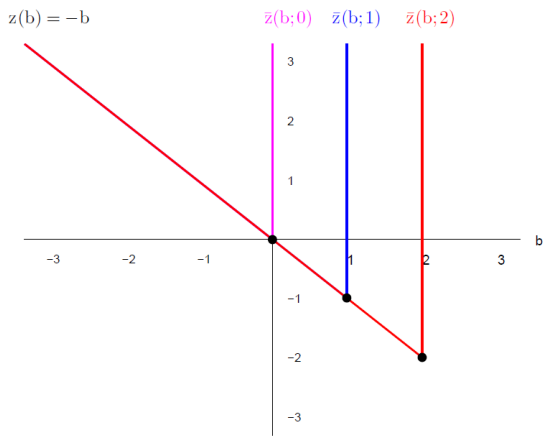


Figure 2.10: MILP Value Function of (2.26) with $B_I = [0, 2]$.

the purposes of illustration, it is clear that there is no point of strict local convexity in the interior of B_I, although epi(z_C) is a pointed cone. □

Let us further examine the phenomena illustrated by the previous example. For a given $\hat{x_I} \in S_I$, let $\hat{b} = A_I\hat{x_I}$. We know that the single extreme point of epi($\bar{z}(\cdot; \hat{x_I})$) is $(\hat{b}, \bar{z}(\hat{b}; \hat{x_I}))$ and that there must therefore be $m+1$ (2 in this example) facets of epi($\bar{z}(\cdot; \hat{x_I})$) whose intersection is this single extreme point. Now, if $\hat{b}$ is not a point of strict local convexity, then on any $\mathcal{N}_\epsilon(\hat{b})$ with $\epsilon > 0$, at most m facets of epi($\bar{z}(\cdot; \hat{x_I})$) coincide with the facets of the epigraph of the value function. This means that there exists a direction in which z is affine in the neighborhood of $(\hat{b}, \bar{z}(\hat{b}; \hat{x_I}))$; that is, $\hat{b}$ cannot be a point of strict local convexity of z. Given that the set B_I is assumed to be bounded, along this direction, the value function contains a point $(\tilde{b}, z(\tilde{b}))$ such that $\tilde{b} \in \text{bd}(conv(B_I)) \cap B_I$. Let $\bar{bd}(B_I) = \text{bd}(conv(B_I)) \cap B_I$. Since epi($z_C$) is pointed, then $\tilde{b}$ has to be a point of strict local convexity of z. This latter point is the one needed to describe the value function—the epigraph of the associated continuous restriction associated with $\tilde{b}$ contains that w.r.t. $\hat{x_I}$, which means that $A\hat{x_I}$ is not contained in the minimal set of points at which we need to know the value function.

We are now almost ready to formally state our main result. So far, we have discussed certain properties of the points of strict local convexity and showed that such points can belong to the interior or boundary of B_I. Our goal is to show that the set B_{SLC} is precisely

the minimal subset of B_I needed to characterize the full value function. Therefore, let us now formally define B_{min} to be a minimal subset of B_I such that

$$z(b) = \inf_{\hat{b} \in B_{min}} z(\hat{b}) + z_C(b - \hat{b}) \quad \forall b \in B. \tag{2.27}$$

Then we have the following result.

Proposition 2.13. $B_{min} = B_{SLC}$.

Proof. First, we show that if $\hat{b} \in B \backslash B_{SLC}$, then it is not in the set B_{min}. If $\hat{b} \in B \backslash B_I$, then from (2.20) it follows that $\hat{b}$ is not necessary to describe the value function, then $\hat{b} \notin B_{min}$. Consider $\hat{b} \in B_I \backslash B_{SLC}$. Let $\hat{x_I} \in S_I(\hat{b})$ such that $c_I^\top \hat{x_I} = z(\hat{b})$. Since $\text{epi}(z_C)$ is assumed to be pointed, we have $z(\hat{b}) = \min\{\bar{z}(\hat{b}; x_1), \bar{z}(\hat{b}; x_2),$
$\ldots, \bar{z}(\hat{b}; x_k)\}$, where $k > 1$ and $x_1, \ldots, x_k \in S_I$. Then, for some $l = 1, \ldots, k$ and $x_l \neq \hat{x_I}$ we have $\min\{\bar{z}(\hat{b}; \hat{x_I}), \bar{z}(\hat{b}; x_l)\} = \bar{z}(\hat{b}; x_l)$ and it follows that $\hat{b} \notin B_{min}$. Therefore, if $\hat{b} \in B_{min}$, then $\hat{b} \in B_{SLC}$.

We next show that if $\hat{b} \in B_{SLC}$, then $\hat{b} \in B_{min}$. Let us denote by $S'(\hat{b})$ the set of points $x_I \in S_I$ such that $(A_I x_I, c_I^\top x_I)$ coincides with the value function at $\hat{b}$. If $\hat{b} \notin B_{min}$, then all the points in $S'(\hat{b})$ can be eliminated from the description of the value function in (2.20). That is, we have

$$z(\hat{b}) = \inf_{x_I \in S_I \backslash S'(\hat{b})} \bar{z}(\hat{b}; x_I).$$

Therefore, for any pair $(x, y) \in S(\hat{b})$ that is an optimal solution to the MILP with right-hand side fixed at $\hat{b}$, we have $y > 0$. This, however, contradicts with Proposition 2.12 and we have that $\hat{b}$ cannot be a point of strict local convexity of z. □

Because it will be convenient to think of the value function as being described by a subset of S_I, we now express our main result in those terms. From Proposition 2.13, it follows that there is a subset of S_{min} of S_I that can be used to represent the value function, as shown in the following theorem. Note, however, that while B_{min} is unique, S_{min} is not.

Theorem 2.7. (Discrete Representation) *Let S_{min} be any minimal subset of S_I such that for any $b \in B_{min}$, $\exists x \in S_{min}$ such that $A_I x = b$ and $c_I^\top x = z(b)$. Then for $b \in B$, we have*

$$z(b) = \inf_{x_I \in S_I} \bar{z}(b; x_I) = \inf_{x_I \in S_{min}} \bar{z}(b; x_I). \tag{2.28}$$

Proof. The proof follows from Proposition 2.13, noting that a point $\hat{x_I} \in S_I$ such that $c_I^\top x_I > z(A_I x_I)$ cannot be necessary to describe the value function. □

Example 2.12. We apply the theorem to (2.15). In this example, over $b \in [-9, 9]$, we have that $B_{min} = \{-8, -4, 0, 5, 6, 10\}$ and $S_{min} = \{[0; 0; 2], [0; 0; 1], [0; 0; 0], [0; 1; 0], [1; 0; 0], [0; 2; 0]\}$. Clearly, the knowledge of the latter set is enough to represent the value function. □

Theorem 2.7 provides a minimal subset of S_I required to describe the value function. We will discuss in Section 2.5 that constructing a minimal such subset exactly may be difficult. Alternatively, we propose an algorithm to approximate S_{min}. This has proven empirically to be a very close approximation. Before further addressing the practical matter of how to generate the representation, we discuss a few more theoretical properties of the value function that arise from our result so far in the next two sections. The reader interested in the computational aspects of constructing the value function can safely skip to Section 2.5 for the proposed algorithm, as that algorithm does not depend on the results in the following two sections.

2.3 Stability Regions

In this section, we demonstrate that certain structural properties of the value function, such as regions of convexity and points of non-differentiability and discontinuity, can also be characterized in the context of our representation. We show that there is a one-to-one correspondence between regions over which the value function is convex and continuous—the so-called *local stability sets*— and the set B_{min}. We also provide results on the relationships between this set and the sets of non-differentiability and discontinuity

of the value function.

We start this section by introducing some notations for the sets of right-hand sides with particular properties.

Definition 2.5.

- $B_{LS}(\hat{b}) = \{b \in B : z(b) = z(\hat{b}) + z_C(b - \hat{b})\}$ is the *local stability set* w.r.t $\hat{b} \in B$;
- $B_{ES}(\hat{b}) = \text{bd}(B_{LS}(\hat{b}))$ is the *local boundary set* w.r.t $\hat{b} \in B$;
- $B_{ES} = \cup_{b \in B_{min}} B_{ES}(b)$ is the *boundary set*;
- $B_{ND} = \{b \in B : z \text{ is not differentiable at } b\}$ is the *non-differentiability set*; and
- $B_{DC} = \{b \in B : z \text{ is discontinuous at } b\}$ is the *discontinuity set*.

Example 2.13. To illustrate the above definitions, consider the value function in Example 2.2. Let $\hat{b} = 3$. Over the interval $[-9, 9]$ we have that the function $z(3) + z_C(b - 3)$ coincides with z at $b \in B_{LS}(\hat{b}) = [2.125, 3]$. Then, $B_{ES}(\hat{b}) = \{2.125, 3\}$. The minimal set is $B_{min} = \{-8, -4, 0, 5, 6\}$. The boundary set consists of the union of the local boundary sets w.r.t. minimal points; i.e.,

$$\begin{aligned} B_{ES} =& \{\{-9, -7.75\} \cup \{-7.75, -3.75\} \cup \{-3.75, 2.125\} \cup \{2, 125, 5.125\} \cup \{5.125, 8\}\} \\ =& \{-9, -7.75, -3.75, 2.125, 5.125, 8\}. \end{aligned}$$

The non-differentiablity set is

$$B_{ND} = \{-9, -8, -7.75, -4, -3.75, 0, 2.125, 5, 5.125, 6, 8, 9\}.$$

Finally, $B_{DC} = \emptyset$. □

The main result of this section is Theorem 2.8. The goal is to show that the value function is convex and continuous over the local stability sets associated with the members of B_{min}. Furthermore, in this theorem we demonstrate the relationship between

the set B_{min}, the boundary set, B_{ES}, and the sets of point of non-differentiability and discontinuity of the value function. We next state the theorem.

Theorem 2.8.

i. Let $\hat{b} \in B$.

- *There exists $x_I^* \in S_{min}$ such that for any $\tilde{b} \in \text{int}(B_{LS}(\hat{b}))$, there exists $x_C \in \mathbb{R}_+^{n-r}$ such that (x_I^*, x_C) is an optimal solution to the MILP with right-hand side $\tilde{b}$.*
- *z is continuous and convex over $\text{int}(B_{LS}(\hat{b}))$.*

ii. $\hat{b} \in B_{ES}$ if and only if for any $\epsilon > 0$, $\nexists x_I^* \in S_I$ such that $z(b) = c_I^\top x_I^* + z_C(b - A_I x_I^*)$ for all $b \in \mathcal{N}_\epsilon(\hat{b})$.

iii. Let $\hat{b} \in B_{min}$. Then, $\text{int}(B_{LS}(\hat{b}))$ is the maximal set of right-hand sides containing $\hat{b}$ over which the value function is convex and continuous.

iv. For the general MILP value function, we have $B_{min} \subseteq B_{ND}$ and $B_{ES} \subseteq B_{ND}$. Furthermore, if the MILP value function is discontinuous, we have $B_{min} \subseteq B_{DC} \subseteq B_{ES} \subseteq B_{ND}$.

Proof. We build to the proof of the theorem, which constitute the remainder of this section, by proving lemmas 2.1–2.10, The first and second parts of the theorem follow from lemma 2.1 and lemma 2.4. The third part of the theorem is shown in lemma 2.5. The last part follows from lemmas 2.7–2.10. □

In the first lemma, we show properties of the function on differentiable regions within local stability sets.

Lemma 2.3. *Let $\hat{b} \in B$. Then there exists $x_I^* \in S_I$ such that for any $\tilde{b} \in \text{int}(B_{LS}(\hat{b}))$, there exists $x_C \in \mathbb{R}_+^{n-r}$ such that (x_I^*, x_C) is an optimal solution to the MILP with right-hand side $\tilde{b}$. Furthermore, z is continuous and convex over $\text{int}(B_{LS}(\hat{b}))$*

Proof. From Theorem 2.7, for any $\hat{b} \in B$ there exists $x^* \in S_{min}$ such that $\text{int}(B_{LS}(\hat{b})) = \text{int}(\{b \in B : z(b) = c_I^\top x_I^* + z_C(b - A_I x_I^*)\})$. Therefore, for any $\tilde{b} \in \text{int}(B_{LS}(\hat{b}))$, $z(\tilde{b}) = c_I^\top x_I^* + c_C^\top x_C^*$ where $x_C^* = \text{argmin}\{c_C^\top x_C : A_C x_C = \tilde{b} - A_I x_I^*, x_C \in \mathbb{R}_+^{n-r}\}$. The convexity and continuity of z on $\text{int}(B_{LS}(\hat{b}))$ follows trivially. □

Theorem 2.1. *If z is differentiable over $\mathcal{N} \subseteq B$, then there exist $x_I^* \in S_I$ and $E \in \mathcal{E}$ such that $z(b) = c_I^\top x_I^* + \nu_E^\top(b - A_I x_I^*)$ for all $b \in \mathcal{N}$.*

Proof. Let an arbitrary $\hat{b} \in \mathcal{N}$ be given. By Theorem 2.7, we know that there exists $\hat{x_I}^* \in S_{min}$ such that $z(\hat{b}) = \bar{z}(\hat{b}; \hat{x_I})$ and $A_I x_I^* \in B_{min}$. Then, we have $z(\hat{b}) = c_I^\top x_I^* + \nu_E^\top(\hat{b} - A_I x_I^*)$ with $E \in \mathcal{E}$ and there exists (x_I^*, x_E, x_N), an optimal solution to the given MILP with right-hand side $\hat{b}$, where x_E and x_N correspond to the basic and non-basic variables in the corresponding solution to the continuous restriction w.r.t. x_I^*. It follows that the vector $(x_I^*, x_E + A_E^{-1}(b - \hat{b}), x_N)$ is a feasible solution for any $b \in \mathcal{N}$.

Now, let another arbitrary point $\tilde{b} \in \mathcal{N}$ be given. We show that $(x_I^*, x_E + A_E^{-1}(\tilde{b} - \hat{b}), x_N)$ must be an optimal solution for right-hand side $\tilde{b}$. Since $\hat{b} \in \mathcal{N}, \tilde{b} \in \mathcal{N}$ and z is differentiable over $\mathcal{N}$, then ν_E is the unique optimal dual solution to the continuous restriction by Proposition 2.7 and we have

$$\begin{aligned} z(\tilde{b}) &= c_I^\top x_I^* + c_E^\top(x_E + A_E^{-1}(\tilde{b} - \hat{b})) + c_N^\top x_N \\ &= z(\hat{b}) + \nu_E^\top(\tilde{b} - \hat{b}) = c_I^\top x_I^* + \nu_E^\top(\hat{b} - A_I x_I^*) + \nu_E^\top(\tilde{b} - \hat{b}) \\ &= c_I^\top x_I^* + \nu_E^\top(\tilde{b} - A_I x_I^*) = \bar{z}(\tilde{b}; x_I^*). \end{aligned}$$

Since $\tilde{b}$ and $\hat{b}$ were arbitrary points in $\mathcal{N}$, the result holds for all such pairs and this ends the proof. □

It follows from the previous result that if the value function is differentiable over $\mathcal{N} \subseteq B$, then its gradient at every right-hand side in $\mathcal{N}$ is a unique optimal dual solution to the continuous restriction problem w.r.t. some $x_I^* \in S_I$. This generalizes Proposition 2.7 on the gradient of the function at a differentiable point of the LP value function to the mixed integer case. As an example, in (2.19) the gradient of z at any differentiable point

is $\nu = \frac{3}{2}$.

Next, we show the second part of Theorem 2.8 in the following result.

Lemma 2.4. *$\hat{b} \in B_{ES}$ if and only if for any $\epsilon > 0$, $\nexists x_I^* \in S_I$ such that $z(b) = c_I^\top x_I^* + z_C(b - A_I x_I^*)$ for all $b \in \mathcal{N}_\epsilon(\hat{b})$.*

Proof. ($\Rightarrow$) Let $\epsilon > 0$ be given and assume $\exists x_I^* \in S_I$ such that $z(b) = c_I^\top x_I^* + z_C(b - A_I x_I^*)$ for all $b \in \mathcal{N}_\epsilon(\hat{b})$. Now, let $\tilde{b} \in B_{min}$ be such that $\hat{b} \in B_{ES}(\tilde{b})$. Then for all $b \in \mathcal{N}_\epsilon(\hat{b})$, we have $z(b) = \bar{z}(b; x_I^*) = z(\tilde{b}) + z_C(b - \tilde{b})$. That is, $\hat{b} \in \text{int}(B_{LS}(\tilde{b}))$.

($\Leftarrow$) Let $\tilde{b} \in B_{min}$ be such that $b \in \text{int}(B_{LS}(\tilde{b}))$. Then from Lemma 2.3, there exists $x_I^* \in S_I$ optimal for all $b \in B_{LS}(\tilde{b})$. □

Next, we arrive at showing the third part of Theorem 2.8.

Lemma 2.5. *Let $\hat{b} \in B_{min}$. Then, $\text{int}(B_{LS}(\hat{b}))$ is the maximal set of right-hand sides containing $\hat{b}$ over which the value function is convex and continuous.*

Proof. Assume the contrary that $B_{LS}(\hat{b})$ with $\hat{b} \in B_{min}$ is not the maximal set. Then, there exists $\tilde{b}$ in the boundary set w.r.t $\hat{b}$, $B_{ES}(\hat{b})$, and $\epsilon > 0$ such that the value function is continuous and convex at $\mathcal{N}_\epsilon(\tilde{b})$. From Theorem 2.7 and Lemma 2.4 we have

$$z(b) = \min_{x_I^i \in S_{min}} \{c_I^\top x_I^i + (b - A_I x_I^i)^\top \nu^i\}, b \in \mathcal{N}_\epsilon(\tilde{b}), \tag{2.29}$$

where ν^i is the optimal dual solution to $z_C(\tilde{b} - A_I x_I^i)$ and the set $x_I^i \in S_{min}$ contains two or more distinct members. Then, z is concave over $\mathcal{N}_\epsilon(\tilde{b})$ unless all the polyhedral functions in (2.29) are the same. But then $\mathcal{N}_\epsilon(\tilde{b})$ is a subset of $B_{LS}(\hat{b})$. □

So far, in Theorem 2.8 we have demonstrated that over the local stability set w.r.t a minimal point, the integer part of the solution to the MILP remains constant and the value function of the MILP is a translation of the continuous restriction value function. This can be viewed as a generalization of the value function of a PILP with inequality constraints, where the value function is constant over local stability sets ($z_C(b) = 0$ for

$b \in \mathbb{R}^m$). These regions are characterized by Schultz et al. (1998). In this case, the minimal points generalize the notion of *minimal tenders* discussed in Trapp et al. (2013).

Before showing the fourth and last part of Theorem 2.8, we need another lemma on the necessary conditions for the continuity of the value function.

Lemma 2.6. *If $z_C(b) < \infty$ for all $b \in B$, then z is continuous over B.*

Proof. If $z_C(\hat{b}) < \infty$ for some $\hat{b} \in B$, then the continuous restriction w.r.t. the origin and its dual are both feasible and have optimal solution values equal to $z_C(\hat{b})$. Therefore, $\bar{z}$ is finite and continuous on B. It can be proved by induction that the minimum of countably many continuous functions defined on B is continuous on B. The continuity of z follows by the representation in (2.20). □

We now proceed to show the last part of the theorem. Lemmas 2.7–2.10 address the relationships between the discontinuity set of the value function with the minimal set of right-hand sides, the boundary set, and the set of non-differentiability points. Combining the following lemmas, the proof of the theorem is complete.

Lemma 2.7. $B_{min} \subseteq B_{ND}$.

Proof. Assume the value function is differentiable at some $\hat{b} \in B_{min}$. Let $\nabla z(\hat{b}) = g$. Then, there exists some $\epsilon > 0$, such that $z(b) = z(\hat{b}) + g^\top (b - \hat{b})$ for all $b \in \mathcal{N}_\epsilon(\hat{b})$. But then, from the definition of a point of strict local convexity, $\hat{b}$ cannot be in B_{SLC} and therefore, $\hat{b} \notin B_{min}$. □

Earlier we showed that the discontinuities of the MILP value function may only happen when it no longer attains its minimum over some translated $\bar{z}$ and a switch to another translation is required. This is used next to show the relationship between the discontinuity and boundary sets.

Lemma 2.8. $B_{DC} \subseteq B_{ES}$.

Proof. Assume to the contrary that there exists $\tilde{b} \in B_{DC}$ but $\tilde{b} \in \text{int}(B_{LS}(\hat{b}))$ for some $\hat{b} \in B_{min}$. Then from Theorem 2.8, there exists $\epsilon > 0$ such that $z(b) = z(\hat{b}) + z_C(b - \hat{b})$ for all $b \in \mathcal{N}_\epsilon(\tilde{b})$. Therefore, z can only be continuous on $\mathcal{N}_\epsilon(\tilde{b})$, which is a contradiction. □

Lemma 2.9. *If the value function is discontinuous, then* $B_{min} \subseteq B_{DC}$.

Proof. Since $B_{DC} \neq \emptyset$, from Lemma 2.6 we have $\mathcal{K} \neq \mathbb{R}^m$. Then, for any $\hat{b} \in B$ we have $\hat{b} \in B_{ES}(b)$; that is, any right-hand side lies on the boundary of its local stability set. Consider $\hat{b} \in B_{min}$. If z is continuous at $\hat{b}$ then there exists $\epsilon > 0$ and $\tilde{b} \in B_{min}$ such that $\tilde{b} \neq \hat{b}$ and for any $b_1 \in \mathcal{N}_\epsilon(\hat{b}) \backslash B_{ES}(\hat{b})$ we have $z(b_1) = z(\tilde{b}) + z_C(b_1 - \tilde{b})$. Consider $b_2 \in \mathcal{N}_\epsilon(\hat{b}) \cap B_{ES}(\hat{b})$. If $z(\tilde{b}) + z_C(b_2 - \tilde{b}) < z(b_2)$, then z cannot be the value function at b_2. On the other hand, if $z(\tilde{b}) + z_C(b_2 - \tilde{b})$ lies above or on $z(b_2)$, it can be easily shown that there cannot exist a supporting hyperplane of z at $\hat{b}$ that lies strictly below z on an arbitrarily small neighborhood of $\hat{b}$. Then $\hat{b}$ cannot be in B_{min}. □

The next result shows that if $\hat{b}$ belongs to the boundary set w.r.t a minimal point, then z is non-differentiable at $\hat{b}$.

Lemma 2.10. $B_{ES} \subseteq B_{ND}$.

Proof. Assume there exist some $\tilde{b} \in B_{ES}(\hat{b}), \hat{b} \in B_{min}$ such that z is differentiable at $\tilde{b}$. Then there exists $\epsilon > 0$ and $E \in \mathcal{E}$ such that for all $b \in N_\epsilon(\tilde{b})$ we have

$$\begin{aligned} z(b) &= z(\tilde{b}) + \nu_E^\top (b - \tilde{b}) \\ &= z(\hat{b}) + \nu_E^\top (\tilde{b} - \hat{b}) + \nu_E^\top (b - \tilde{b}) \\ &= z(\hat{b}) + \nu_E^\top (b - \hat{b}) = z(\hat{b}) + z_C(b - \hat{b}). \end{aligned} \tag{2.30}$$

But this contradicts the third part of Theorem 2.8. □

We finish this section by applying Theorem 2.8 to the continuous value function in Example 2.2 and the discontinuous value function in Example 2.5.

Example 2.14. Consider the value function (2.15). Figure 2.11 shows the optimal integer parts $x_I^1, \ldots, x_I^4$ of solutions to the corresponding MILP over the local stability sets $B_{LS}(-4), B_{LS}(0), B_{LS}(5)$ and $B_{LS}(6)$, respectively. One can observe that both the minimal set and the boundary set of the value function are subsets of its set of non-differentiability points.

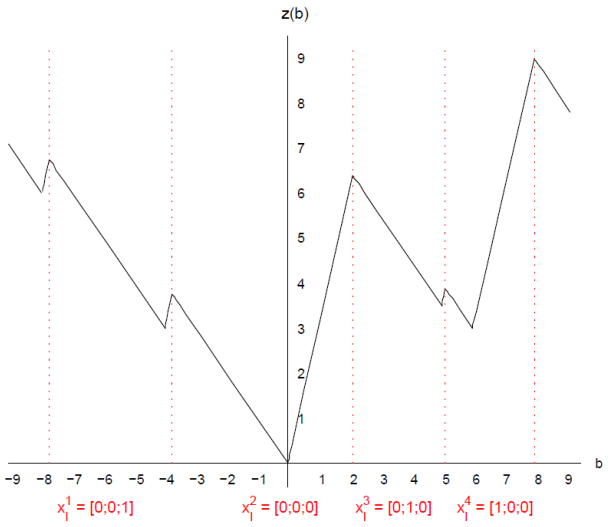


Figure 2.11: Local stability sets and corresponding integer part of solution in (2.15).

Similarly, in Example 2.5, $x_I^1 = [1\ 2]^\top, x_I^2 = [2\ 3]^\top, x_I^3 = [3\ 4]^\top, x_I^4 = [0\ 0]^\top, x_I^5 = [1\ 1]^\top, x_I^6 = [2\ 2]^\top, x_I^7 = [3\ 3]^\top$ are respectively the integer parts of the solutions for right-hand sides in the local stability sets $B_{LS}(-0.75), B_{LS}(-0.5), \ldots, B_{LS}(0.5), B_{LS}(0.75)$. In this case, the value function is discontinuous on the points that belong to the minimal set and we have $B_{min} \cup B_{ES} = B_{ES} = B_{DC} = B_{ND}$. □

Remark 2.1. If z is continuous over B, then $S_D \neq \emptyset$. This follows from the fact that if $S_D = \emptyset$, then $z(0) = z_C(0) = -\infty$ which contradicts $z(0) = 0$. Therefore, we have that $z_C(b) > -\infty$ for all $b \in \mathbb{R}^m$. However, we may still have $z_C(b) = \infty$ for some $b \in \mathbb{R}^m$. The following is an example.

Example 2.15. The value function defined by (Ex.12) below is continuous on $\mathbb{R}$, although $\mathcal{K} = \mathbb{R}_+$.

$$\begin{aligned} z(b) = \ & \inf x_1 - x_2 \\ \text{s.t.} \ & -x_1 + x_2 = b \\ & x_1 \in \mathbb{Z}_+,\ x_2 \in \mathbb{R}_+. \end{aligned} \tag{Ex.12}$$

□

2.4 Simplified Jeroslow Formula

Blair (1995) identified a closed form representation of the MILP value function called the

Jeroslow Formula. In this formula, the value function is obtained by taking the minimum of $|\mathcal{E}|$ functions, each consisting of a PILP value function and a linear term. In this section, we study the connection between our representation of the MILP value function and the representation in the Jeroslow Formula and provide a simpler representation of it.

Let us denote by $\lfloor \cdot \rfloor$ the component-wise floor function. For $E \in \mathcal{E}$, we define

$$\lfloor b \rfloor_E = A_E \left\lfloor A_E^{-1} b \right\rfloor \ \forall b \in B,\ T_E = \{b \in B : A_E^{-1} b \in \mathbb{Z}^m\},\ \text{and}\ T = \bigcap_{E \in \mathcal{E}} T_E.$$

For a given $\hat{b} \in \mathcal{K}$, let $E \in \mathcal{E}$ such that $\hat{x_{IE}} = A_E^{-1}\hat{b}$ is the corresponding solution to the continuous restriction w.r.t. the origin. When $\hat{b} \in T_E$, see that $\hat{b} = A_E \hat{x_{IE}}$ is an integer linear combination of vectors in feasible basis A_E. Hence, the same is true for any member of T.

Now consider the continuous restriction w.r.t to a given $\hat{x_I} \in S_I$. Then we have more generally that the corresponding solution to the continuous restriction w.r.t. $\hat{x_I}$ at a given $\hat{b} \in \mathcal{K} + A_I \hat{x_I}$ is

$$\tilde{x}_E = A_E^{-1}(\hat{b} - A_I \hat{x_I}),$$

where $E \in \mathcal{E}$. In this case, when $\hat{b} \in T$, we can no longer guarantee that $\tilde{x}_E \in \mathbb{Z}^m$. By an appropriate scaling, however, we can ensure this property, and this is one of the key steps in deriving the Jeroslow formula. Since all matrices are assumed to be rational, there exists $M \in \mathbb{Z}_+$ such that $MA_E^{-1}A_j \in \mathbb{Z}^m$ for all $E \in \mathcal{E}$ and all $j \in I$, with A_j denotes the j^{th} column of A. Then, since $A_E^{-1}b$ is integral for any $b \in T$ and $E \in \mathcal{E}$, we have that the value function of the following PILP is equal to the value function of the original MILP for all $b \in T$.

Proposition 2.14. *(Blair, 1995) There exists $M \in \mathbb{Z}_+$ such that $z(b) = z_M(b)$ for all $b \in T$, where*

$$\begin{aligned} z_M(b) = \inf\ & c_I^\top x_I + \frac{1}{M} c_C^\top x_C \\ \text{s.t. } & A_I x_I + \frac{1}{M} A_C x_C = b \\ & (x_I, x_C) \in \mathbb{Z}_+^r \times \mathbb{Z}_+^{n-r}. \end{aligned} \tag{2.31}$$

Proof. Let $M \in \mathbb{Z}_+$ such that $MA_E^{-1}A_j$ is a vector of integers for all $E \in \mathcal{E}$ and $j \in I$. Scaling A_C and c_C by $\frac{1}{M}$ guarantees that $A_j \in T$ for all $j \in I$. Therefore, $MA_E^{-1}(b - A_I x_I) \in \mathbb{Z}^m$ for all $x_I \in \mathbb{Z}^r$ and $E \in \mathcal{E}$. It follows that the solution value to z and z_M is equal for any $b \in T$. □ □

We illustrate the scaling procedure in the following example.

Example 2.16. Consider Example 2.2. In (2.15) we have $A_I^j \in \{6, 5, -4\}$ for $j = 1, \ldots, 3$ and $\mathcal{E} = \{\{1\}, \{2\}\}$ with $A_{\{1\}} = 2$ and $A_{\{2\}} = -7$. We choose $M = 14$ so that $MA_E^{-1}A_I^j \in \mathbb{Z}$ for all $E \in \mathcal{E}$. The corresponding scaled PILP problem is

$$\begin{aligned} z_M(b) = \inf\ & 3x_1 + \frac{7}{2}x_2 + 3x_3 + \frac{3}{7}x_4 + \frac{1}{2}x_5 \\ \text{s.t. } & 6x_1 + 5x_2 - 4x_3 + \frac{1}{7}x_4 - \frac{1}{2}x_5 = b \\ & x_1, x_2, x_3, x_4, x_5 \in \mathbb{Z}_+. \end{aligned} \tag{2.32}$$

Figure 2.12a demonstrates the value function (2.32) for $b \in [-9, 9]$. From the figure we can see that z and z_M coincide on intervals of length $\frac{1}{7}$ where $A_{\{1\}} = 2$ is optimal, while the two functions coincide at intervals of length $\frac{1}{2}$ where $A_{\{2\}} = -7$ is optimal.

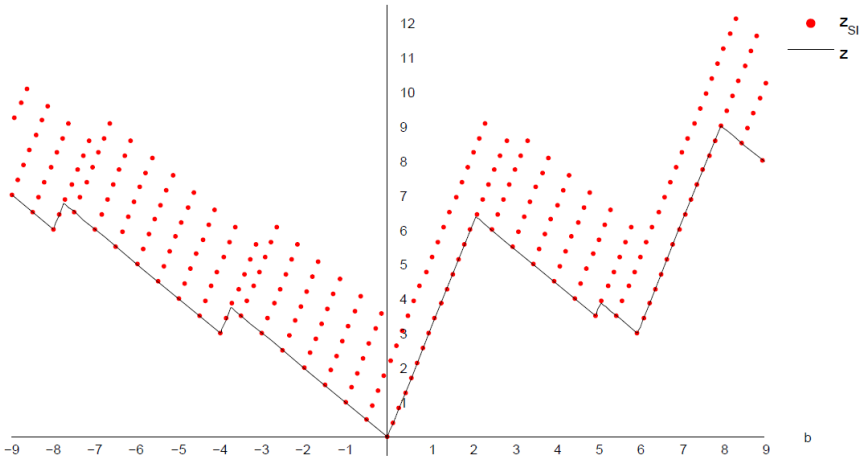
Let us have a closer look at the interval $[2, 2.6]$ illustrated in Figure 2.12b. The set of feasible right-hand sides for the scaled PILP (2.32) in this interval is $\{2, 2\frac{1}{14}, 2\frac{2}{14}, 2\frac{3}{14}, \ldots, 2\frac{8}{14}\}$. Among these points, the value function coincides with (2.32) at 2 and 2.5. We have

$$z(b) = z_C(b) = \nu_{\{1\}} b = 3b, \text{ for } b \in [2, 2.125]$$

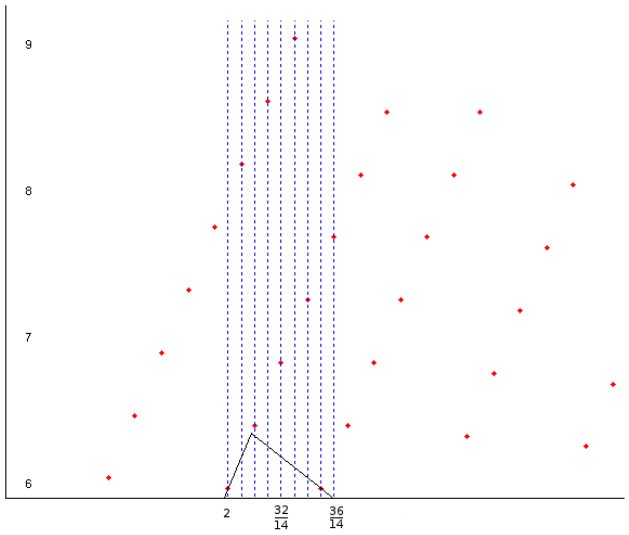
and

$$z(b) = \bar{z}(b; [0, 1, 0]^\top) = \frac{7}{2} + \nu_{\{2\}}(b - 5) = \frac{7}{2} - (b - 5), \text{ for } b \in [2.125, 2.6].$$

Since $T_{\{1\}} = \{b : MA_{\{1\}}^{-1} b = 7b \in \mathbb{Z}\}$, we have $T_{\{1\}} \cap [2, 2.6] = \{b : b = \frac{i}{7}, i = 14, \ldots, 18\}$. Similarly, $T_{\{2\}} \cap [2, 2.6] = \{b : b = \frac{i}{2}, i = 4, 5\}$.



(a)



(b)

Figure 2.12: The scaled PILP value function (2.32).

Over the intervals for which A_{E^*} is the optimal dual basis for the corresponding continuous restriction, z and z_M coincide at $T_{E^*} = \{b \in B : b = \lfloor b \rfloor_{E^*} = kMA_{E^*}^{-1}, k \in \mathbb{Z}_+\}$. For instance, $z(2) = z_M(2)$ with $2 \in T_{\{1\}}$, but $z(2\frac{1}{7}) \neq z_M(2\frac{1}{7})$ with $2\frac{1}{7} \in T_{\{1\}}$. This is due to the fact that $A_{\{1\}}$ is the optimal dual basis at $z(2) = z_C(2)$ but not at

$\bar{z}(2\frac{1}{7}; [0,1,0]^\top) = z(2\frac{1}{7})$. □

Remark 2.2. Note that the z and z_M may coincide at some right-hand side that is not in the set T, e.g., $b = 2.5 \in T_{\{2\}} \backslash T_{\{1\}}$.

Blair and Jeroslow (1984) identified a class of functions called Gomory functions and showed that for any PILP, there exists a Gomory function whose value coincides with that of the value function of the PILP wherever it is finite. To extend this result to the MILP case, Blair (1995) proposed "rounding" any $b \in B$ to some $\lfloor b \rfloor_E$ with $E \in \mathcal{E}$ and evaluating the latter using a PILP. Note that (2.31) has to be modified to be used for this purpose, since it is not necessarily feasible for all $\lfloor b \rfloor_E$, $E \in \mathcal{E}$; i.e., it is possible to have $\tilde{x}_E = M(A_E^{-1} - A_E^{-1} A_I \hat{x_I}) < 0$ for $\hat{x_I} \in \mathbb{Z}_+^r$. To achieve feasibility for all $\lfloor b \rfloor_E$, Blair (1995) proposed the following modification of (2.31) and used it in the Jeroslow formula.

$$\begin{aligned} z_{JF}(t) = \inf\ & c_I^\top x_I + \frac{1}{M} c_C^\top x_C + z(-\frac{1}{M} \sum_{j \in C} A_j) y \\ \text{s.t. } & A_I x_I + \frac{1}{M} A_C x_C + (-\frac{1}{M} \sum_{j \in C} A_j) y = t \\ & x_C \in \mathbb{Z}_+^{n-r}, x_I \in \mathbb{Z}_+^r, y \in \mathbb{Z}_+. \end{aligned} \tag{2.33}$$

Finally, he used linear terms of the form of $\nu_E^\top (b - \lfloor b \rfloor_E)$ to compensate for the "rounding" of b to $\lfloor b \rfloor_E$ with $E \in \mathcal{E}$. Together, he showed that for any MILP, there is a Gomory function G corresponding to the value function of the PILP (2.33) with

$$z(b) = \inf_{E \in \mathcal{E}} \{ G(\lfloor b \rfloor_E) + \nu_E^\top (b - \lfloor b \rfloor_E) \}. \tag{2.34}$$

The representation of the value function in (2.34) is known as the Jeroslow Formula.

Although it is a bit difficult to tease out, given the technical nature of the Jeroslow Formula, there is an underlying connection between it and our representation. In particular, the set T has a role similar to the role of B_{min} in our representation—it is a discrete subset of the domain of the value function of the original MILP over which the original value function agrees with the value function of a related PILP. This is the same property our set B_{min} has and it is what allows the value function to have a discrete

representation. Furthermore, the correction terms in the Jeroslow Formula play a role similar to the value function of the continuous restriction in our representation.

The advantage our representation has over the Jeroslow Formula is that B_{min} is potentially a much smaller set and the value M in the Jeroslow formula would be difficult to calculate a priori. Furthermore, even if M could be obtained in some cases, evaluating the value function for a given $b \in B$ using the Jeroslow formula ostensibly requires the evaluation of a Gomory function for every $\lfloor b \rfloor_E$ for all $E \in \mathcal{E}$, including those feasible basis A_E that are not optimal at b. The number of evaluations required is equal to the size of $\bigcup_{E \in \mathcal{E}} T_E$. These drawbacks relegate the Jeroslow formula to purely theoretical purposes. On the surface, there does not seem to be any way to utilize it in practice. Nevertheless, it is possible to simplify the Jeroslow Formula, replacing T by B_{min} and eliminating the need to calculate M in the process. This leads to a more practicable variant of the original formula. First, we show formally that B_{min} is a subset of T.

Proposition 2.15. *$B_{min} \subseteq T$.*

Proof. Let $(\hat{x_I}, \hat{x_C})$ be an optimal solution to (MVF) at $\hat{b} \in B_{min}$. From Proposition 2.12, we have that $\hat{x_C} = 0$ in any optimal solution of the value function at $\hat{b}$. Then for all $E \in \mathcal{E}$ we have

$$\left\lfloor \hat{b} \right\rfloor_E = \frac{1}{M} A_E \left\lfloor M A_E^{-1} A_I \hat{x_I} + M A_E^{-1} A_C \hat{x_C} \right\rfloor = A_I \hat{x_I} = \hat{b}.$$

Then, $\left\lfloor \hat{b} \right\rfloor_E = \hat{b}$ for all $E \in \mathcal{E}$ and $\hat{b} \in T$. □

Theorem 2.2. *If $\hat{b} \in B_{min}$, then $z(\hat{b}) = z_I(\hat{b}) = z_M(\hat{b}) = z_{JF}(\hat{b}) = G(\hat{b})$ where G is the PILP value function in (2.34).*

Proof. The first equality follows from Proposition 2.12. The second equality holds since $x_C = 0$ in any optimal solution to (MVF) at a right-hand side in B_{min}. $z_M(b)$ is equal to $z_{JF}(b)$ since for $b = \lfloor b \rfloor_E$ for all b when $b \in B_{min}$ and y can be fixed to zero in (2.33). The last equality holds for any $\lfloor b \rfloor_E$, $E \in \mathcal{E}$. □

Theorem 2.9. (Simplified Jeroslow formula)

$$z(b) = \inf_{\hat{b}\in B_{min}, E\in\mathcal{E}} \{z_I(\hat{b}) - \nu_E^\top(b-\hat{b})\}. \tag{2.35}$$

Proof. We have

$$\begin{aligned} z(b) &= \inf_{E\in\mathcal{E}} \{G(\lfloor b\rfloor_E) + \nu_E^\top(b - \lfloor b\rfloor_E)\} \\ &= \inf_{\hat{b}\in T_E, E\in\mathcal{E}} \{G(\hat{b}) + \nu_E^\top(b-\hat{b})\} \\ &= \inf_{\hat{b}\in B_{min}} \{z_I(\hat{b}) + \sup_{E\in\mathcal{E}} \nu_E^\top(b-\hat{b})\} \\ &= \inf_{\hat{b}\in B_{min}, E\in\mathcal{E}} \{z_I(\hat{b}) - \nu_E^\top(b-\hat{b})\}. \end{aligned}$$

The first equation is the Jeroslow Formula. The second one is because $\lfloor b\rfloor_E \in T_E$ for any $E \in \mathcal{E}$ and $b \in B$. From Theorem 2.7, $z(b) = \inf\{\bar{z}(b; x_I) : A_I x_I \in B_{min}\}$, then the third equality holds. The last equation follows trivially. □

The above result provides a variation of the Jeroslow formula where there is no need to find the value of M, or to evaluate the PILP problem z_{JF} for members of $\bigcup_{E\in\mathcal{E}} T_E$. Instead, we need to evaluate the simpler PILP problem z_I for the set $B_I \subseteq T \subseteq \bigcup_{E\in\mathcal{E}} T_E$. The difference in the size of B_I and $\bigcup_{E\in\mathcal{E}} T_E$ can be significant. We provide an illustrative example next.

Example 2.17. The value function of (2.32) for right-hand sides in $T_{\{1\}} \cup T_{\{2\}}$ is plotted in Figure 2.13 with filled blue circles. At a point $\hat{b} \in T_{\{1\}} \cup T_{\{2\}}$, we have $z_M(\hat{b}) = G(\hat{b})$ where G is the Gomory function corresponding to the PILP (2.32). The Jeroslow formula for the MILP value function over $[-9, 9]$ requires finding all such points. Alternatively, we can have a smaller representation by constructing the value function of the integer restriction of (2.15). i.e., $z_I(\hat{b}) = \inf\{3x_1+\frac{7}{2}x_2+3x_3 : 6x_1+5x_2-4x_3 = \hat{b}, x_1, x_2, x_3 \in \mathbb{Z}^+\}$. This value function is plotted in Figure 2.14. However, the alternative formulation (2.35) requires finding $G(\hat{b}) = z_I(\hat{b})$ for $\hat{b} \in B_{min} = \{-8, -4, 0, 4, 5, 10\}$. □

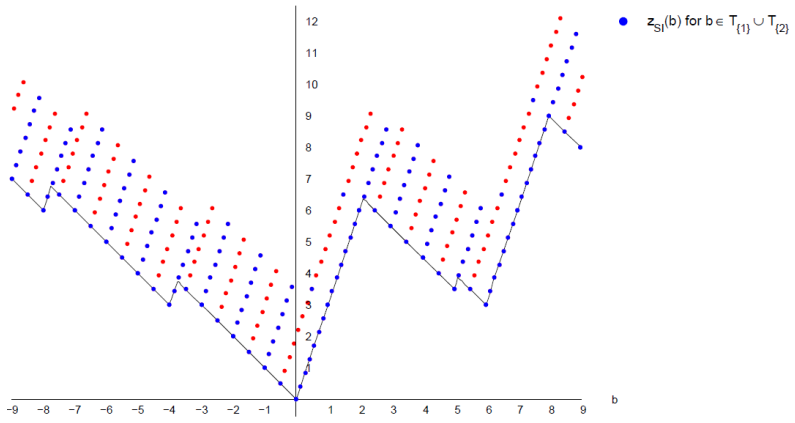


Figure 2.13: The value function of (2.32) for $b \in T_{\{1\}} \cup T_{\{2\}} \cup [-9, 9]$.

2.5 Algorithm for Construction

In this section, we discuss the use of our representation in computational practice. To sum up what we have seen so far, we have shown that there exists a discrete set S_{min} (not necessarily unique) over which the value of z can be determined by solving instances of the integer restriction. Theorem 2.7 tells us that, in principle, if we knew $z(A_I x_I)$ for all $x_I \in S_{min}$, then $z(b)$ could be computed at any $b \in B$ by solving $|S_{min}|$ LPs.

Our discrete representation of the value function in Theorem 2.7 is the same as

$$z(b) = \inf_{x_I \in S_{min}} c_I^\top x_I + z_C(b - A_I x_I). \tag{2.36}$$

If $|S_{min}|$ is relatively small, this yields a practical method. The most straightforward way to utilize our representation would then be to generate the set S_{min} a priori and to apply the above formula to evaluate $z(b)$ for $b \notin B_{min}$.

In general, however, obtaining an exact description of the set S_{min} seems to be difficult. One solution to this problem would be to instead generate the value function of the integer restriction first by the procedure of Kong et al. (2006), which is finite under our assumptions. We illustrate this hypothetical procedure in the following example.

Example 2.18. Consider constructing the value function defined by (2.15) for $b \in [-7, 7]$. The value function of the integer restriction z_I is plotted in Figure 2.14. Clearly, com-

plete knowledge of z_I is unnecessary to describe the MILP value function, as this requires evaluation for each point in S_I, whereas we have already shown that evaluation of points in S_{min} is enough. In this example, over $b \in [-7, 7]$, we have that $S_{min} = \{0; 0; 1], [0; 0; 0], [0; 1; 0], [1; 0; 0]\}$. Therefore, four evaluations is enough, yet at least 15 are required for constructing the value function of the PILP. □

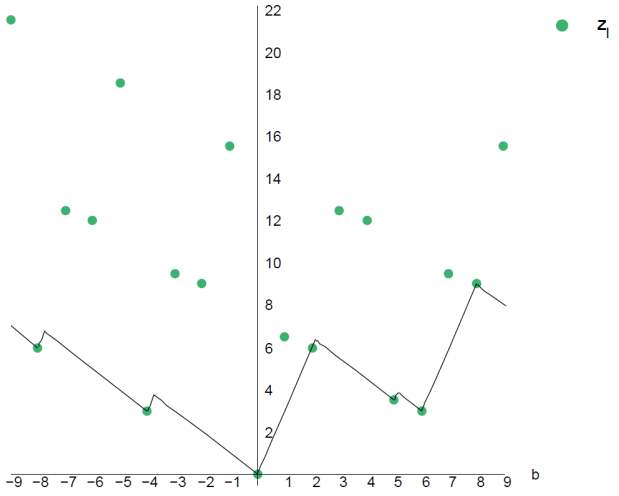


Figure 2.14: The value function of the integer restriction of (2.15) for $b \in [-9, 9]$.

Hence, this approach does not seem to be the solution. Instead, we anticipate overcoming this difficulty in two different ways, depending on the context in which the value function is needed. First, working with a subset of S_{min} still yields an upper approximation of z, which might be useful in particular applications where an approximate solution will suffice. Second, we anticipate that in most cases, it would be possible to dynamically generate the set S_{min}, adding points only as necessary for improving the approximation in the part of the domain required for solution of a particular instance. This approach would be similar to that of using dynamic cut generation to solve fixed MILPs.

We demonstrate the potential of both such techniques here by describing a method for iteratively improving a given discrete approximation of the value function by dynamically generating improving members of S_I after the fashion of a cutting plane algorithm for MILP. At iteration k, we begin with an approximation arising from $S^k \subseteq S_I$ using the

formula

$$\bar{z}(b) = \inf\{c_I^\top x_I + z_C(b - A_I x_I) : x_I \in S_I^k, z(A_I x_I) = c_I^\top x_I\} \tag{2.37}$$

and we generate the set S^{k+1} by determining the point at which the current approximation is maximally different from the true value function. This is akin to generation of the most violated valid inequality in the case of MILP. An important feature of the algorithm is that it produces a performance guarantee after each step, which bounds the maximum gap between the approximation and the true function value. In what follows, we denote the current upper bounding function by $\bar{z}$.

In addition to the initial assumption $z(0) = 0$, we also assume the set B_I is non-empty and bounded (while B can remain unbounded) to guarantee finite termination. Note, however, that it is possible to apply the algorithm even if this is not the case. It is a simple matter, for example, to generate the value function within a given box, even if B_I is an unbounded set. We note that we do not require the assumption $\mathcal{K} = \mathbb{R}^m$, although this is not a restrictive assumption in practice anyway, since A_C can always be modified to satisfy this assumption (Kall and Mayer, 2010).

Algorithm

Initialize: Let $\bar{z}(b) = \infty$ for all $b \in B$, $\Gamma^0 = \infty$, $x_I^0 = 0$, $S^0 = \{x_I^0\}$, and $k = 0$.

while $\Gamma^k > 0$ **do**:

- Let $\bar{z}(b) = \min\{\bar{z}, \bar{z}(b; x_I^k)\}$ for all $b \in B$.
- $k \leftarrow k + 1$.
- Solve

$$\begin{aligned} \Gamma^k = \max\ & \bar{z}(b) - c_I^\top x_I \\ \text{s.t. } & A_I x_I = b \\ & x_I \in \mathbb{Z}_+^r. \end{aligned} \tag{2.38}$$

  to obtain x_I^k.

- Set $S^k \leftarrow S^{k-1} \cup \{x^k\}$

end while

return $z(b) = \bar{z}(b)$ for all $b \in B$.

The key to this method is effective solution of (2.38). We show how to formulate this problem as a mixed integer nonlinear program below. For practical computation, (2.38) can be rewritten conceptually as

$$\begin{aligned} \Gamma^k = \max\ & \theta \\ \text{s.t. } & \theta \le \bar{z}(b) - c_I^\top x_I \\ & A_I x_I = b \\ & x_I \in \mathbb{Z}_+^r. \end{aligned} \tag{2.39}$$

The upper approximating function $\bar{z}(b)$ is a non-convex and non-concave piecewise polyhedral function that is obtained by taking the minimum of a finite number of convex piecewise polyhedral functions $\bar{z}$. In particular, in iteration $k > 1$ of the algorithm we have $\bar{z}(b) = \min_{i=1,\ldots,k-1} \bar{z}(b; x_I^i)$. Therefore, the first constraint in (2.38) can be reformulated as $k-1$ constraints, the right-hand side of each of which is a convex piecewise polyhedral function.

$$\theta + c_I^\top x_I \le c_I^\top x_I^i + z_C(b - A_I x_I^i) \quad i = 1, \ldots, k-1. \tag{2.40}$$

Next, we can write z_C as

$$z_C(b - A_I x_I^i) = \sup\{(b - A_I x_I^i)^\top \nu^i : A_C^\top \nu^i \le c_C, \nu^i \in \mathbb{R}^m\} \tag{2.41}$$

and reformulate each of $k-1$ constraints in (2.40) as

$$\begin{aligned} & \theta + c_I^\top x_I \le c_I^\top x_I^i + (b - A_I x_I^i)^\top \nu^i \\ & A_C^\top \nu^i \le c_C \\ & \nu^i \in \mathbb{R}^m \end{aligned} \tag{2.42}$$

for $i \in \{1, \ldots, k-1\}$. Together, then, in each iteration we solve

$$\begin{aligned} \Gamma^k = \ & \max \theta \\ & \text{s.t. } \theta + c_I^\top x_I \leq c_I^\top x_I^i + (A_I x_I - A_I x_I^i)^\top \nu^i \quad i = 1, \ldots, k-1 \\ & A_C^\top \nu^i \leq c_C \quad i = 1, \ldots, k-1 \\ & \nu^i \in \mathbb{R}^m \quad i = 1, \ldots, k-1 \\ & x_I \in \mathbb{Z}_+^r. \end{aligned} \tag{2.43}$$

Due to the first constraint, the resulting problem is a non-linear optimization problem. Nevertheless, solvers do exist, e.g. Couenne (Belotti, 2009) that are capable of solving these problems. Assuming that there is a finite method to solve (2.43), we next show that the proposed algorithm terminates finitely and returns the correct value function.

Theorem 2.10. (Algorithm for Construction) *Under the assumptions that B_I is non-empty and bounded and (2.43) can be solved finitely, the algorithm terminates with the correct value function in finitely many steps.*

Proof. For any $x_I \in S_I$, $c_I^\top x_I \geq z(b)$ for all $b \in B$. From Proposition 2.12, we have that for $x_I \in S_{min} \subseteq S_I$, $c_I^\top x_I = z(A_I x_I)$. Therefore, for the solution of (2.43) at iteration k we have $x_I^k \in S_I$ and $c_I^\top x_I^k = z(A_I x_I^k)$. Since B_I is assumed to be bounded, then there is a finite number of such points that can be generated in the algorithm. That is, $\bar{z}$ can only be updated a finite number of times.

To see that at termination, $\bar{z}$ is the value function, first note that Proposition 2.9 implies that the initialization and the updates of the approximating function result in valid upper bounding functions. If in iteration k, the approximation $\bar{z}(b)$ is strictly above the value function at some $b \in B$, then $\Gamma^k > 0$ and there is some $x_I \in S_I$ for which $c_I x_I$ lies on the value function and below the approximation. The subproblem is guaranteed to find such a point, therefore, in each intermediate iteration we improve the approximation. When no such a point is found, the approximation is exact everywhere and we terminate with $\Gamma^k = 0$. □ □

To illustrate, we apply the algorithm to two value functions: the first one is the func-

tion (2.15). The second value function is from two-stage stochastic integer optimization literature and refers to the value function of the second-stage problem of the stochastic server location problem (SSLP) in (Ntaimo and Sen, 2005).

Example 2.19. Consider (2.15) where $x_1, x_2, x_3 \in \{1, \ldots, 5\}$. Figure 2.15 plots Γ^k normalized by Γ^1, the initial gap reported with $\bar{z} = z_C$, versus the iteration number for problem (2.43) . When the algorithm is executed, over $b \in [-7, 7]$, the updates only occur for $\hat{x_I}$ such that $A_I\hat{x_I} \in \{-4, 5, 6\}$. This is because the remainder of the right-hand sides $A_I\hat{x_I}$ in $[-7, 7]$ correspond to $(A_I\hat{x_I}, c_I^\top\hat{x_I})$ (green circles in Figure 2.14) that lie either on or above z_C (and therefore below the following updated approximating functions). □

The proposed algorithm can be applied to MILPs with inequality constraints by adding appropriate non-negativity restrictions to the dual variables ν in (2.43). We see an example next.

Example 2.20. Consider the second-stage problem of SSLP with 2 potential server locations and 3 potential clients. The first-stage variables and stochastic parameters are captured in the right-hand sided $b_1, \ldots, b_5$. The resulting formulation is

$$\begin{aligned} z(b) = \min\ & 22y_{12} + 15y_{21} + 11y_{22} + 4y_{31} + 22y_{32} + 100R \\ \text{s.t. } & 15y_{21} + 4y_{31} - R \leq b_1 \\ & 22y_{12} + 11y_{22} + 22y_{32} - R \leq b_2 \\ & y_{11} + y_{12} = b_3 \\ & y_{21} + y_{22} = b_4 \\ & y_{31} + y_{32} = b_5 \\ & y_{ij} \in \mathbb{B}, i \in \{1, 2, 3\}, j \in \{1, 2\}, R \in \mathbb{R}_+. \end{aligned} \tag{2.44}$$

The normalized gap $\frac{\Gamma^k}{\Gamma^1}$ versus the iteration number k is plotted in Figure 2.15. For this example, non-positivity constraints on the dual variables corresponding to the first two constraints are added to (2.43). □

As one can observe in Figure 2.15, the quality of approximations improves significantly

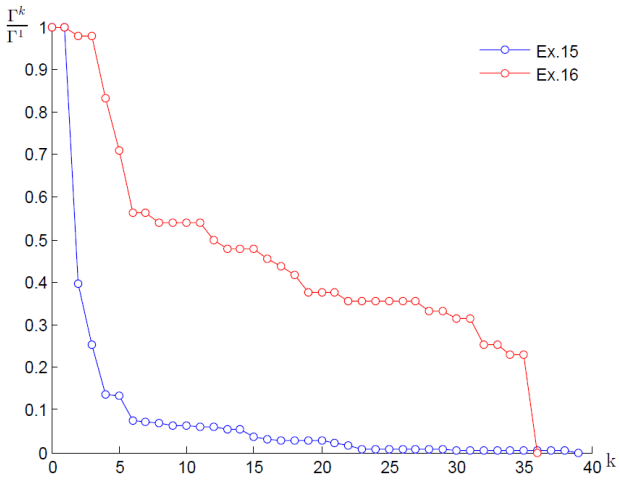


Figure 2.15: Normalized approximation gap vs. iteration number.

as the algorithm progresses. The upper-approximating functions $\bar{z}$ obtained from the intermediate iterations of the algorithm can be utilized within other solution methods that rely on bounding a MILP from above. Clearly, such piecewise approximating functions $\bar{z}$ are structurally simpler than the original MILP value function. Furthermore, like the SSLP, a common class of two-stage stochastic optimization problems considers stochasticity in the right-hand side. With a description of the value function of the second-stage problem, finding the solution to different second-stage problems reduces to evaluations of the value function at different right-hand sides. The proposed algorithm therefore can be incorporated in methods to solve stochastic optimization problems with a large number of scenarios.

2.6 Approximation Methods

The dual functions to the value function of a MILP introduced in Section 2.1 essentially provide lower bounding functions for the value function, however, the dual problem introduced in (1.14) is rather general and we are yet to find methods to construct dual functions in practice. We then turn our attention to constructing functions that bound the MILP value function from above.

We first review certain methods to construct dual functions. The overview we are

about to provide is not meant to be comprehensive, instead, we focus on the dual functions that can be constructed by known relaxations and solution algorithms for integer optimization problems. These dual functions are important computationally, as they can be generated as a given MILP instance is solved with the algorithm. There are other approaches to construct dual functions by finite procedures to obtain explicit and closed form dual functions. Examples of this approach are the methods developed by Laatsch et al. (1964) and Klabjan (2007) for PILPs, both of which result in subadditive dual functions. For a detailed overview of these methods we refer to (Güzelsoy, 2009). Here, we focus on dual functions that can be derived from commonly used algorithms to solve MILPS, namely, the cutting plane, branch-and-bound, branch-and-cut, and Lagrangian relaxation algorithms. We use the dual functions from the branch-and-bound method extensively in the next chapter to solve two-stage mixed integer optimization problems.

Cutting plane. The idea behind *cutting plane* algorithms is to find the optimal solution of a given instance of MILP by iteratively refining the approximation of the convex hull of S, which we denote by $convS$. In each iteration of the algorithm, a relaxation of the original MILP is solved to obtain a point $t \in \mathbb{R}^n$. Then, valid inequalities are generated to separate $conv(S)$ from the point t. In this step, either a valid inequality $\pi x \geq \pi_0, \Pi \neq 0$ for all $x \in conv(S)$ is constructed or no valid inequality is constructed in the case that the point t belongs to the convex hull of S, in which case it is the optimal solution of the original MILP. The generated hyperplane $\pi x \geq \pi_0$ are called *cutting planes* or *cuts*. Frequently, the algorithm starts with the linear relaxation of the original MILP and in the iteration k of the algorithm, the following problem is solved

$$\begin{aligned} \inf\ & c^\top x \\ \text{s.t. } & Ax = \tilde{b} \\ & \Pi x \geq \Pi_0 \\ & x \in \mathbb{R}^n_+, \end{aligned} \tag{2.45}$$

where $\Pi = [\pi^1, \ldots, \pi^k]^\top \in \mathbb{R}^{k \times n}$ and $\Pi_0 = [\pi_0^1, \ldots, \pi_0^k]^\top \in \mathbb{R}^k$ are respectively the matrix of coefficients and the vector of the right-hand sides of the k cuts generated in the algorithm so far. From the weak duality theorem for LPs, we can derive lower bound for the objective value of the original MILP from the solution to the dual of the LP (2.45)

$$\begin{aligned} &\sup \tilde{b}^\top \eta + \Pi_0^\top w \\ &\text{s.t. } A^\top \eta + \Pi^\top w \leq c \\ &\eta \in \mathbb{R}^m, w \in \mathbb{R}^k_+, \end{aligned} \tag{2.46}$$

where η and w respectively denote the dual variables corresponding to the original constraints and added cuts. The objective function of (2.46) is also a dual function to the value function of the LP (2.45). However, it does not directly yield to a dual function for the original MILP instance, because the valid inequalities in (2.45) can be only valid for $conv(S)$. As $\tilde{b}$ is modified, the previously generated valid inequalities may no longer be valid for the new polyhedral. The question that follows is that whether the valid inequality can be formulated parametrically in $\tilde{b}$ such that it remains valid for the convex hull of the modified polyhedral with modified $\tilde{b}$. Johnson (1973) and Jeroslow (1978) show that in fact, any valid inequality for $conv(S)$ is equivalent or dominated by a valid inequality in the form of

$$\sum_{j \in I} F(A^j) x_j + \sum_{j \in C} \bar{F}(A^j) x^j \geq F(b), \tag{2.47}$$

where F is a subadditive function and $\bar{F}$ is the upper d-directional derivative defined earlier in (2.4). The following result states this formally.

Theorem 2.11. *If F is a subadditive function with $F(0) = 0$ and with the upper d-directional derivatives existing, we have*

$$\sum_{j \in I} F(A^j) x_j + \sum_{j \in C} \bar{F}(A^j) x^j \geq F(b). \tag{2.48}$$

Conversely, if $\pi x \geq \pi_0$ is a valid inequality for $conv(S)$, then there is a subadditive

function F_π such that

$$\begin{aligned} F_\pi(A^j) &\leq \pi_j \quad \forall j \in I \\ \bar{F}_\pi(A^j) &\leq \pi_j \quad \forall j \in C \\ F_\pi(\tilde{b}) &\geq \pi_0. \end{aligned} \tag{2.49}$$

In theory, therefore, if the subadditive representation of each is known, the i^{th} cut in the form of $\pi x \geq \pi_0$ can be written parametrically as a function of the right-hand side $b \in \mathbb{R}^m$ in the form of

$$\sum_{j \in I} F_i(\sigma_i(A^j))x_j + \sum_{j \in C} \bar{F}_i(\bar{\sigma}_i(A^j))x^j \geq F_i(\sigma_i(b)), \tag{2.50}$$

to ensure the validity of the cut as the right-hand side varies, where σ and $\bar{\sigma}$ are functions mapping $\mathbb{R}^m$ to $\mathbb{R}^{m+i-1}$ and are defined by

$$\begin{aligned} &\sigma_1(b) = \bar{\sigma}(b) = b, \\ &\sigma_i(b) = [b \; F_1(\sigma_1(b)) \; \ldots \; F_{i-1}(\sigma_{i-1}(b))] \quad \text{for } i \geq 2, \\ &\bar{\sigma}_i(b) = [b \; \bar{F}_1(\bar{\sigma}_1(b)) \; \ldots \; \bar{F}_{i-1}(\bar{\sigma}_{i-1}(b))] \quad \text{for } i \geq 2. \end{aligned} \tag{2.51}$$

Let (η^k, w^k) be an optimal solution to (2.46) in iteration k of the algorithm. Then the following is a feasible function to the subadditive dual problem (2.4).

$$\underline{z}(b) = \eta^k b + \sum_{i=1}^{k} w_i^k F_i(\sigma_i(b)). \tag{2.52}$$

In practice, obtaining a closed-form subadditive representation of each cut is not computationally feasible. For certain family of cuts, however, a closed-form subadditve representation is obtained. In the case of Gomory fractional cuts, for example, Wolsey (1981a) derived the subadditve representation when these cuts are applied to an LP relaxation of a PILP. This representation takes the form of

$$F_i(b) = \left\lceil \sum_{k=1}^{m} \lambda_k^{i-1} b_k + \sum_{k=1}^{i-1} \lambda_{m+k}^{i-1} F_k(b) \right\rceil, \tag{2.53}$$

where $\lambda^{i-1} = (\lambda_1^{i-1}, \ldots, \lambda_{m+i-1}^{i-1}) \geq 0$. Then, (2.53) can be used in (2.52) to find a dual function. Llewellyn and Ryan (1993) later extended this idea and developed a primal-dual algorithm with subadditive dual functions from Gomory fractional cuts to solve PILPs. In the case where the valid inequalities for a knapsack problem are *cover* cuts, Schrage and Wolsey (1985) derived an explicit formulation of the cut as a function of the right-hand side b.

Branch-and-bound. The principle of the branch-and-bound method is to construct disjunctions of the feasible region of the LP relaxation of a given MILP instance and keeping them in a tree structure. During the execution of the algorithm a global upper bound (initialized with $+\infty$) and a list of *candidate* problems corresponding to the leaf nodes of the tree (initialized with an empty set) are kept. The algorithm starts with solving the LP relaxation of the MILP in the *root node* of a tree. This problem is added to the candidate list. In every iteration, a problem is selected from the candidate list and solved to optimality. If the solution to this problem satisfies the integrality restrictions of the original MILP, its solution is feasible to the original MILP and its objective value is used to potentially decrease the upper bound. In the case that some variables $x_j, j \in I$ take fractional values k_j in a nodal problem, one such variable is selected and *branching* is carried out by adding constraints in the form of $x_j \leq \lfloor k_j \rfloor$ and $x_j \leq \lfloor k_j \rfloor + 1$ to the description of the feasible region of the nodal problem to create two new subproblems. Candidate nodes are frequently checked to see if they can be *pruned*, in which case their corresponding problems are deleted from the candidate list. A node is pruned if its objective value is greater than the upper bound. This procedure is commonly performed after solving a new candidate. The algorithm terminates when there is no candidate left in the list. The upper bound and its corresponding solution are respectively the optimal objective value and solution of the original MILP.

Consider evaluation of (MILP) by a branch-and-bound algorithm. The problem solved

at an arbitrary node t of the tree takes the form

$$z^t(b) = \min\{c^\top x \mid x \in S^t(b)\}, \tag{P^t}$$

where

$$S^t(b) = \{x \in \mathbb{R}^n_+ \mid Ax = b, l_t \leq x \leq u_t\}, \tag{2.54}$$

and $l_t, u_t \in \mathbb{Z}^n_+$ are the branching bounds applied to the variables (the upper bounds may be $+\infty$). When the right-hand side is fixed to $\hat{b} \in \mathbb{R}^m$, LP dual of (P^t) is

$$\max\ \{\hat{b}^\top \eta^t + {l_t}^\top \underline{\eta}^t - u_t^\top \overline{\eta}^t \mid (\eta^t, \underline{\eta}^t, \overline{\eta}^t) \in \mathcal{D}\}, \tag{D^t}$$

where η^t represents the dual variables associated with the matrix A from the original formulation and $\underline{\eta}^t, \overline{\eta}^t$ are the dual variable associated with the lower and upper bound constraints, respectively. The feasible set of the dual is

$$\mathcal{D} = \{(\eta^t, \underline{\eta}^t, \overline{\eta}^t) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \mid A^\top \eta^t + \underline{\eta}^t - \overline{\eta}^t \leq c, \underline{\eta}^t \geq 0, \overline{\eta}^t \geq 0\}. \tag{2.55}$$

The dual problem only depends on $\hat{b}$ through its objective function, so the set $\mathcal{D}$ is independent of b. By LP duality, we then have that for any $(\eta^t, \underline{\eta}^t, \overline{\eta}^t) \in \mathcal{D}$ and $b \in \mathbb{R}^{m_2}$,

$$b^\top \eta^t + {l_t}^\top \underline{\eta}^t - u_t^\top \overline{\eta}^t \leq c^\top x. \tag{2.56}$$

From the left-hand side of (2.56), we have a dual function for the value function of the LP solved at node t. Wolsey (1981a) was the first to observe this and extend it to construct dual functions for the value function of the original MILP, z.

Theorem 2.12. *(Wolsey, 1981a) Let $\hat{b} \in \mathbb{R}^m$ be such that $z(\hat{b}) < \infty$ and suppose T indexes the set of leaf nodes of a branch-and-bound tree resulting from evaluation of $z(\hat{b})$. Then there exists a function $\underline{z}(b)$ dual to z defined by*

$$\underline{z}(b) = \min_{t \in T}\ (b^\top \eta^t + \alpha^t) \quad \forall b \in \mathbb{R}^m, \tag{2.57}$$

where $\eta^t \in \mathbb{R}^m$ and $\alpha^t \in \mathbb{R}$ are feasible solutions to the dual of the LP relaxation associated with node t and for which $\underline{z}(\hat{b}) = z(\hat{b})$.

Proof. Consider evaluating $z(\hat{b})$ by a branch-and-bound procedure and let $t \in T$ be the index of a given node in the final branch-and-bound tree. The bound yielded by the associated subproblem is obtained by evaluation of the value function of its LP relaxation, which is of the form of (P^t). We consider two cases.

1. If $z^t(\hat{b}) < \infty$, then letting $(\hat{\eta}^t, \hat{\underline{\eta}}^t, \hat{\overline{\eta}}^t)$ be the optimal solution to (D^t), we have that

$$\underline{z}^t(b) = b^\top \hat{\eta}^t + {l_t}^\top \hat{\underline{\eta}}^t - u_t^\top \hat{\overline{\eta}}^t = b^\top \hat{\eta}^t + \alpha^t \tag{2.58}$$

   is dual to z^t.

2. If $z^t(\hat{b}) = \infty$, we let $(\hat{\eta}^t, \hat{\underline{\eta}}^t, \hat{\overline{\eta}}^t)$ be any member of $\mathcal{D}$ such that

$$b^\top \hat{\eta}^t + {l_t}^\top \hat{\underline{\eta}}^t - u_t^\top \hat{\overline{\eta}}^t > z(\hat{b}). \tag{2.59}$$

   Such a member of $\mathcal{D}$ must exist since $z^t(\hat{b}) = \infty$. Then by the same argument as we gave previously,

$$b^\top \hat{\eta}^t + {l_t}^\top \hat{\underline{\eta}}^t - u_t^\top \hat{\overline{\eta}}^t = b^\top \hat{\eta}^t + \alpha^t \tag{2.60}$$

   is dual to z^t.

Finally, by taking the minimum over the set of dual functions for the individual nodes, we obtain the function

$$\underline{z}(b) = \min_{t \in T} \underline{z}^t(b) \leq z(b), \tag{2.61}$$

which is dual to z and has the form posited in the statement of the theorem.

It remains to show that $\underline{z}$ is strong. To see this, note that there must exist a node $t^* \in T$ such that $\underline{z}^{t^*}(\hat{b}) = z(\hat{b})$. Furthermore, we must have that $\underline{z}^t(\hat{b}) \geq z(\hat{b})\ \forall t \in T$, since for each $t \in T$, we have either that (P^t) is feasible (in which case $\underline{z}^t(\hat{b}) \geq z(\hat{b})$ by the optimality of the branch-and-bound tree) or (P^t) is infeasible (in which case $\underline{z}^t(\hat{b}) \geq z(\hat{b})$ by construction). □

One subtle point we should address further has to do with the infeasible nodes. In the proof above, we simply appealed to the existence of a dual solution that could be used to construct an appropriate dual function. In practice, we need to be able to obtain such dual solution in a practical way. Consider again a node $t \in T$ for which $z^t(\hat{b}) = \infty$. One method of obtaining an appropriate dual solution is to let $(\sigma^t, \underline{\sigma}^t, \overline{\sigma}^t)$ be an extreme ray of $\mathcal{D}$ that proves the infeasibility of (P^t) (produced by the simplex algorithm used to solve the LP) and let $(\tilde{\eta}^t, \underline{\tilde{\eta}}^t, \overline{\tilde{\eta}}^t)$ be the member of $\mathcal{D}$ generated just prior to discovery of the dual ray. By adding an appropriately chosen scalar multiple of the ray to this dual solution, we obtain a second dual solution with the desired property. More formally, let $\lambda \in \mathbb{R}^+$ be a given scalar and consider

$$(\hat{\eta}^t, \underline{\hat{\eta}}^t, \overline{\hat{\eta}}^t) = (\tilde{\eta}^t, \underline{\tilde{\eta}}^t, \overline{\tilde{\eta}}^t) + \lambda(\sigma^t, \underline{\sigma}^t, \overline{\sigma}) \tag{2.62}$$

By choosing λ large enough, we obtain a solution appropriate for use in (2.60). In practice, we may also avoid this issue by putting an explicit bound on the dual objective function value, since once the objective value of the current dual solution exceeds the global upper bound, the solution method can be terminated. In this case, the dual solution generated in the last iteration would itself be a solution that has the required property.

Due to the constant term added to the (D^t) from branching, the function $\underline{z}$ associated with a given branch-and-bound tree is not a subadditve function but is concave and piecewise polyhedral. This can impose challenges to encode and use it. In Chapter 3, we apply such a dual function to two-stage mixed integer optimization problems and reformulate it such that it can be used in general MILP solvers.

Branch-and-cut. The branch-and-cut procedure improves on branch-and-bound by allowing the nodal problems of the tree to be solved with the cutting plane method. In this case, the additional cuts change the description of the nodal problem (P^t). The resulting subproblems are in the form of (P^t) with the feasible region is defined as

$$S^t(b) = \{x \in \mathbb{R}^n_+ \mid Ax = b, l_t \leq x \leq u_t, \Pi_t x \geq \Pi_{0,t}\}, \tag{2.63}$$

where $l_t, u_t \in \mathbb{Z}^n_+$ are the branching bounds applied to the variables and the matrices $\Pi \in \mathbb{R}^{k(t)\times n}$ and $\Pi^t_0 \in \mathbb{R}^{k(t)}$ determine the cuts added to the subrproblem of node t. Following the steps taken for the branch-and-bound procedure, we write the dual of the subproblem

$$\max\{\hat{b}^\top \eta^t + {l_t}^\top \underline{\eta}^t - u_t^\top \overline{\eta}^t + \Pi_{0,t}^\top w^t \mid (\eta^t, \underline{\eta}^t, \overline{\eta}^t) \in \mathcal{D}\}, \tag{2.64}$$

where η^t represents the dual variables associated with the matrix A from the original formulation and $\underline{\eta}^t, \overline{\eta}^t$ are the dual variable associated with the lower and upper bound constraints, respectively. The feasible set of the dual is

$$\mathcal{D} = \{(\eta^t, \underline{\eta}^t, \overline{\eta}^t, w^t) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{k(t)} \mid A^\top \eta^t + \underline{\eta}^t - \overline{\eta}^t + \Pi_t^\top w^t \leq c, \underline{\eta}^t, \overline{\eta}^t, w^t \geq 0\}, \tag{2.65}$$

where $w^t \in \mathbb{R}^{k(t)}$ is the dual variable associated with the constraints $\Pi_t x \geq \Pi_{0,t}$. As discussed in the cutting plane method, the added cuts may not be valid for the feasible region of the MILP with a modified right-hand side. In the case that a subadditive representation of each of the $k(t)$ inequalities in $\Pi_t x \geq \Pi_{0,t}$ is known and can be written as (2.47), following the steps of the dual function derived for the branch-and-bound method, we can construct a dual function for the value function of the original MILP. This is shown in the next theorem.

Theorem 2.13. *(Güzelsoy, 2009) Let $\hat{b} \in \mathbb{R}^m$ be such that $z(\hat{b}) < \infty$ and suppose T indexes the set of leaf nodes of a tree resulting from evaluation of $z(\hat{b})$ by the branch-and-cut procedure. Then there exists a function $\underline{z}(b)$ dual to z defined by*

$$\underline{z}(b) = \min_{t\in T} \{b^\top \eta^t + \alpha^t + \sum_{i=1}^{k(t)} w_i^t F_i^t(\sigma_i^t(b))\} \quad \forall b \in \mathbb{R}^m, \tag{2.66}$$

and we have $\underline{z}(\hat{b}) = z(\hat{b})$.

The resulting dual function is not subadditive. Güzelsoy (2009) considers the case

where the MILP has explicit upper and lower bounds on all variables and constructs a subadditive dual function in the form of (2.66) by including the bounds as part of the right-hand side.

We now turn our attention to finding functions that approximate the value function from above. Like our discussion on dual functions, we are interested in upper approximating functions that are strong with respect to a given right-hand side $\hat{b} \in B$. We found an upper approximating function for the MILP value function in Proposition 2.9. The function $\bar{z}$, defined in (CR), is not guaranteed to be strong at any right-hand side, i.e., it can lie strictly above the value function. From our discrete representation in Theorem 2.7 and the third part of Theorem 2.8, it follows that if the integer part of the solution $\hat{x_I}$ belongs to the set S_{min}, then the continuous restriction $\bar{z}(\cdot, \hat{x_I})$ is strong over the maximal stability set $\text{int}(B_{LS}(A_I \hat{x}_I))$ (see Figure 2.9 for an example).

The continuous restriction function $\bar{z}$ is obtained from a specific *restriction* of the MILP value function by partitioning variables into two sets, the sets of integer and continuous variables. Other upper bounding functions, however, can be obtained from different partitioning schemes. The following result generalizes Proposition 2.9.

Proposition 2.16. *(Güzelsoy, 2009) Let $H \subseteq N$ and $t_i \in \mathbb{R}_+, i \in H$ be given and define the function $G : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\}$ such that*

$$G(b) = \sum_{i \in H} c_i t_i + z_{N \setminus H}(b - \sum_{i \in H} A^i t_i) \ \forall b \in \mathbb{R}^m, \tag{2.67}$$

where

$$\begin{aligned} z_{N \setminus H}(d) \ &= \min \sum_{i \in N \setminus H} c_i x_i \\ \text{s.t.} \ &\sum_{i \in N \setminus H} A^i x_i = d \\ &x_i \in \mathbb{Z}_+, i \in I, x_i \in \mathbb{R}_+, i \in C. \end{aligned} \tag{2.68}$$

Then, $G(b) \geq z(b)$ for all $d \in \mathbb{R}^m$, if $t_i \in \mathbb{Z}_+$ for $i \in I \cap H$ and $t_i \in \mathbb{R}_+$ for $i \in C \cap H$.

Clearly, the above result reduces to Proposition 2.9 if we let $H = I$.

Another way to obtain an upper bounding function for the value function is to extend an existing local description of the value function to obtain an upper bounding function for the value function over its entire domain. To describe this method, we first define an *extension* of a function.

Definition 2.6. Given a function $g : \Lambda \to \mathbb{R}$ with $\Lambda \subset \mathbb{R}^m$, we call any function G an extension of g from Λ to $\mathbb{R}^m$ if

- $G(b) = g(b)\ \forall b \in \Lambda$ and
- for each $b \in \mathbb{R}^m \setminus \Lambda$, there exists a collection $\mathcal{C}(b) = \{\rho_1, \ldots, \rho_R\}$ defined in Λ with the property $\sum_{\rho \in \mathcal{C}(b)} \rho = b$ and $G(b) = \sum_{\rho \in \mathcal{C}(b)} g(\rho)$.

Let $q \in \mathbb{Q}^m_+$ be defined as the vector of the maximum coefficients of the rows of A. That is,

$$q_i = \max\{a_{ij} : j \in N\}.\ \forall i = 1, \ldots, m \tag{2.69}$$

The following proposition states the conditions to obtain an upper bounding function for the value function through constructing extensions functions.

Proposition 2.17. *(Güzelsoy, 2009) Let q be defined as in (2.69) and g be a function from $[0, q]$ to $\mathbb{R}$ such that $g(b) \geq z(b)$ for $b \in [0, q]$, where z is the value function (MVF). Then, if G is an extension of g from $[0, q]$ to $\mathbb{R}^m_+$, then $G(b) \geq z(b)$ for all $b \in \mathbb{R}^m_+$.*

The specifics of the extension function in Proposition 2.17 depends on the set $\mathcal{C}(b)$. Obviously, the functions that approximate the value function closely are desirable but they may be expensive to construct. Güzelsoy (2009) showed that, in fact, it is possible to construct the MILP value function using the described extension procedure. Specifically, the value function can be constructed when it is used as the seed function over $[0, q]$ and the function $G(b) = \min_{C(b) \in \mathcal{C}} \sum_{\rho \in \mathcal{C}(b)} z(\rho)$ is used elsewhere, where $\mathcal{C}$ denotes the set of all finite collections $\{\rho_1, \ldots, \rho_R\}$ such that $\sum_{i=1}^{R} \rho_i = b$. This method is called the *maximal subadditive extension*. We refer the reader to (Güzelsoy, 2009) for more details.

Chapter 3

Algorithms for Two-stage Stochastic Optimization

In Chapter 1, we introduced the stochastic mixed integer linear problem

$$\min_{x \in S_1} \Psi(x) = \min_{x \in S_1} c^\top x + \Xi(x), \tag{3.1}$$

where $S_1 = \{x \in \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1 - r_1} \mid Ax = \tilde{b}\}$ and for a given first-stage decision $\hat{x}$, $\Xi(\hat{x})$ represents the average cost of implementing second-stage (recourse) decisions taken to correct the outcome of implementing $\hat{x}$. The function Ξ is defined by

$$\Xi(x) = \mathbb{E}_{\omega \in \Omega}[z(h_\omega - T_\omega x)], \tag{3.2}$$

for $x \in S_1$, where $T_\omega \in \mathbb{Q}^{m_2 \times n_1}$ and $h_\omega \in \mathbb{Q}^{m_2}$ represent the realized values of the stochastic inputs to the second stage for scenario $\omega \in \Omega$. The function z is the *second-stage value function*, which encodes the cost of the recourse decision for a given first-stage solution x and realization ω defined for any $b \in \mathbb{R}^{m_2}$, we have

$$z(b) = \inf\{q^\top y \mid y \in S_2(b)\}, \tag{RV}$$

where $S_2(b) = \{y \in \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2-r_2} \mid Wy = b\}$. We dedicated the last chapter to study the function z. We let B_2 denote the set of right-hand sides in the second-stage problem, i.e., $B_2 = \cup_{\omega\in\Omega}\{h_\omega - T_\omega x \mid x \in S_1\}$. We assume the following:

A1 $\Xi(x) = \mathbb{E}_{\omega\in\Omega}\left[z(h_\omega - T_\omega x)\right]$ is finite for all $x \in S_1$.

A2 S_1 is compact.

A3 The random variable ω is drawn from a discrete distribution with finite support.

Assumptions A1–A3 are not restrictive and are common in the literature. In Assumption A1, $\Xi[x] < +\infty$ requires the feasibility of the recourse problem for any right-hand side $h_\omega - T_\omega x$, $\omega \in \Omega$ with a given $x \in S_1$. This is known as the *relative recourse* property and can be guaranteed by adding artificial variables to the recourse problem. By $\Xi[x] > -\infty$, we require the dual polyhedron of the linear relaxation of the recourse problem to be nonempty. We make Assumption A2 to guarantee the finite termination of the Generalized Benders' algorithm we introduce in Section 4.3 (this is not required to use the algorithm in practice).

Assumption A3 assures the expectation in (SP) can be expressed as the sum of a finite number of terms, which allows for reformulation of (SP) as the deterministic equivalent problem (DE) we introduced earlier

$$\begin{aligned} \min\ & c^\top x + \sum_{\omega\in\Omega} p_\omega q^\top y_\omega \\ \text{s.t. } & Ax = b \\ & T_\omega x + W y_\omega = h_\omega \quad \forall \omega \in \Omega \\ & x \in \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1-r_1}, y \in \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2-r_2}. \end{aligned} \tag{DE}$$

For a fixed x, (SP) can be separated into $|\Omega|$ independent subproblems, so applying a decomposition method directly to this reformulation is one possible solution approach. On the surface, such a decomposition by scenario makes sense, but one argument against this tactic is that with a large number of scenarios, there may be substantial overlap in the areas of the value function that are relevant in each scenario. One the other hand,

the Benders' method relies on obtaining approximating functions for the value function that can be used across all scenarios. In this chapter, we apply this method to the general two-stage mixed integer linear optimization problems. Next, we provide a brief overview of the structure and solution methods of the continuous variation of (SP), where $r_1 = r_2 = 0$, the continuous two-stage stochastic optimization problem.

3.1 The Continuous Case

We studied the value function of a linear optimization problem in the Chapter 2. Naturally, in a continuous two-stage stochastic optimization problem, the second-stage value function (RV) adopts the same properties. The following results formalize this.

Proposition 3.1. *(Ruszczynski and Shapiro, 2003) The second-stage value function* $z_{LP} : \mathbb{R}^n \to \mathbb{R} \cup \pm\infty$ *is lower semi-continuous at every point of its domain iff its epigraph is a closed subset of* $\{\mathbb{R}^n \times \mathbb{R}\}$.

Theorem 3.1. *(Birge and Louveaux, 1997; Kall and Mayer, 2010) For a continuous two-stage stochastic program in the form of* (SP) *such that* $z(h_\omega - T_\omega x) > -\infty$ *for all* $x \in S_1$ *and* $\omega \in \Omega$, z *is*

- *a piecewise polyhedral convex function in* (h_ω, T_ω) *for all* $\omega \in \Omega$,
- *a piecewise polyhedral convex function in* x *for all* $x \in S_1$.
- *a piecewise polyhedral concave function in* q,

Proof. The first and second properties follow from the convexity of the value function of a LP in its right-hand side. To show the third property, for a given $\hat{x} \in S_1$ let $q_\lambda = \lambda q_1 + (1-\lambda)q_2, y^*_\lambda$ optimal for $z_{LP}(q_\lambda), y^*_1$ optimal for $z_{LP}(q_1)$ and y^*_2 optimal for $z_{LP}(q_2)$. We have:

$$\begin{aligned} &\lambda z_{LP}(q_1) + (1-\lambda) z_{LP}(q_2) = \lambda q_1^\top y^*_1 + (1-\lambda) q_2^\top y^*_2 \\ &\leq \lambda q_1^\top y^*_\lambda + (1-\lambda) q_2^\top y^*_\lambda = q_\lambda^\top y^*_\lambda = z_{LP}(q_\lambda), \end{aligned} \tag{3.3}$$

where the inequality follows from the feasibility of $y_1^*, y_2^*, y_\lambda^*$ for the corresponding problem. Finally, the piecewise linearity is a result of the representation of the value function of an LP in (2.18). □

From Proposition (1.2) we have that z_{LP} is convex and subdifferentiable at any right-hand side and at any given right-hand side, its subdifferential is given by the set of optimal dual solutions to the corresponding LP instance. We observed in Chapter 1 that we can derive dual functions for the LP value function by constructing the subgradient inequalities for fixed right-hand sides. The Benders' method, however, requires lower approximating functions for the expected recourse function Ξ. The following result shows that the subgradients of the expected recourse function can be derived from the combination of the subgradients of the second-stage value function.

Proposition 3.2. *(Ruszczynski and Shapiro, 2003) Suppose the expected recourse $\Xi(x)$ is finite for at least one point $\hat{x} \in S_1$. Then $\Xi(x)$ is polyhedral and for any $x \in S_1$ and*

$$\partial\, \Xi(x) = \sum_{\omega \in \Omega} p_\omega \partial z_{LP}(h_\omega - T_\omega x).$$

The first step in specifying the Benders' algorithm is to formulate the master problem, which we introduced in Section 1.3. Recall that we re-wrote (DE) in (1.39) as

$$\min\{c^\top x + \theta \mid \theta \geq \sum_{\omega \in \Omega} p_\omega z(h_\omega - T_\omega x), x \in S_1\}.$$

To form the master problem, we replaced the value function with such an approximation to obtain

$$\min\{c^\top x + \theta \mid \theta \geq \sum_{\omega \in \Omega} p_\omega \max_{f \in \mathcal{F}_\omega} f(h_\omega - T_\omega x)\}, \tag{3.4}$$

where $\mathcal{F}_\omega$ represents the set of all dual functions associated with scenario ω that have been generated so far.

Let us provide further details on derivation of dual functions. At iteration k of the algorithm, solving each scenario subproblem yields to optimal dual solutions ν_ω^k for each

$\omega \in \Omega$. The feasible region of the dual problem of each subproblem is identical for all the subproblems and is defined by

$$\mathcal{D}_{LP} = \{\nu \in \mathbb{R}^m \mid W^T\nu \leq q\}. \tag{3.5}$$

From the finiteness of the expected recourse function for all $x \in S_1$ we have that there exist ν_ω^k, the optimal extreme point of the dual problem for the subproblem, such that

$$z_{LP}(h_\omega - T_\omega x^k) = (h_\omega - T_\omega x^k)^\top \nu_\omega^k. \tag{3.6}$$

From the convexity of z_{LP} we have

$$z_{LP}(h_\omega - T_\omega x) \geq z_{LP}(h_\omega - T_\omega x^k) - T_\omega^\top \nu_\omega^k (x - x^k) = (h_\omega - T_\omega x)^\top \nu_\omega^k. \tag{3.7}$$

Finally, from Proposition 3.2 we have

$$\Xi(x) \geq \sum_{\omega \in \Omega} p_\omega (h_\omega - T_\omega x)^\top \nu_\omega^k. \tag{3.8}$$

From the right-hand side of (3.8) we have valid dual functions for the second-stage value function and we can set $f_\omega^k = (h_\omega - T_\omega x)^\top \nu_\omega^k$ in (1.42). The resulting inequality is known as an *optimality cut*, as it provides a lower bound on the optimal solution of the subproblem. As the algorithm progresses, the collection of the optimality cuts gathered from solving subproblems in the previous iterations provide a stronger approximation of the value function for that scenario. The algorithm terminates where the approximation coincide with the value function for all subproblems at a proposed solution of the master problem.

The classical Benders' method as described does not incorporate any upper bounds on the approximation of the expected recourse, rather it is based on lower bound construction through aggregating polyhedral lower bounding functions that provide the exact solution to their respective scenario subproblems. In the case that solving the scenario

subproblems to optimality or obtaining the exact subgradients is computationally expensive, methods for approximating the bounds are desirable. One variation of the basic Benders' method is the the Benders' method with *sequential bounding approximations* first proposed by Madansky (1959). The idea of the method is to approximate lower and upper bounds on the expected recourse $Q(x)$. The method starts by partitioning Ω and calculating the expected value of the recourse conditioned to each partition. Then a lower bound on each partition is obtained that is later combined with the ones of other partitions to obtain a lower bound for the expected recourse. The lower bounds from partitions can be obtained for instance from Jensen inequalities. Similarly, an upper bound on the expected recourse can be constructed by evaluating the expected recourse for a feasible first-stage decision. The stopping criteria is a measure of the difference between these bounds. If the error not satisfactory, the method proceeds by refining the partitions to get tighter bounds. Several methods for constructing upper and lower bounds on the expected recourse are proposed by Edmundson (1957) and Madansky (1959).

Another well-known approximation based method to solve the continuous two-stage optimization problem is the *Sample Average Approximation* (SAA). This method does not depend on the convexity of the second-stage value function or the independence of the samples drawn from from Ω. Like the previous method, SAA constructs lower and upper approximations on the expected recourse. The approximations constructed are unbiased estimators of the expected recourse driven from its Monte Carlo simulation by drawing N independent samples from Ω. We denote these samples with $\omega^1, \omega^2, \ldots, \omega^N$. The expected recourse function is then approximated by

$$\Xi_N(h_{\omega^i} - T_{\omega^i}x) = \frac{1}{N}\sum_{i=1}^{N}\Xi(h_{\omega^i} - T_{\omega^i}x). \tag{3.9}$$

Combining with the first-stage problem, the resulting N problems are

$$\Psi_N^i(x) = \min\{c^\top x + \Xi_N(h_{\omega^i} - T_{\omega^i}x) \mid x \in S_1\}. \tag{SAA}$$

Mak et al. (1999) showed that if ω^i, $i = 1, \ldots, N$ are i.i.d. random variables, then

$$\frac{1}{N}\sum_{i=1}^{N}\Psi_N^i(x) \leq f^*$$

and

$$\frac{1}{N}\sum_{i=1}^{N}\Psi_N^i(x) \leq \frac{1}{N}\sum_{i=1}^{N}\Psi_{N+1}^i(x),$$

where Ψ^* denotes the solution to the original two-stage problem. This result builds the lower bounding piece of the method. It also promises improving it as we continue to add to the number of the drawn samples. To construct upper bounds, having any candidate solution $\hat{x}$, we can compute the unbiased estimator of $\Xi(\hat{x})$ from

$$c^\top x + \frac{1}{N}\sum_{i=1}^{N} z_{LP}(h_{\omega^i} - T_{\omega^i}x)),$$

which entails solving N recourse subproblem. Like the sequential bounding procedure, the candidate $\hat{x}$ can come the solution of the lower bounding problems in (SAA).

Here, we provided an overview of some of the most well-known methods to solve the continuous two-stage stochastic problems. There are several other exact or approximation-based solution methods. We refer to (Birge and Louveaux, 1997; Kall and Mayer, 2010) for a comprehensive review of these algorithms.

3.2 Solution Methods

To date, the majority of the work done on solution of two-stage stochastic linear optimization problems has been on the case of a (mixed) binary or pure integer second-stage problem. Table 3.1 provides a summary of the methods proposed to date and the assumptions required for the employment of each method. The first two sets of columns specify the assumptions made on integrality of variables, while the third set of columns describes the stochasticity in the input. Below, we briefly review the algorithms in this table, as well as other related work.

It is natural that various special cases involving binary variables have received the most attention, given the rich theory that has been developed specifically addressing this case. In the early work of Carøe and Tind (1997), the authors suggested the use of disjunctive programming and lift-and-project cuts when the recourse is mixed binary. They first decomposed the feasible set of (DE) into $|\Omega|$ subsets, which they sequentially convexified to generate the convex hull of each subset. Sherali and Fraticelli (2002) modified Benders' method by generating valid inequalities in the subproblems. Assuming the first-stage variables are binary, each such valid inequality can be generated using the reformulation linearization technique (RLT) or lift-and-project. To ensure the generated valid inequality is globally valid, it can be re-expressed as a function of the first-stage variables. Using the dual solution of the optimal subproblem, optimality cuts for the master problem can be generated. In the same vein, Sen and Higle (2005) developed valid inequalities in both stages. The valid inequalities to augment the linear relaxation of the second-stage problem were generated so as to be valid for the union of disjunctive sets obtained by a disjunction arising from a fractional second-stage variable. Ntaimo (2010) provided a variation of the latter for problems with fixed T and h. Sherali and Zhu (2006) extended the framework of Sherali and Fraticelli (2002) to accommodate binary variables in the first stage by using decomposition and global branch-and-bound methods. Sherali and Smith (2009) used the RLT to devise a specialization of Benders' algorithm for two-stage stochastic risk management problems with a pure binary first-stage problem.

For the case of a pure integer second-stage problem, Gade et al. (2012) recently used Benders' decomposition and generated valid inequalities in both stages. The proposed method solves the second-stage subproblems as LPs and iteratively adds Gomory valid inequalities to the description of the subproblems. The generated valid inequalities are parametrized as a function of the first-stage variables. Taking a different approach, Schultz et al. (1998) characterized regions of the right-hand side over which the second-stage value function is constant. This allows for a countable partition on the set S_1 using which, one can determine the corresponding level-sets of the objective function. The

candidate points from these level-sets are enumerated and evaluated using the Gröbner basis of the recourse PILP. Building on results on level sets of a pure integer value function, Ahmed et al. (2004) proposed the value function reformulation of (SP). Through a variable transformation, a global branch-and-bound algorithm was applied to the problem and was shown to be finite. This work assumed general first-stage variables and a fixed matrix T. Restricting the stochasticity to the right-hand-side vector h, Kong et al. (2006) proposed a procedure for finite construction of the value function when S_1 is finite. Furthermore, combining the results of Schultz et al. (1998) on level sets and the value function reformulation of Ahmed et al. (2004), the authors provided a characterization of certain candidate points, the so-called *minimal tenders*, in partitioning the right-hand side region. Trapp et al. (2013) proposed an alternative global branch-and-bound method that optimizes over a certain integral monoid. Finiteness of all algorithms in this category rely on pure integrality in the second-stage problem.

The first work to consider general recourse problems (those with both continuous and integer variables) was due to Laporte and Louveaux (1993). They proposed a modification of Benders' method that requires the solution of second-stage subproblems, as in the classical method, but the optimality cut used was a specialized linear cut that is valid only for problems with binary first-stage variables. Carøe and Tind (1998) pioneered the use of integer programming duality theory to develop optimality cuts for the Benders' framework, but their algorithm was designed for problems with pure integer recourse. They demonstrated how dual functions generated from a cutting-plane method with Gomory cuts can be used to generate optimality cuts. In the disjunctive decomposition branch-and-bound method of Sen and Sherali (2006), the second-stage problems were general and were solved with a branch-and-bound algorithm. The dual function obtained from each tree was modified to derive a valid inequality for the first-stage problem. Finally, a convexification technique similar to the one of Sen and Higle (2005) was used to linearize these inequalities. The disjunctive convexification technique used requires binary first-stage variables. Computational improvements to cut generation in the latter work were reported in Yuan and Sen (2009). Carøe and Schultz (1998) accommodated general

	First Stage			Second Stage			Stochasticity			
	$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{B}$	$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{B}$	**W**	**T**	**h**	**q**
Laporte and Louveaux (1993)			*	*	*	*	*	*	*	
Carøe and Tind (1997)	*		*	*		*	*	*	*	*
Carøe and Tind (1998)	*	*	*		*	*		*	*	
Carøe and Schultz (1998)	*	*	*	*	*	*		*	*	*
Schultz et al. (1998)	*				*	*			*	
Sherali and Fraticelli (2002)			*	*		*	*	*	*	*
Ahmed et al. (2004)	*	*	*		*	*	*		*	*
Sen and Higle (2005)			*	*		*		*	*	
Sen and Sherali (2006)			*	*	*	*		*	*	
Sherali and Zhu (2006)	*		*	*		*	*	*	*	
Kong et al. (2006)		*	*		*	*	*	*	*	*
Sherali and Smith (2009)			*	*		*	*	*	*	*
Yuan and Sen (2009)			*	*		*		*	*	*
Ntaimo (2010)			*	*		*	*			*
Gade et al. (2012)			*		*	*	*	*	*	*
Trapp et al. (2013)		*	*		*	*			*	
Generalized Benders' algorithm	*	*	*	*	*	*		*	*	

Table 3.1: Assumptions made in related algorithms

variables in both stages through a fundamentally different approach called *dual decomposition*, which relaxed the so-called *non-anticipativity* constraint in a Lagrangian fashion. Carøe and Tind (1998) suggested the use of dual functions as optimality cuts in a fashion similar to what we describe herein. Sen and Sherali (2006) solved the subproblems from each scenario as a generic MILP and obtained dual functions from the branch-and-bound tree, as we do, but then convexified them, which again restricts the form of the first-stage problem.

3.3 The Branch-and-Bound Representation

In Section 2.2, we proposed a characterization of the value function that we now show leads to a deterministic reformulation of (SP) that is distinct from (DE). Recall that the structure of the second-stage value function arises from the structure of two related functions:

$$z_C(d) = \min_{y_C \in \mathbb{R}^{n_2 - r_2}_+} \{q_C^\top y_C \mid W_C y_C = d\}, \tag{3.10}$$

which we called the continuous restriction, and

$$z_I(d) = \min_{y_I \in \mathbb{Z}^{r_2}_+} \{q_I^\top y_I \mid W_I y_I = d\}, \tag{3.11}$$

which we called the integer restriction. The second-stage value function (RV) can be rewritten as a combination of these two related value functions in the following way

$$z(b) = \min_{y_I \in \mathbb{Z}^{r_2}_+} \{z_I(W_I y_I) + z_C(b - W_I y_I)\} = \min_{y_I \in \mathbb{Z}^{r_2}_+} \{q_I^\top y_I + z_C(b - W_I y_I)\}, \tag{3.12}$$

We further showed that there exists a set S_{min} that is (1) finite under mild assumptions, (2) can be constructed algorithmically, and (3) is necessary and sufficient to describe z and we have the following representation of the second-stage value function

$$z(b) = \min_{y_I \in S_{min}} \{q_I^\top y_I + z_C(b - W_I y_I)\} \quad \forall b \in \mathbb{R}^{m_2}. \tag{3.13}$$

This result guarantees that points in S_{min} are solutions to the pure integer restriction, so one consequence is that, roughly speaking, we only need to consider solutions of this pure integer linear optimization problem when constructing the value function of a general MILP. Furthermore, we only need to consider a small subset of those solutions to get the full value function. The notion of strict local convexity that we utilize generalizes the notion of minimal tenders from (Trapp et al., 2013). Each such point is the minimizer over an associated region for which the value function is convex. Over such regions, the integer part of the optimal solution remains constant and the value function of the MILP is simply a translation of the value function of the continuous restriction to that point. These regions are a generalization of the regions described by Schultz et al. (1998) over which the value function remains constant in the pure integer case. Figure **??**, which appears later in this chapter, shows these regions for the second-stage problem from Example 1.6.

A Value-function Reformulation

In Section 2.5, we described an algorithm for generating a superset of S_{min} that can be implemented in practice. This result gives us a way of generating a complete description of z that can be embedded in (SP) to obtain the aforementioned deterministic reformulation. Note that in this context, we are only really interested in parts of the domain that can actually arise from first-stage solutions in some scenario, e.g., we only need to know the value function over the set B_2, defined in the beginning of this chapter, which is bounded by our assumptions. We do not depend on the finiteness of S_{min}, but on that of the set

$$\mathcal{J} = S_{min} \cap \text{proj}_{\mathbb{Z}^{r_2}} \left(\cup_{b \in B_2} S_2(b) \right). \tag{3.14}$$

To derive the reformulation, we first rewrite (SP) as

$$\min_{x \in S_1} c^\top x + \sum_{\omega \in \Omega} p_\omega \left(\min_{y_I \in \mathcal{J}} \left\{ q_I^\top y_I + z_C (h_\omega - T_\omega x - W_I y_I) \right\} \right). \tag{3.15}$$

using Theorem 2.7. From assumption A1, we have that the dual polyhedron of the linear relaxation of the recourse problem is always nonempty. In principle, one can generate the set of extreme points of a bounded polyhedron via a vertex enumeration method, such as the method of Avis and Fukuda (1992). The same method can be used to obtain extreme rays by bounding the polyhedron artificially. Let $\{\nu^i\}_{i \in K}$ be the set of extreme points of this polyhedron indexed by set K and $\{\sigma^j\}_{j \in L}$ be its set of extreme directions indexed by set L. Then, we can rewrite (3.15) as a deterministic MILP. We let the set $\mathcal{J}$ be indexed

by set J.

$$\min\ c^\top x + \sum_{\omega\in\Omega} p_\omega \gamma(x,\omega)$$

$$\text{s.t. } \gamma(x,\omega) \geq q_I^\top y_I^j + \zeta(x,\omega) - M^{j,\omega}(1-u^{j,\omega}) \qquad \forall j\in J, \omega\in\Omega \qquad (3.16)$$

$$\zeta(x,\omega) \geq (h_\omega - T_\omega x - W_I y_I^j)^\top \nu^i \qquad \forall i\in K, j\in J, \omega\in\Omega \qquad (3.17)$$

$$0 \leq (h_\omega - T_\omega x - W_I y_I^j)^\top \sigma^i \qquad \forall i\in L, j\in J, \omega\in\Omega \qquad (3.18)$$

$$\sum_{j\in J} u^{j,\omega} = 1 \qquad \forall \omega\in\Omega \qquad (3.19)$$

$$x\in S_1 \qquad (3.20)$$

$$u^{j,\omega}\in\mathbb{B}. \qquad \forall j\in J, \omega\in\Omega \qquad (3.21)$$

where $M^{j,\omega} \geq \max_{x\in S_1}\{|(h_\omega - T_\omega x - W_I y_I^j)^\top \nu^i|\}$ for all $\omega\in\Omega$, $i\in K$ and $j\in J$. Here, $\gamma(x,\omega)$ represents the cost of the recourse for a given scenario and first-stage solution. In the first constraint, this variable is used to represent the value function of the second-stage problem as described in (3.13), by minimizing over the set $\mathcal{J}$. This constraint, together with (3.19), guarantees that equality holds for at least one of the constraints associated with $y_I\in\mathcal{J}$ and holds for that which z achieves the minimum. The variable $\zeta(x,\omega)$ is the cost of the continuous restriction problem with respect to the fixed integer vector y_I, i.e., $\zeta(x,\omega) = z_C(h_\omega - T_\omega x - W_I y_I)$. (3.17) and (3.18) represent z_C in terms of the extreme points and rays of the associated dual problem.

Generating the formulation (3.16) requires generation of $\mathcal{J}$, K, and L a priori. Although these operations would clearly be expensive, they are nevertheless finite under our assumptions. Therefore, this reformulation yields a finite algorithm that can be seen as an alternative to solution of the deterministic equivalent. Although a complete description of the value function is embedded within (3.16), its size is independent of the size of the formulation of the second-stage problem and depends only on the number of scenarios and the cardinalities of $\mathcal{J}$, K and L. This reformulation could be smaller than (DE) in cases where z has a simple structure and the cardinalities of K and L are relatively

small. In practice, however, these sets can be large and solving (3.16) directly would be cumbersome in many cases. An obvious idea for overcoming this complexity, which is the basis for our method, would be to generate parts of the formulation dynamically. We describe how to do that in the next section.

Warm Starting the Approximations

As a first step toward a complete specification of the generalization of Benders' method, we now describe the details of the most critical component, a procedure for dynamically constructing lower approximations of the value function from information generated by solution of instances of the second-stage problem. We describe in Section 3.4 how the procedure can be embedded within Benders' framework and tightly coupled with the procedure for solving the master problem. In Section 2.6, we discussed that dual functions are lower approximating functions we need and they can be most practically generated as a by-product of particular solution algorithms. We discussed a number of different primal solution frameworks that can result in strong dual functions. The two primary such frameworks are the cutting-plane method and the branch-and-bound method. We discussed that the cuts that can be used in the cutting-plane method should have a parametric representation as a function of their right-hand sided. This limits the families of cuts that can be used. Furthermore, the cutting-plane method is well-known to suffer from numerical instability. Dual functions arising from branch and bound (and also potentially from branch and cut) appear to be more practical. We derived branch-and-bound dual functions in (2.57) and demonstrated that the dual function obtained from solving an instance of the second-stage problem is strong for that instance. Strong dual functions are important in the context of stochastic optimization not only because they provide proofs of optimality, but also they provide a natural way of performing sensitivity analyses, since such a function provides provable bounds on the optimal solution value for modified instances. Moreover, they provide methods of warm-starting the solution process, since the function produced when evaluating $z(\hat{b})$ for some $\hat{b} \in \mathbb{R}^{m_2}$ is likely similar or even identical to that produced when evaluating z in a close neighborhood of

$\hat{b}$.

Let us once more consider the branch-and-bound dual function (2.57).

The interpretation of the function $\underline{z}$ above is conceptually straightforward. The solution to the LP relaxation (P^t) of node t in the branch-and-bound tree yields the standard LP dual function, which bounds the optimal value of that subproblem. The overall lower bound yielded by the tree is the smallest bound yielded by any of the leaf nodes. This is the usual lower bound yielded by a branch-and-bound-based MILP solver during the solution process. By interpreting the optimal solution to the dual of the LP relaxation in each node as a *function*, we obtain $\underline{z}$.

Theorem 2.12 provides a recipe for producing a tree that corresponds to a strong dual function for a given $b \in \mathbb{R}^{m_2}$. This is already enough to allow us to state a convergent generalization of Benders' method in which we produce a different such function each time we solve the second-stage problem. In essence, this would mean constructing a new dual function for each scenario in each iteration of the algorithm. This approach would work in principle, but in practice, there may be substantial overlap between the dual functions produced and the resulting master problem may be much bigger than necessary. Furthermore, starting each subproblem solve from scratch will result in many repeated computations. We can improve on this basic framework by incorporating *warm-starting* techniques.

The goal of warm starting procedures in integer linear optimization to accelerate solution of new problem instances by utilizing information obtained from solution of a base instance. The process is initialized by collecting the optimal bases of the current tree's terminating nodes, as described in the proof of Theorem 2.12. These previous optimal bases are used to warm-start solution of the LP relaxations in each leaf node, producing a set of candidate terminating nodes that require further branching (due to the introduction of fractional value for some of the integer variables). After the initial phase, branching is continued as usual until a tree that is optimal with respect to the new right-hand side is found. We now illustrate the concept of warm-starting the solution process by showing how we would solve a sequence of subproblems within Benders' method,

warm starting the solution of each subproblem from the tree generated by solution of the previous subproblem.

Example 3.1. Although we have not formally stated our generalization of Benders' method yet, it should be clear that the method consists of the iterative solution of a sequence of subproblems with different right-hand sides. The goal of each step in this process is to improve the approximation of the value function in an area where it is currently not strong in response to the proposed first-stage solution, generated with respect to that current approximation.

Consider now the second-stage problem from Example 1.6. In Figure 3.1– 3.4, we show the steps in solving this second-stage problem for four right-hand sides, $5.5, 11.5, 4$ and 10, arising from execution of Benders' method. Consider first the value function of the LP relaxation of the original problem. Over the interval $(0, 12)$, this function is $g_0 = 0.8b$ (see Figure 3.1) and is strong at right-hand sides $0, 5$ and 10. For these right-hand sides, the solution to the LP relaxation is integer-valued and the branch-and-bound tree would consist of a single node.

The function g_0 lies strictly below the value function at $b = 5.5$, which indicates that for this right-hand side, one of the integer variables (y_2) takes on a fractional value. We therefore branch on y_2 to produce two affine functions, g_1 and g_2. The resulting dual function, $\min\{g_1, g_2\}$, is a piecewise concave dual function and is strong at $b = 5.5$.

In evaluating z at 11.5 in Figure 3.2, we begin with the optimal tree from the previous iteration. Re-optimizing the LPs at nodes 1 and node 2 with the new right-hand side, we obtain a new dual solution for node 2, while the function of node 1 is already optimal for $b = 11.5$ and the primal solution at node 1 is still integer feasible. Node 2 produces a primal solution in which y_2 is fractional and we thus have to branch on y_2 *again*. Note that this situation cannot arise in branch-and-bound for a single fixed instance, since the value of the variable that was just branched on must be integer in the resulting subproblems. After branching, we obtain two new affine functions.

For $b = 4$, we similarly re-optimize the LP relaxations at nodes $1, 3$ and 4, branch in response to the fractional solution that is now produced in node 1 and produce affine

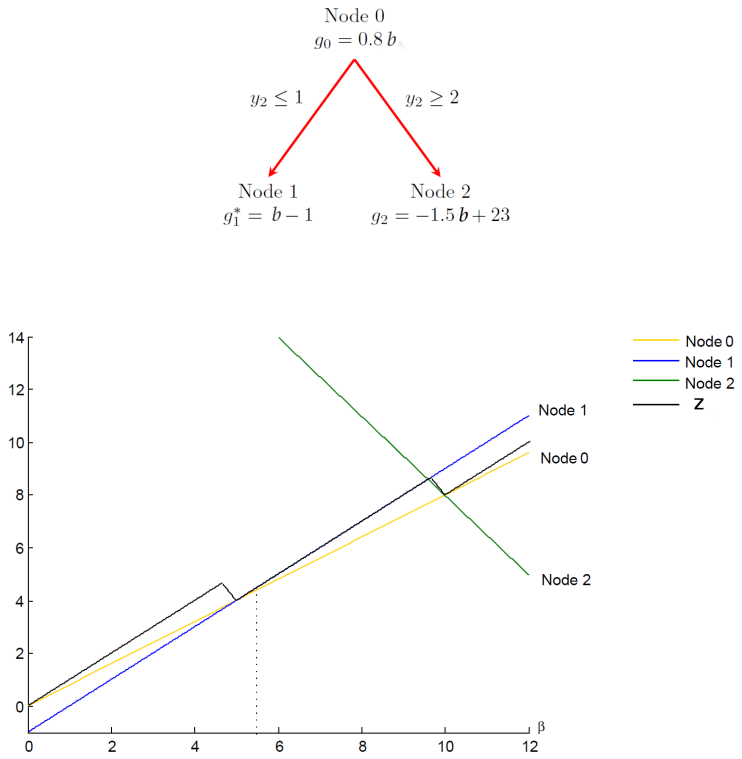
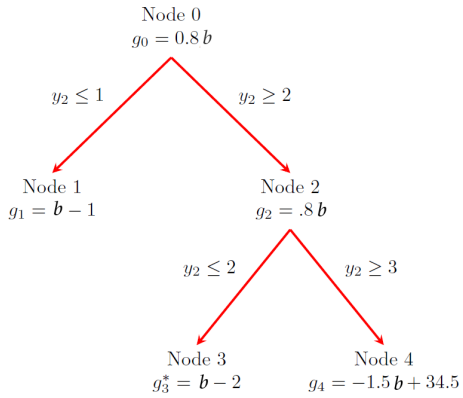


Figure 3.1: Strong dual functions from warm-starting branch-and-bound - RHS = 5.5

functions g_5 and g_6. The function arising from node 5 coincides with the value function on the interval $(0, \frac{14}{3})$. In Figure 3.4 we can observe that the same tree is optimal for $b = 10$ and the optimal solution is obtained by re-optimizing the leaf nodes. □

An important observation from Example 3.1 is that in warm-starting the computation using a given tree, the function arising from the procedure of Theorem 2.12 may change, although no further branching has been performed (this happened in the last iteration of the above example). This results from the re-optimization of the LP relaxation in each leaf node, which could yield a different solution to the continuous restriction. With respect to this new solution, it could be the case that variables whose values were previously integer become fractional (this can happen for integer variables whose values are not completely fixed by branching yet), requiring further branching. However, if we were to discard previously generated dual solutions in this case, important information would be lost, as we will see, and the function in one iteration might no longer be strong for right-hand



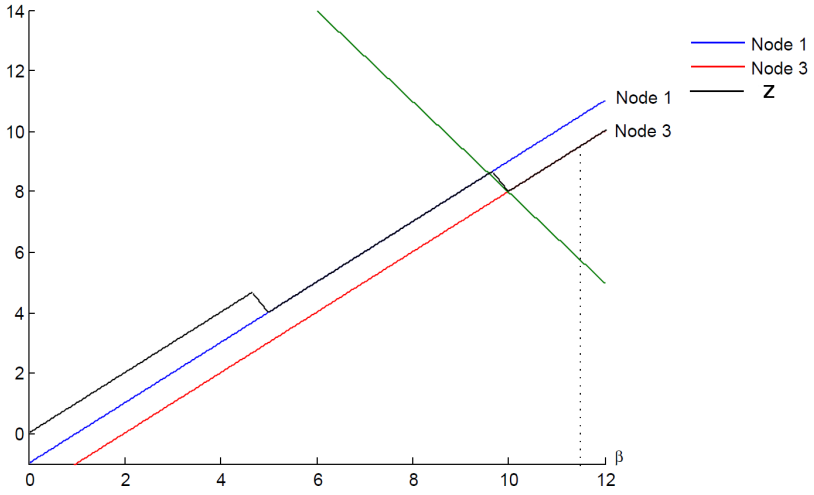
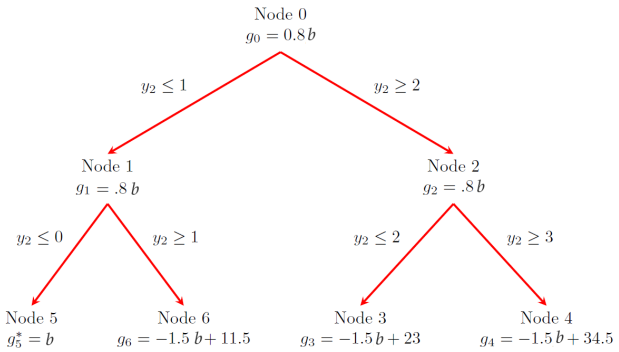


Figure 3.2: Strong dual functions from warm-starting branch-and-bound - RHS = 11.5



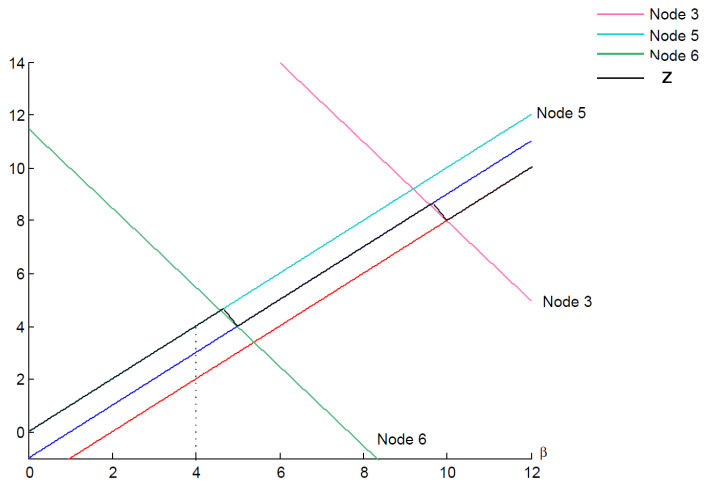
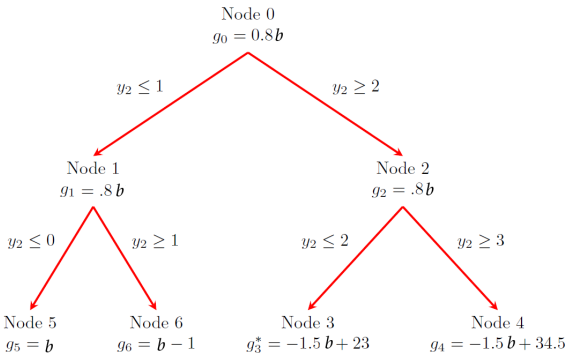


Figure 3.3: Strong dual functions from warm-starting branch-and-bound - RHS = 4



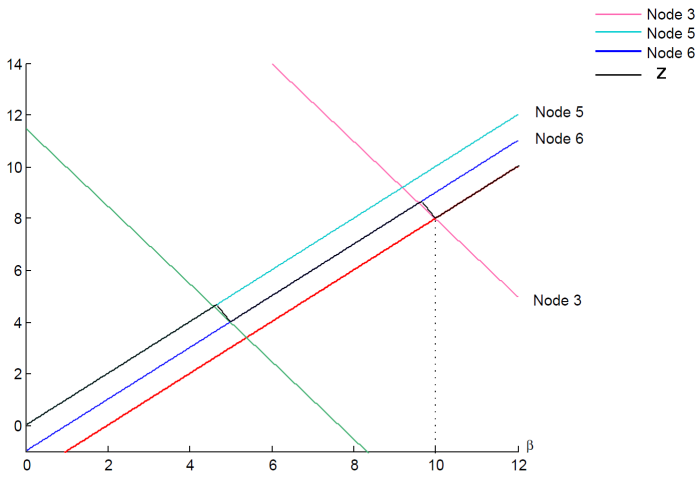


Figure 3.4: Strong dual functions from warm-starting branch-and-bound - RHS = 10

sides from previous iterations. What is needed is a continuously refined dual function that remains strong for all previously arising right-hand side values and is made strong for a new right-hand side in a given iteration.

To maintain a single dual function that is strong for all previously occurring right-hand sides within a single tree, we must maintain a collection of all the linear functions that provide lower bounds for any given node, including all functions generated in previous iterations in either that node or any of its ancestors. Taking the maximum of all linear functions in this collection yields a convex lower approximation of the value function of the subproblem associated with that node (which is comprised of the original MILP plus some branching constraints). This function can be interpreted as a convex lower approximation of the value function of the LP relaxation of the given node (which is itself the best convex lower approximation of the value function of that subproblem). We can interpret the process of collecting these linear functions associated with each node as a process of building a convex approximation of its value function.

When branching occurs, the collection of linear functions of a given node must be inherited by *both* children. Branching can be interpreted as a process of identifying a region in which a convex lower approximation is not strong enough and then branching in order to divide the region into multiple subregions, each of which has its own convex bounding function. Through this process, we progressively improve the strength of the overall function. It should be clear that when doing the partitioning, we need to begin with the description of the original convex function in both of the child nodes and proceed from there. Naturally, in practice, many of the functions generated at higher levels of the tree will eventually be redundant and can be discarded.

To formalize this, let us consider the strongest bound that could be obtained by considering a given branch-and-bound tree if we are allowed to re-optimize the LP relaxations in each of the leaf nodes when evaluating the function with respect to a new right-hand side. Allowing the optimal solution to the LP relaxation to be changed without changing the tree yields a dual function stronger than the one described in Theorem 2.12 that is obtained by taking the minimum over the *full value functions* of the LP relaxations of

each of the nodes in the tree. We define this strengthened function as

$$\underline{z}^*(b) = \min_{t \in T} z^t(b). \tag{3.22}$$

Going a step further, we note that if some of the branching bounds have served to fix the values of certain integer variables, we can explicitly indicate that these variables can be treated as constants to yield another equivalent definition.

$$\underline{z}^*(b) = \min_{t \in T} q_{I_t}^\top y_{I_t}^t + z_{N \setminus I_t}(b - W_{I_t} y_{I_t}^t), \tag{3.23}$$

where I_t is the set of indices of fixed variables, $y_{I_t}^t$ are the values of the corresponding variables in node t, and $z_{N \setminus I_t}$ is the value function of the linear program including only the unfixed variables.

With this strengthening, we obtain a function from a single tree that is the minimum over a collection of convex functions, just as the value function itself is. In fact, when the branching process is carried to its logical extreme, the above strengthened procedure will eventually yield the full value function. The fact that we can obtain the full value function from a single tree can be observed from the strong connection between the function (3.23) and the representation of the full value function in (3.13). If our branching decisions eventually lead to the fixing of variables in such a way that each member of S_{min} is represented among the fixed portions of solutions in some set of leaf node, this would be enough to ensure that the tree would in fact represent the entire value function. This can be stated formally as the following result.

Theorem 3.2. *Under the assumption that* $\{b \in \mathbb{R}^{m_2} \mid z_I(b) < \infty\}$ *is finite, there exists a branch-and-bound tree with respect to which* $\underline{z}^* = z$.

Proof. To construct such a tree, we need to impose branching decisions that guarantee that $S_{min} \subseteq \{y_{I_t} \mid t \in T, |I_t| = r_2\}$. Such a set of branching decisions can be easily constructed. □

Returning to the second-stage problem in Example 1.6, we can see graphically that the

members of the set S_{min} from (3.13) that lie in the interval $(-10, 10)$ is

$$\{(0,0,5),(0,0,4),(0,0,3),(0,0,2),(0,0,1),(0,0,0),(0,1,0),(0,2,0)\}. \tag{3.24}$$

Therefore, over the domain $(-10, 10)$, we have

$$\begin{aligned} z(b) = \min\ \{&15 + z_C(b+10), 12 + z_C(b+8), 9 + z_C(b+6), 6 + z_C(b+4), \\ &3 + z_C(b+2), z_C(b), 4 + z_C(b-5), 8 + v_C(b-10)\}. \end{aligned} \tag{3.25}$$

To obtain this function from a given tree, we must ensure that each member of the above set produced by solutions of the LP relaxation of some leaf node in the tree *over the entire interval for which the piece coincides with the value function.* Figure 3.5 illustrates this principle by showing how to obtain the value function of the second-stage problem in Example 1.6 from a single branch-and-bound tree over the given interval. This is a minimal such tree—each leaf node corresponds to one affine piece of the value function. That is, the leaf nodes correspond exactly to the members of S_{min} from (3.13). In general, the described procedure may result in a larger tree that contains extra affine functions in its nodes—these can be safely discarded. Figure 3.6 also shows the stability sets corresponding to each node.

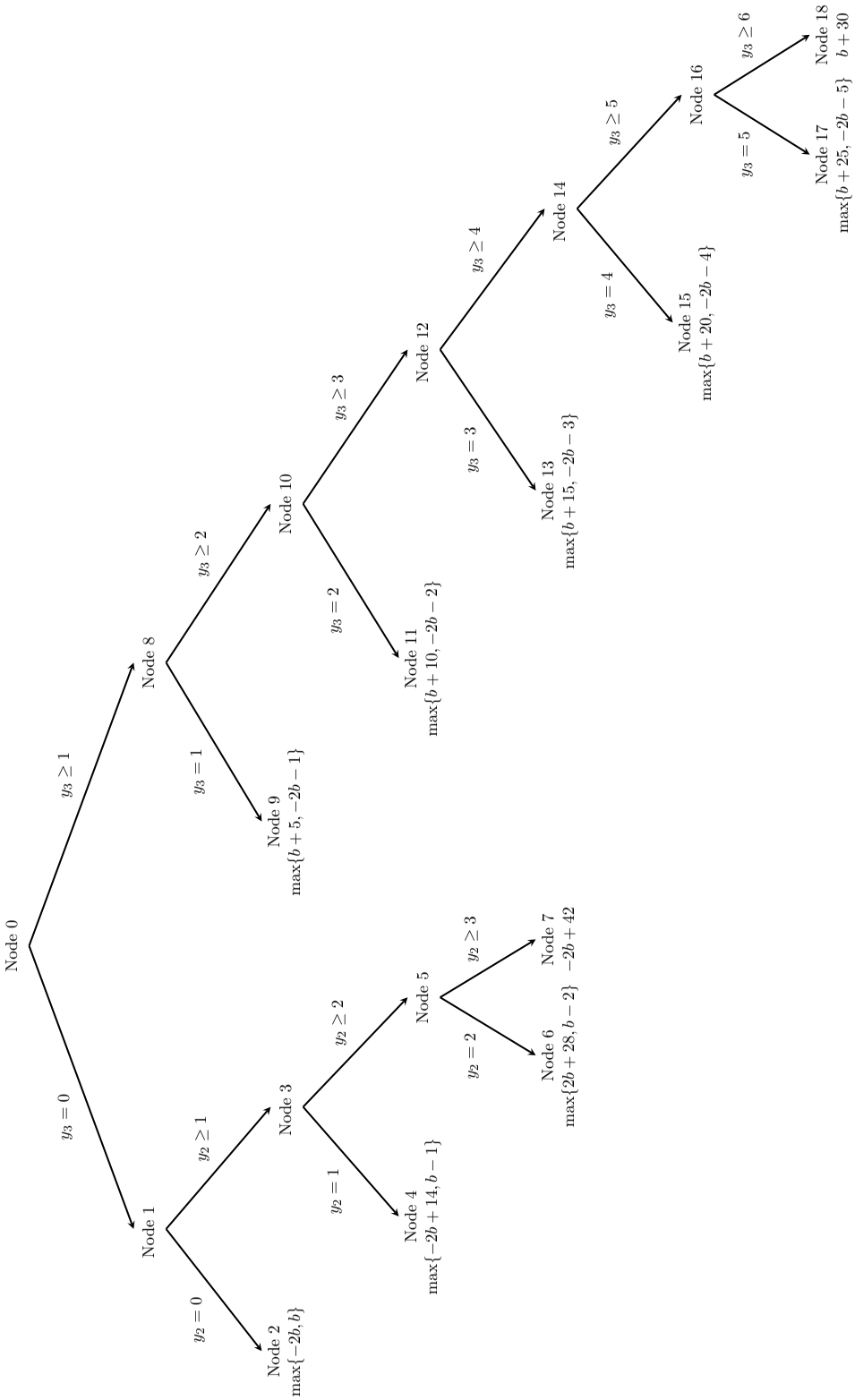


Figure 3.5: Branch-and-bound tree and value function correspondence for Example 1.6

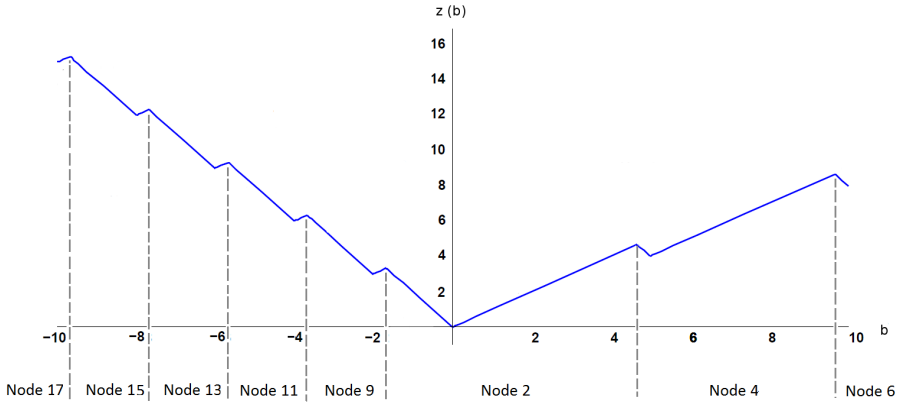


Figure 3.6: The MILP value function corresponding to the tree in Figure 3.5

3.4 The Generalized Benders' Algorithm

We now come to the formal statement of our generalization of Benders' method. At a high level, the approach consists of solving the value function reformulation (3.16), generating relevant parts of the formulation dynamically, as in a traditional cutting plane method. Viewed in terms of the dual functions introduced in the previous section, the dynamic generation consists of further refinements of the branch-and-bound tree in which we solve instances of the second-stage and which we use to derive our current value function approximation.

The version of Benders' method we are proposing here differs from a classical implementation, such as the one first suggested by Carøe and Tind (1998) in this context, in that a straightforward interpretation of Benders' method would lead to generation of a new strong dual function in each iteration for each scenario, while we are simply refining the previously constructed dual function using the method described in the previous section. From a theoretical standpoint, these interpretations amount to the same thing. In terms of implementational details, however, the difference is quite significant. To be consistent with the traditional interpretation, we initially describe the method as generating a different dual function in each iteration before being more specific about the details of

our implementation.

The first step in specifying the algorithm is to formulate the master problem. The master problem takes the form of the one of the classical Benders' method (1.42). Recall that earlier we established that f_ω^k represents a strong dual function produced for each scenario with respect to a proposed first-stage solution $x^k \in S_1$, the solution to the master problem in the previous iteration. In what follows, we denote the set of dual functions f_ω^i generated for the scenario ω in iterations $i = 1, \ldots, k$ by $\mathcal{F}_\omega^k$.

Generic Version of Benders' Algorithm

Step 0. Initialize

a) Initialize the dual function lists $\mathcal{F}_\omega^0 := \emptyset$ for all $\omega \in \Omega$.

b) Let $x^1 := \text{argmin}\{c^\top x : x \in S_1\}$, $\theta^1 := -\infty$ and $k := 1$.

Step 1. Update the lower approximation function

a) For each $\omega \in \Omega$, solve the subproblem (RV) for the right-hand side $h_\omega - T_\omega x^k$ and construct the function f_ω^k dual to z and strong at $h_\omega - T_\omega x^k$.

b) Check whether the current approximation is exact for x^k. Stop if $\theta^k = \sum_{\omega\in\Omega} p_\omega z(h_\omega - T_\omega x^k)$. $x^* := x^k$ is an optimal solution.

Step 2. Solve the master problem

a) Update the dual functions list: $\mathcal{F}_\omega^k := \mathcal{F}_\omega^{k-1} \cup \{f_\omega^k\}$.

b) Solve (1.42) to obtain optimal solution (x^{k+1}, θ^{k+1}) to an optimal solution of it.

c) Set $k := k + 1$. Go to Step 1.

As described in the previous section, our implementation of this generic algorithm utilizes a single global dual function that is a slight variant of (3.23), rather than individual dual functions for each scenario. For efficiency, we do not explicitly derive a

complete description of the value function of each LP relaxation, but generate pieces of it dynamically as part of the overall algorithm. This dual function consists of the minimum of a collection of piecewise polyhedral convex functions (one for each leaf node in the branch-and-bound tree). This makes the master problem a nonlinear optimization problem involving this single piecewise polyhedral function for which we do not have explicit descriptions of the polyhedral regions over which it is affine. Nevertheless, it is possible to reformulate (1.42) as a standard MILP similar to (3.16) by introducing auxiliary variables and constraints.

Let T be the set of leaf nodes of the current tree. Recall that corresponding to each leaf node, we have a set of affine dual functions that have been obtained in the leaf node and its ancestors as we refined the tree in solving previous subproblems. Let $\mathcal{I}(t)$ denote the set of indices of these affine functions at a leaf node $t \in T$. The dual function obtained from the current tree is

$$\underline{z}(b) = \min_{t \in T} \max_{i \in \mathcal{I}(t)} b^\top \eta^i + \alpha^i. \tag{DF}$$

As explained earlier, η and α can be obtained from (2.58) and (2.60). We can then write (1.42) as the following MILP

$$\begin{aligned}
\Gamma^k = \min\ & c^\top x + \sum_{\omega \in \Omega} p_\omega \underline{z}_{\ \omega} & & \\
\text{s.t. } & \underline{z}_{\ \omega} \leq q_{t,\omega} & & \forall t \in T, \omega \in \Omega \\
& \underline{z}_{\ \omega} \geq q_{t,\omega} - M_\omega (1 - u_{t,\omega}) & & \forall t \in T, \omega \in \Omega \\
& q_{t,\omega} \geq (h_\omega - T_\omega x)^\top \eta^i + \alpha^i & \quad & \forall i \in \mathcal{I}(t), t \in T, \omega \in \Omega \\
& \sum_{t \in T} u_{t,\omega} = 1 & & \forall \omega \in \Omega \\
& u_{t,\omega} \in \mathbb{B}, & & \forall t \in T, \omega \in \Omega \\
& x \in X. & &
\end{aligned} \tag{MP}$$

In (MP), $q_{t,\omega}$ represents the piecewise convex function obtained at a leaf node of the tree. That is, the fourth constraint ensures that $q_{t,\omega} = \max_{i \in \mathcal{I}(t)} b^\top \eta^i + \alpha^i$. $\underline{z}_{\ \omega}$ is the approximation of recourse for scenario ω. The second constraint guarantees that $\underline{z}_{\ \omega}$

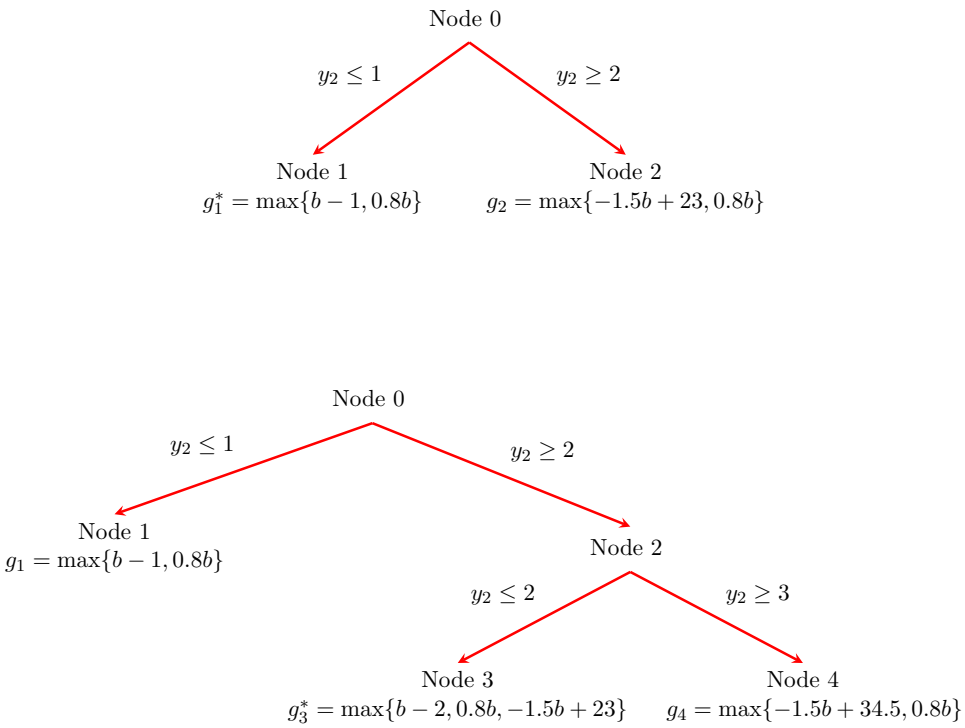


Figure 3.7: Strengthened dual functions from warm-starting (a)

is less than all the convex approximations obtained at the leaf nodes, while the third and fifth constraints guarantee that one of them holds at equality and $\underline{z}_{\omega}$ achieves the minimum of the leaf nodes. In the third constraint, M_ω is an appropriately large positive number. For instance, $M_\omega \geq \max_{t \in T} \mid q_{t,\omega}|$. Before formally proving correctness and finiteness, we illustrate the algorithm with an example where we show the details of this approach.

Example 3.2. We consider a modified version of Example 1.6 for which $S_1 = \{x_1, x_2 \in \mathbb{B} \mid x_1 + x_2 \leq 1\}$. The sequence of subproblems solved here is the same as the sequence from Example 3.1, but we now show in Figures 3.7 and 3.8 the full details of the single tree approach, including retention of all linear pieces to ensure a dual that is strong for all evaluated right-hand sides.

The starting point $(x_1^1, x_2^1) = (0, 1)$ in Step 0 is obtained through solving

$$\Gamma^1 = \min\{-3x_1 - 3.8x_2 \mid x_1 + x_2 \leq 1, x_1, x_2 \in \mathbb{B}\}.$$

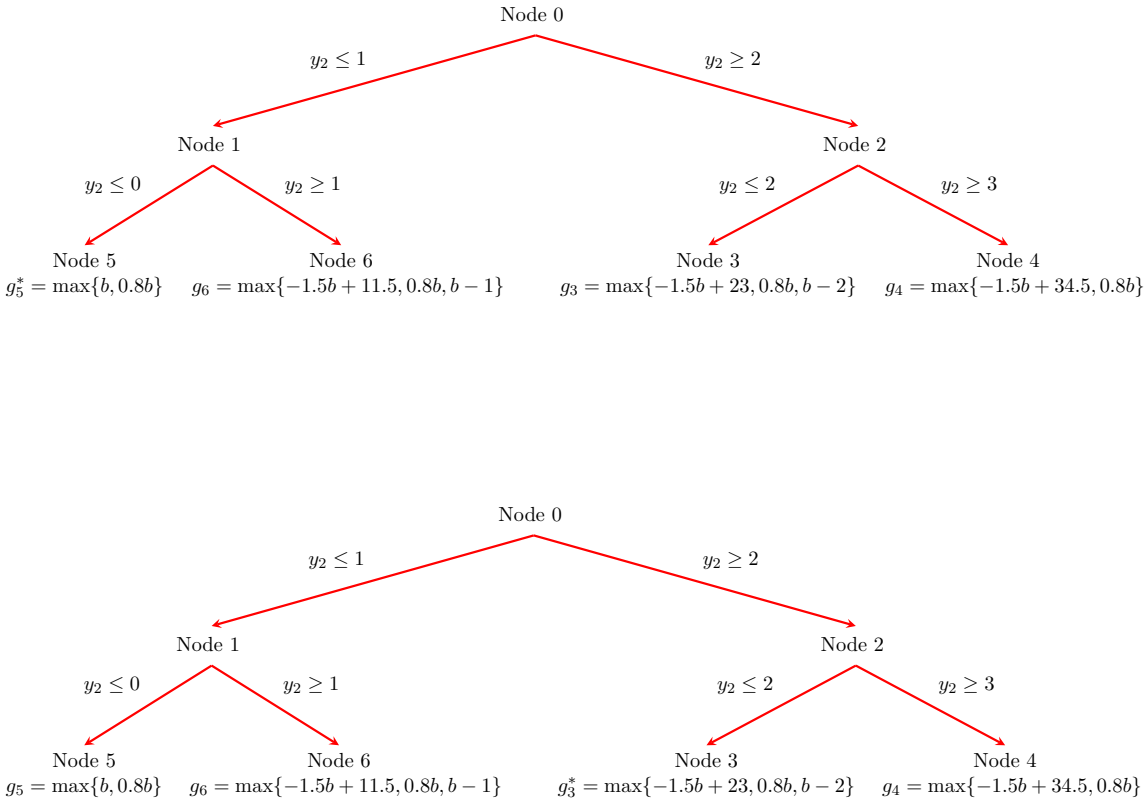


Figure 3.8: Strengthened dual functions from warm-starting (b)

Let $\theta^1 = -\infty$ and $k = 1$.

Iteration 1. In Step 1, we solve the second-stage problem for $6 - 0.5$ and $12 - 0.5$ with branch and bound to obtain the dual function

$$\underline{z}^1(b) = \min\{\max\{b-1, 0.8b\}, \max\{b-2, 0.8b, -1.5b+23\}, \max\{-1.5b+34.5, 0.8b\}\}.$$

Note that we retain the linear piece from solving the root node in each of the children, which yields a stronger dual function overall, though this piece is not required for strength at either of the two right-hand sides in the first iteration. The updated master problem is

$$\begin{aligned} \Gamma^2 = \ & \min -3x_1 - 3.8x_2 + 0.5\underline{z}^1(6 - 2x_1 - 0.5x_2) + 0.5\underline{z}^1(12 - 2x_1 - 0.5x_2) \\ & \text{s.t. } x_1 + x_2 \leq 1, \ x_1, x_2 \in \mathbb{B}. \end{aligned} \tag{3.26}$$

This would normally be formulated as in (MP) and solved as a generic MILP in practice, but for clarity and compactness, we avoid this step here. The solution is $\Gamma^2 = 2.6$ with $\theta^1 = 5.6$ and $(x_1^2, x_2^2) = (1, 0)$. Let $k = 2$.

Iteration 2. We solve the subproblem with the right-hand sides 4 (corresponding to the first scenario) and 10 (corresponding to the second scenario) to obtain our second dual function

$$\begin{aligned} \underline{z}^2(b) = \min\{ & \max\{b, 0.8b\}, \\ & \max\{-1.5b+11.5, 0.8b, b-1\}, \\ & \max\{-1.5b+23, 0.8b, b-2\}, \\ & \max\{-1.5b+34.5, 0.8b\}\} \end{aligned} \tag{3.27}$$

Since $5.6 < 0.5(4+8)$, we continue. In Step 2, we solve the updated master problem

$$\begin{aligned} \Gamma^3 = \ & \min -3x_1 - 3.8x_2 + 0.5\underline{z}^2(6 - 2x_1 - 0.5x_2) + 0.5\underline{z}^2(12 - 2x_1 - 0.5x_2) \\ & \text{s.t. } x_1 + x_2 \leq 1, \ x_1, x_2 \in \mathbb{B} \end{aligned} \tag{3.28}$$

to obtain $\Gamma^3 = 3$ with $\theta^3 = 6$ and $(x_1^3, x_2^3) = (1, 0)$. We let $k = 3$.

Iteration 3. The solution x^3 is identical to x^2. Since $\theta^3 = 0.5(\underline{z}^2(4) + \underline{z}^2(10))$, we stop. $(x_1^*, x_2^*) = (1, 0)$ is an optimal solution.

□

Having now demonstrated the algorithm, we proceed to formally state the following result.

Theorem 3.3. *Consider the generalization of Benders' algorithm in which*

- *dual functions of the form* (DF) *are generated within a single branch-and-bound tree and*
- *a master problem of the form* (MP) *is utilized.*

This algorithm terminates with the correct global minimum in finitely many steps.

Proof. We first show that if the termination check is satisfied, the algorithm terminates with a correct optimal solution. Assume the algorithm terminates in iteration k with the dual function obtained from tree T^{k-1} and let x^* be the optimal solution to the problem. Then, we have

$$\begin{aligned}
&c^\top x^k + \sum_{\omega \in \Omega} p_\omega \min_{t \in T^{k-1}} \max_{i \in \mathcal{I}(t)} \{(h_\omega - T_\omega x^k)^\top \eta^i + \alpha^i\} \\
&= c^\top x^k + \sum_{\omega \in \Omega} p_\omega z(h_\omega - T_\omega x^k) \\
&\leq c^\top x^* + \min_{t \in T^{k-1}} \max_{i \in \mathcal{I}(t)} \{(h_\omega - T_\omega x^*)^\top \eta^i + \alpha^i\} \\
&\leq c^\top x^* + \sum_{\omega \in \Omega} p_\omega z(h_\omega - T_\omega x^*) \\
&\leq c^\top x^k + \sum_{\omega \in \Omega} p_\omega z(h_\omega - T_\omega x^k).
\end{aligned}$$

where the first line is from the termination condition, the second line follows from the optimality of x^k for the master problem in iteration k, the third line follows since (DF) is a dual function for the value function of the recourse and the last line holds since x^* is the optimal solution to the problem.

To show the finiteness of the algorithm, note that if any first-stage solution of the master problem repeats in a future iteration, the termination criterion is satisfied and we have an optimal solution. This holds because the dual function obtained from an optimal tree is strong with respect to all previous right-hand sides. Therefore, if the same first-stage solution repeats, the dual function will already be strong in all scenarios and the termination check will then be satisfied. Furthermore, observe that in each iteration, we must either generate at least one new linear dual function within the branch-and-bound tree at some node or branch on some node. From Assumption A2, the set $\mathcal{J}$ defined in (3.14) is finite. There are a finite number of branching operations that can be done before achieving a complete description of this set. Assumption A1 also implies that there is a finite number of extreme points and rays of the dual polyhedron $\mathcal{D}$ of the recourse problem. Together, we have a finite number of operations that can occur during this process and a finite number of dual functions that can arise. In a finite number of steps, we therefore generate the full formulation (3.16), at which point the tree must contain a full description of the value function over B_2 and the process must terminate. □

Chapter 4

Computational Results

Studying how the optimal solution value of an optimization problem changes in response to perturbations in the problem data is important both in theory and practice. In practice, we are often interested in identifying solutions that are optimal under changes in problem data. This analysis is important because we rarely solve optimization problems that model a truly deterministic system—the problem data usually involves uncertainty. In theory, analyzing the optimal solution is important in several cases. For example, where solving a family of the problems whose structures are closely related where the optimal solution to one problem may also be optimal or provide a bound for another problem. In this case, identifying the ranges of problem data that result in the same optimal solution is desirable. *Post-optimality analysis* addresses these questions through *sensitivity analysis* and *warm-start* procedures. Sensitivity analysis procedures check whether the optimal solution to a given instance of an optimization problem remains valid under change in the problem data. They also identify the ranges of the problem data that result in the same optimal solution. Warm-starting procedures, on the other hand, are used to accelerate the solution of a new problem instance by using the information collected from the solution process of a given base instance.

In the case of LPs, verifying the optimality of a solution for a modified problem can be done by checking the optimality conditions, whose information is available, for example, from the optimal simplex tableaux used to solve the original LP. In the case that the

solution is not optimal for the new problem, this simplex tableaux can be modified to carry out further simplex iterations.

In the case of MILPs however, checking the optimality conditions is more challenging. Given that the most common algorithms such as branch-and-bound and branch-and-cut are based on constructing an exponential number of disjunction on the feasible region of some relaxation of a MILP, checking the optimality of the solution obtained from the execution of these algorithms is computationally intensive. Furthermore, the amount of information to be saved to perform warm-starting procedures can be massive. Despite such challenges, warm-starting techniques for MILPs exist and are used in practice. In particular, like the case of the LP, where a simplex tableau can be reused to derive the optimal solution for a new LP instance, in the case of MILPs we can also reuse a branch-and-bound tree to find the solution to a new MILP instance (Wolsey, 1981a).

Two-stage stochastic optimization problems are perfect candidates for utilizing post-optimality analysis techniques, as decomposing these problems yields subproblems that differ only in problem data. In the case of continuous two-stage optimization problems, these techniques have been used in finding solution methods for these problems. Examples of such studies are (Morton, 1996; Ruszczyński and Świtanowski, 1997; Zhao, 2001).

For two-stage mixed integer optimization problems, however, incorporation of warm-starting techniques within algorithms for two-stage mixed integer optimization problems is not yet explored. In this chapter, we provide details on implementing the Generalized Benders' algorithm of Chapter 3 with warm-starting capabilities for the two-stage mixed integer optimization problem, introduce different warm-starting techniques to be utilized within the algorithm and report the results of our experiments. We start the chapter by a brief overview on results on sensitivity analysis and warm-starting for MILPs. In our review, we pay a particular attention to the implementation of these results within the branch-and-bound algorithm as the primary method we use in the Generalized Benders' algorithm to solve the scenario subproblems. For further applications of post-optimality procedures in other algorithms for MILPs, we refer to the works of Roodman (1972, 1974); Piper and Zoltners (1976); Shapiro (1977); Loukakis and Muhlemann (1984).

4.1 MILP Sensitivity Analysis and Warm Starting

In this section we review the results on sensitivity analysis and warm-starting for MILPs. The forthcoming results are mainly due to Marsten and Morin (1977); Geoffrion and Nauss (1977); Nauss (1979); Wolsey (1981a). Güzelsoy (2009) provides a survey on the post-optimality methods existing for MILPs. Here we provide the key results. We discuss how to perform sensitivity analysis and warm-starting when some of the problem data is modified. Because they are the most closely related to solving two-stage optimization problems with the Generalized Benders' method, we are particularly interested in those results where the modifications are made to the right-hand side. We consider the primal problem (MILP) under modifications mainly in the right-hand side and objective coefficients.

Sensitivity analysis. Consider the primal problem (MILP). This problem can be defined by the quadruple (A, b, c, r). We are interested in examining the optimality of a modified instance $(\tilde{A}, \tilde{b}, \tilde{c}, \tilde{r})$. Assume both problems are feasible and the original problem is solved to optimality. Let x^* be the optimal solution to the original problem. The following result shows the conditions under which x^* remains optimal for the general case where new problem is obtained by modifications in the right-hand side, coefficient matrix and objective coefficients.

Proposition 4.1. *(Geoffrion and Nauss, 1977; Wolsey, 1981a; Güzelsoy, 2009)*

(i) If the original problem is a relaxation of the new problem and x^* is feasible for the new problem, then x^* remains optimal.

(ii) When a new activity $(\tilde{c}_{n+1}, \tilde{A}_{n+1})$ is introduced, $(x^*, 0)$ remains feasible.

(iii) When a new constraint $\tilde{a}_{m+1}$ with right-hand side $\tilde{b}_{m+1}$ is introduced, if x^* is feasible then it remains optimal for the new problem.

(iv) Let $\tilde{c}$ satisfy $\tilde{c}_j \geq c_j$ for all j such that $x^*_j = 0$, and $\tilde{c}_j = c_j$ for $x^*_j > 0$. Then x^* remains optimal for the new problem.

Furthermore, if F^* is a subadditive dual function, we have

(v) If a new activity $(\tilde{c}_{n+1}, \tilde{A}_{n+1})$ is introduced and $F^*(A_{n+1}) \leq \tilde{c}_{n+1}$, then $(x^*, 0)$ is the optimal solution to the new problem.

(vi) Let $\tilde{F} : \mathbb{R}^{m+1} \to \mathbb{R}^m$ be defined by $\tilde{F}(b, b_{m+1}) = F^*(b)$. Then $\tilde{F}$ is dual feasible for the new problem.

(vii) If the right-hand side is changed to $\tilde{b}$, then F^* remains dual feasible for the new problem. Therefore, $c^\top \tilde{x}^* \leq F^*(\tilde{b})$. If F^* also remains optimal, then the new optimal solution is in $\{y \mid F^*(Ay) = c^\top y\}$, since from $z(\tilde{b}) = c^\top \tilde{x}^*$ we have $c^\top \tilde{x}^* = F^*(\tilde{b}) = F^*(A\tilde{x}^*)$.

The above observation is interesting in theory, but in practice, we are most interested in obtaining the optimality conditions of a modified MILP instance when the original instance is solved with a primal algorithm, particularly when primal algorithm is one of the well-known MILP algorithms, such as the cutting-plane or branch-and-bound methods. We discussed earlier that in the case where the original instance is solved by a cutting-plane method, a subadditive dual can be derived as a by-product of the cutting-plane method when the subadditive representation of each cut added to the description of the problem is known. We showed this parametric representation for a generic cut in (2.50). We also discussed that such a representation is known for certain classes of cuts, such as Gomory cuts whose subadditive representation was given by Wolsey (1981a). For this case, sufficient conditions for optimality of a modified instance are given by Klein and Holm (1979) for PILPs. We provide two key results next.

Consider the relaxed problem (MILP) obtained at iteration i of the cutting-plane algorithm with Gomory cuts. Let B^i be the set of indices of variables in the optimal basis in iteration i and F_i and σ_i be the functions defined in (2.50) obtained at iteration i.

Proposition 4.2. *If $\tilde{x} = (A_{B^i})^{-1}\sigma_i(\tilde{b})$ is non-negative and integer, then $\tilde{x}$ is an optimal solution to the modified problem.*

The above result is strengthened if we further have the integrality of $(A_{B^i})^{-1}$.

Theorem 4.1. *If $\tilde{x} = (A_{B^i})^{-1}\sigma_i(\tilde{b})$ is non-negative and $(A_{B^i})^{-1}$ is integer, then then $\tilde{x}$ is an optimal solution to the modified problem.*

Güzelsoy (2009) proposes a different set of optimality conditions using upper and lower bounding approximations for the value function. This result is based on the observation that lower approximating functions for the value function can be found simply by taking the maximum of the dual function obtained so far, as follows

$$\underline{z}(b) = \max_i F^i(b). \tag{4.1}$$

To obtain an upper bounding function, we can write

$$\bar{z}(b) = \begin{cases} \min_i\{c_{B^i}^\top (A_{B^i})^{-1}\sigma_i(b)\} & \text{if } (A_{B^i})^{-1}\sigma_i(b) \geq 0 \text{ and integer for some i} \\ \infty & \text{otherwise} \end{cases}$$

(4.2) is an upper bounding function because it guarantees the feasibility of the basis in some iteration $1, \ldots, i$. The following result states the optimality condition for the modified instance.

Proposition 4.3. *For a modified right-hand side $\tilde{b} \in \mathbb{R}^m$, if $\bar{z}(\tilde{b}) = \underline{z}(\tilde{b})$, then the lower bounding function $\underline{z}$ is strong at $\tilde{b}$.*

Güzelsoy (2009) further extends this result to the case where the primal instance is solved with branch-and-bound or branch-and-cut algorithm by modifying the upper and lower bounding functions. We next provide more details on these functions.

Consider the branch-and-bound dual function (2.57) that we introduced in Section 2.6 and used in the Generalized Benders' algorithm. The function (2.57) is constructed by collecting the dual information of the LP relaxations of the terminating nodes. We explained that because at any stage during the execution of the branch-and-bound method we have a valid partition of the feasible region of the LP relaxation, by collecting the dual information of the leaf nodes of the tree in any intermediate iteration we can obtain

a dual function. Such a dual function, however, may not be strong. Nevertheless, such dual functions potentially approximate different parts of the value function and their dual functions can be combined to strengthen (2.57). This observation was further generalized by considering the dual functions obtained from subtrees of the branch-and-bound tree resulting from evaluating the value function at a given point. Let $\mathcal{H}$ be the set of all connected subtrees rooted at the root node such that both left and right children of an intermediate node is also in the subtree and suppose $\mathcal{I}_h$ indexes the set of leaf nodes of $h \in \mathcal{H}$. Then the following function strengthens (2.57).

$$\underline{z} = \max_{h \in \mathcal{H}} \min_{t \in \mathcal{I}_h} (b^\top \eta^t + \alpha^t) \quad \forall b \in \mathbb{R}^m. \tag{4.2}$$

Schrage and Wolsey (1985) provide a recursive algorithm to evaluate the above formulation. Let t be a node of the branch-and-bound tree. If t is not a leaf node, let $L(t)$ and $R(t)$ respectively be its left and right child. Let κ^t be the objective function of the dual to the problem associated with t parameterized in the right-hand side. That is

$$\kappa^t(b) = b^\top \eta^t + \alpha^t. \tag{4.3}$$

Proposition 4.4. *(Schrage and Wolsey, 1985) For each node t belonging to the branch-and-bound tree, let the function $\zeta^t : \mathbb{R}^m \to \mathbb{R}$ be defined as*

$$\zeta^t(b) = \begin{cases} \kappa^t(b) & \text{if } t \text{ is a leaf node} \\ \max\{\kappa^t(b), \min\{\kappa^{L(t)}(b), \kappa^{R(t)}(b)\}\} & \text{otherwise.} \end{cases}$$

Then, $F(b) = \zeta^0(b)$, where F is defined in (4.2).

A dual function in the form of (4.2) can be derived similarly for the branch-and-cut algorithm when the subadditive representations of the cuts used are available. In this case, for each node t of the tree we have the following modification of (4.3)

$$\kappa^t(b) = b^\top \eta^t + \alpha^t + \sum_{i=1}^{k(t)} w_i^t F_i^t(\sigma_i^t(b)) \quad \forall b \in \mathbb{R}^m, \tag{4.4}$$

where $k(t), F_i^t$ and σ_i^t are defined as in (2.66).

It remains to obtain an upper bounding function to combine with the related lower bounding function (4.2) to have a result analogous to Proposition 4.3.

Proposition 4.5. *(Güzelsoy, 2009) For a given $\hat{b} \in \mathbb{R}^m$, let t be a node of the optimal tree used to evaluate $z(\hat{b})$. Also, let $T_U(\tilde{b})$ be the set of nodes of the tree whose basis yields a basic feasible solution $x^t \in S^t(b) \cap \{x \in \mathbb{Z}_+^r \times \mathbb{R}_+^{n-r}\}$ with $S^t(b)$ defined in (2.54). Then,*

$$\bar{z}^t(b) = \begin{cases} \min_{t \in T_U(\tilde{b})} \; c^\top x^t & \text{if } \; T_U(b) \neq \emptyset \\ \infty & \text{otherwise.} \end{cases}$$

is an upper bounding function for z.

4.2 Warm Starting in SYMPHONY

We use SYMPHONY as the MILP solver to implement the generalized Benders' algorithm. SYMPHONY is a customizable open-source software framework to solve MILPs. This MILP solver is a part of the computational infrastructure for operations research (COIN-OR) project and is completely integrated with other projects within the COIN-OR framework. The solver has interfaces with both C and C++ and can be built on the Windows operating systems as well as Unix-like environments (see (Ralphs and Güzelsoy, 2005)).

SYMPHONY allows for modifying the default behavior of the solver by writing *callback functions* and changing various parameters. The user callback functions can be used to change the branching rules, node selection and diving strategies, cutting plane generation and management of the cut pool. These functions are implemented in both C and C++. We use the C interface in our implementation. SYMPHONY has a modular implementation. The functions implemented within the solver are grouped into the following modules

– The master module. This module is the i/o handler. It contains functions for reading and storing the problem data, computing bounds and storing the information

of the best solution.

- The tree manager module. This module maintains the search tree, tracks the upper bound, generates the children of the active node to add to the candidate list.
- The node processing module. Processing of candidates, as well as feasibility checks and the selection of branching objects are carried out in this module.
- The cut generator module. This module interacts with the node processing module by taking the solution of the candidate node LP and returning a collection of cuts violated by the solution.
- The cut pool management module. This module maintains a list of "effective" inequalities and returns those that are violated by a solution to the LP. These cuts are returned to the node processing module to be added to the candidate node LP.

SYMPHONY is capable of performing warm-starting and sensitivity analysis on MILPs. The primary algorithm that SYMPHONY uses is branch-and-cut. When the user enables that warm-starting description should be kept via specifying the appropriate parameter, the solver collects and stores information from the final tree used to solve the problem, as well as other auxiliary information needed to restart the computation of a potentially modified instance. The information collected from the tree contains information about each solution to the relaxation associated with each node in the tree, including the list of active variables and constraints and the branching decisions that have led to the nodal problem, as well as information needed to perform sensitivity analysis on the nodal LP including the optimal basis and dual information. The auxiliary information stored includes information about upper and lower bounds on the tree and the best feasible solution found so far. After the warm-start description is stored from the initial solve, the user can modify the parameters used to solve the initial problem and make a call to **sym_warm_solve()** to solve the modified instance.

In the case of parameter modification, the master module stores the necessary information from the final solve of the original instance. The tree manager module then

```
1 #include "symphony.h"
2
3 int main(int argc, char **argv)
4 {
5    sym_environment *env = sym_open_environment();
6    sym_parse_command_line(env, argc, argv);
7    sym_load_problem(env);
8
9    sym_set_int_param(env, "keep_warm_start", true);
10   sym_set_int_param(env, "node_selection_strategy",
        DEPTH_FIRST_SEARCH);
11   sym_set_int_param(env, "find_first_feasible", TRUE);
12
13   sym_solve(env);
14
15   sym_set_int_param(env, "node_selection_strategy",
        BEST_FIRST_SEARCH);
16
17   sym_warm_solve(env);
18   sym_close_environment(env);
19   return(0);
20 }
```

Figure 4.1: Use of SYMPHONY warm-start with parameter modification

traverses the tree and marks the leaf nodes as candidates and prepares them for further processing. A candidate node is selected according to the new node selection rule and the algorithm continues as usual. Figure 4.1 shows the code of a basic example of storing the warm-start description and using it to solve a modified instance in the C API.

In Figure 4.1, line 5–6 are respectively responsible for opening a new SYMPHONY environment and parsing the user setting and reading the name of the problem. Line 7 reads the problem data and sets it as the root subproblem. Line 9 tells SYMPHONY to store the warm-start information from the final solve to be used after modifications to the parameters. Line 10 and 11 respectively require SYMPHONY to use the depth-first-search strategy to solve the problem until a feasible solution is found. After the problem is solved, in line 15, best-first-search is used as the search strategy and the modified problem is resolved by making a call to **sym_warm_solve()**. In addition to what is discussed here, SYMPHONY offers a variety of options for customizing model parameters and solution

methods. For a comprehensive reference for the solver we refer the reader to the solver manual which is due to Ralphs et al. (2013).

In the context of stochastic optimization, we are often interested in solving scenario subproblems which have the same structure, but differ in problem data. The SYMPHONY warm-start can also be used to solve such modified instances. Currently, changes in the vectors of right-hand side and objective function coefficients are permitted when warm-starting a solve call. When a call to warm-solve is invoked after changes to the these data, SYMPHONY must make appropriate modification to the leaf nodes of the tree whose information is stored in a warm-start description structure. In the case in which branch-and-cut is the solution algorithm, for example, the solver needs to ensure that all the cuts that are added to the description of the nodal problem remain valid for the modified nodal instance. After the modifications, each leaf node should be marked as a candidate to be reprocessed. The algorithm proceeds as usual from the resulting tree. The information from the leaf nodes obtained from the old tree can be used to expedite solving the new problems. For instance, the optimal basis obtained in the leaf node in the initial solve can be used as a starting basis for the new nodal instance. We show an example of the code for warm-starting a problem with modifications in the right-hand side of the first constraint in the original problem in Figure 4.2.

4.3 The Generalized Benders' Algorithm

In this section we provide details on the implementation of the Generalized Benders' algorithm. We overview the main modules and the options for solving the master and subproblems. We explain different methods to construct the master problem to control its size and describe different warm-starting strategies. Finally, we apply the algorithm with the above options to the Stochastic Server Location (SSLP) problems from the Stochastic Integer Programming Library (SIPLIB) and report the result of the computational experiments. We begin by reviewing the implementation.

```
1  int main(int argc, char **argv)
2  {
3     sym_environment *env = sym_open_environment();
4     sym_parse_command_line(env, argc, argv);
5     sym_load_problem(env);
6
7     sym_set_int_param(env, "keep_warm_start", true);
8     sym_solve(env);
9
10    /* Change the upper bound of the first constraint to 1.2 */
11    sym_set_row_upper(env, 0, 1.2);
12
13    sym_warm_solve(env);
14    sym_close_environment(env);
15    return(0);
16 }
```

Figure 4.2: Use of SYMPHONY warm-start with problem data modification

Implementation Details

The Generalized Benders' algorithm is implemented in C and uses the SYMPHONY callable library to solve the master and subproblems. The implementation consists of the following modules

- Main Module
- Master Solution Module
- Recourse Solution Module
- Read Duals Module.

We highlight the important details of each module next.

Main module. This module contains calls to the main components of the Generalized Benders in 3.4. It starts with reading the master, subproblem and stochastic file, which contains the right-hand sides associated with each scenario and the probability of each scenario. It then proceeds with solving a relaxation of the input master problem, fixing the right-hand sides of the subproblems accordingly. The main loop iterates between

```
1  get_input_args();
2  solve_original_master_problem();
3  initialize_subproblems_rhs();
4  initialize_bounds();
5  do{
6     solve_master();
7     update_LB();
8
9     for each scenario do{
10       setup_subproblem();
11       solve_subproblem();
12       update_exact_expected_recourse();
13       construct_dual_functions();
14       store_dual_functions();
15    }
16
17    update_UB();
18 } while (!check_terminate())
```

Figure 4.3: Sketch of the main module

the solve-master module and solve-recourse module. Throughout the execution, we use a single environment for all the master problems, therefore, the approximations obtained from solving the subproblems can be stored in the description of the master problem and get updated as new approximations become available. The updates in the upper and lower bounds of the problem is performed in this module using the output of the solve-master and solve-recourse modules respectively. The main steps are illustrated figure 4.3.

Master Solution Module. This module receives the description of the master problem, updates it with the new approximations obtained from solving the subproblems and calls SYMPHONY to solve it. After the original master problem is read from the input (a modified .cor file), the problem is reformulated by adding the expected recourse variables z_ω in MP for all scenarios $\omega \in \Omega$. In each iteration of the algorithm, it adds the variables $q_{t,\omega}$ in MP for all the scenarios and the leaf node of the tree used to solve the scenario subproblem denoted by t. It then populates the master problem with the approximation and makes a call to the MILP solver.

Recourse Solution Module. This module takes the solution to the master problem and uses it to update the right-hand sides of the recourse problems with the stochastic input .sto file. Then, it makes calls to SYMPHONY with user options and the selected warm-start option. The necessary SYMPHONY options to solve the recourse problems are

```
sensitivity_analysis 1
generate_cgl_cuts 0,
```

where the first option is needed for SYMPHONY to store the dual information which are used for sensitivity analysis, and the second option if to disable cut generation, which we discussed in section 3.4 is required for the algorithm. Finally, pointers to the description of the recourse subproblems are preserved or discarded depending on the warm-start option.We will discuss the later option later in this chapter.

Read Duals Module. To retrieve the dual information of the leaf nodes of the branch-and-bound tree used to solve the scenario subproblems, the `sym_get_dual_pruned` function is added to the callable library of SYMPHONY. The dual information is retrieved by a recursive call to a subroutine that gathers the related data from the description of the tree. This description is kept in the `warm_start_desc` structure stored in the problem's environment, `env->warm_start`.

Alternative Warm Starting Strategies

The most straightforward method of constructing dual functions from branch-and-bound trees is to start a new tree every time a scenario is solved and refine this tree until optimality is achieved. From each scenario subproblem tree, a dual function is obtained that can be used to strengthen the approximation of the value function for that or any other scenario subproblem. In the case that $|\Omega|$ separate approximations are kept, this strategy is analogous to the multi-cut Benders' method. Since each time a new tree is constructed from scratch, the call to solve the recourse problem is simply

```
1 /* Store the warm starting descriptions in an array of size
2   number of scenarios */
3 warm_start_desc** wsArray = (warm_start_desc**)
4       malloc(sizeof(warm_start_desc*)* num_scen);
5
6 /* Keep the warm start description in the first iteration and
     use it in the following iterations */
7 sym_set_int_param(recourse_environment, "keep_warm_start", TRUE)
    ;
8
9 if (iteration_number == 1){
10    sym_solve(recourse_environment);
11  } else {
12   sym_set_warm_start(recourse_environment, wsArray[scen]);
13   sym_warm_solve(recourse_environment);
14 }
15
16 wsArray[scen] = sym_get_warm_start(recourse_environment, TRUE);
```

Figure 4.4: Warm-starting the solution to each scenario

`sym_solve(recourse_environment).`

The second strategy that we examine is to warm solve the solution to each scenario by keeping a single tree for each scenario. Each of the $|\Omega|$ trees is created in the first call to the solve-recourse module. The description of the trees are then kept in memory and is retrieved in the next solve to that problem. The previous tree is refined further to obtain an optimal tree for the new scenario. The following figure provides more details.

As a third alternative, the description of the first scenario in the first iteration can be used as the initial tree to solve all the scenario subproblems that arise during the execution of the algorithm by refining the tree. The block of code used for this option is shown in Figure 4.5. A pointer to a single warm starting description, `warm_start_desc** ws`, is allocated in the beginning of the algorithm and is kept during the execution of it.

With any of the choices above, the implementation accommodates combining the approximation of two or more scenarios by taking the maximum of the approximations and using a single approximation for the combined scenarios. Whether using few or many approximations is beneficial depends on the specifics of the problem. Keeping one or a few

```
1 if (iteration_number == 1 &&  scen == 0){
2   sym_solve(recourse_environment);
3 } else {
4   sym_set_warm_start(recourse_environment, *ws);
5   sym_warm_solve(recourse_environment);
6 }
7 *ws = sym_get_warm_start(recourse_environment, TRUE);
```

Figure 4.5: Warm-starting a single tree

number of approximations is desirable if the subproblems require exploring roughly the same area of the domain of the value function of the second-stage. In this case, multiple approximations are combined to construct a stronger approximations of the value function over a local region of the right-hand sides. In the case that the stochasticity is coming from the scenarios or the first stage solutions result in examining the second-stage value function sparsely, keeping separate approximations may result in a more effective overall approximation.

4.4 Computational Experiments

In this section, we illustrate the application of the Generalized Benders' algorithm on the several examples from the literature and the Stochastic Integer Programming Library (SIPLIB). All the experiments we report are run on a Linux (Debian 6.0.9) operating system running 16 AMD processors on a 800 MHz speed and 31 GB RAM, compiled with g++. The mixed integer optimization solver used is SYMPHONY version 5.6 customized for our specific application and configured with the sensitivity analysis option.

Examples from the literature

The first problem we solve is originally developed by (Schultz et al., 1998) and subsequently used in (Ahmed et al., 2004) and (Gade et al., 2012). This problem consists of a pure binary first-stage problem and a mixed binary recourse with an auxiliary variable that is added to the recourse problem in order to ensure relative completeness.

Example 4.1.

$$\begin{aligned} f(b) = \inf \ & -1.5x_1 - 4x_2 + \mathbb{E}[z(x,\omega)] \\ \text{s.t. } & x_1, x_2 \in \mathbb{B}^1, \end{aligned} \tag{4.5}$$

where

$$\begin{aligned} z(b) = \inf \ & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\ \text{s.t. } & 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \leq h(x) - x_1 \\ & 6y_1 + y_2 + 3y_3 + 2y_4 - R \leq h(x) - x_2 \\ & y_i \in \mathbb{B}, i = 1, \ldots, 4, R \in \mathbb{Z}_+. \end{aligned} \tag{4.6}$$

Here, we have two scenarios $\Omega \in \{\omega_1, \omega_2\}$, each with probability of 0.5. The distribution of the right-hand side is $(h_1(\omega_1), h_2(\omega_1)) = (5, 2)$ and $(h_1(\omega_2), h_2(\omega_2)) = (10, 3)$.

We apply the generalized Bender's method to this problem. The algorithms starts with solving $\inf\{-1.5x_1 - 4x_2 \mid x_1, x_2 \in \mathbb{B}^1\}$ and finds $x_1^* = x_2^* = 1$ and lets $\beta_i = x_i^* = 1$ for $i = 1, 2$. In the first iteration, the right hand sides of the constraints of (4.6) is then $(4, 1)$ and $(9, 2)$ for the first and second scenarios, respectively. From solving the recourse problem in the first scenario, the expected recourse is $0.5(-19 - 38) = -23.5$. The upper bound on the problem is -29. The master problem after adding the dual functions obtained form solving the two subproblem can be formulated as follows.

$$\begin{aligned} \inf \ & -1.5x_1 - 4x_2 + 0.5(\underline{z}_1 + \underline{z}_2) \\ \text{s.t. } & \underline{z}_1 = 14x_2 - 33 \\ & \underline{z}_2 \leq 7.67x_2 - 34.33 \\ & \underline{z}_2 \geq 7.67x_2 - 34.33 - Mu_{1,2} \\ & \underline{z}_2 \leq 100x_2 - 128 \\ & \underline{z}_2 \geq 100x_2 - 128 - Mu_{2,2} \\ & u_{1,2} + u_{2,2} = 1 \\ & x_1, x_2, u_{1,2}, u_{2,2} \in \mathbb{B}^1. \end{aligned} \tag{4.7}$$

Solving (4.7) results in a lower bound of -82 with $x_1^* = 1, x_2^* = 0$. The new right-hand

sides are $(4, 2)$ and $(9, 3)$. The upper bound on the problem is then updated to -34.5. Solving the master problem, we obtain the lower bound of -37.5 with $x_1^* = x_2^* = 0$. The new dual function added to the master problem is

$$\max\{\min\{100x_1 - 28, 100x_2 + 77, 14x_2 + 35, 19\}, 7.67x_2 - 47\}. \tag{4.8}$$

After solving the two new subproblems with right-hand sides of $(5, 2)$ and $(10, 3)$, the new upper bound on the problem is -37.5 and the algorithm is terminated. The following table summarizes these results. In this table, the first column is the iteration number and the second column shows the optimal solution obtained from solving the corresponding master problem. The third and fourth column are the solutions to the first and second subproblems, respectively. The last two columns are respectively the global upper and lower bound of the problem in the corresponding iteration. □

k	x^*	Scen.1 obj.	Scen.2 obj.	Global UB	Global LB
1	(1,1)	-19	-28	-29	-82
2	(1,0)	-19	-47	-34.5	-37.5
3	(0,0)	-28	-47	-37.5	–

Table 4.1: Iterations of the Generalized Benders' algorithm applied to Example 4.6

We next report the computational result of applying the algorithm on larger instances from (Gade et al., 2012).

Example 4.2. Consider the problem solved in Example 4.1 with the following modification. Let $y_i \in \{0, 1, \ldots, 5\}, i = 1, \ldots, 4$, $(h_1(\omega_1), h_2(\omega_1)) = (10, 4)$ and $(h_1(\omega_2), h_2(\omega_2)) = (13, 8)$. Table 4.2 shows the results of applying the algorithm to this problem.

k	x^*	Scen.1 obj.	Scen.2 obj.	Global UB	Global LB
1	(1, 1)	-57	-76	-76.5	-76.625
2	(0, 1)	-57	-80	-72.5	-72.5

Table 4.2: Iterations of the Generalized Benders' algorithm applied to Example 4.2

To construct larger instances, we generate more scenarios by allowing the right hand sides of the two constraints to be in the set $\{5, 5+\Delta, 5+2\Delta, \ldots, 15\} \times \{5, 5+\Delta, 5+2\Delta, \ldots, 15\}$ to generate $4, 9, 36, 121$ scenarios with $\Delta \in \{1, 2, 5, 10\}$. We further extend the set by changing 15 to 20 to generate 225 scenarios with $\Delta = 1$. The size of the resulting problems is reported in Table 4.3.

No. Scen.	4	9	36	121	225
No. Vars	24	47	182	607	1127
No. Const	8	18	72	242	450

Table 4.3: Size of the deterministic equivalent of the test instances in Example 4.2

Table 4.4 shows the computational report of applying the algorithm to the resulted five problems. The second column in the table shows the optimal value of the problem obtained with the algorithm. The third column shows the number of iterations taken by the algorithm, column four shows the node and size of the master problem in the final iteration in the form of (number of variables, number of constraints). Columns five and six respectively show the user time spent to solve the master problems and the final solution time in seconds. These results are obtained by setting the warm-start option to the full-tree construction. For these examples, there was no significant difference in computational time observed with different warm-starting options.

	Obj	No. Iterations	Nodes/Size Final Master	Time in Master (s)	Total Time (s)
4-Scen	-63.50	3	1 (27, 47)	0.10	0.16
9-Scen	-65.67	3	1 (71, 127)	0.3	0.32
36-Scen	-66.83	3	17 (301, 552)	1.0	1.92
121-Scen	-67.17	4	55 (1022,1891)	11.36	15.91
225-Scen	-79.66	4	5 (1883, 3465)	16.00	27.40

Table 4.4: Performance of the algorithm applied to problems in Example 4.2.

□

The Stochastic Server Location Instances

We introduce the Stochastic Server Location Problem (SSLP) from the SIPLIB library next. These problems are developed by Ntaimo and Sen (2005). The objective of the problem is to maximize the revenue gained from serving clients with stochastic demands less the cost of locating servers. We denote by $\mathcal{I}$ and $\mathcal{J}$ the index set for the clients and servers respectively. We further let $\mathcal{Z}$ denote a given set of zones and $|\mathcal{I}| = n$ and $|\mathcal{J}| = m$. The data to the problem are

c_j: cost of locating a server at location j

q_{ij}: revenue from client i being served by server j

d_{ij}: client i resource demand from server j

u: server capacity

ν: an upper bound on the total number of servers that can be located

w_z: minimum number of servers to be located in zone $z \in \mathcal{Z}$.

$\mathcal{J}_z$: the subset of servers locations that belong to zone z

$$h^i(\omega) = \begin{cases} 1, & \text{if client } i \text{ is present in scenario } \omega \in \Omega \\ 0, & \text{otherwise.} \end{cases}$$

p_ω: the probability of scenario ω. The problem is formulated as a two-stage integer problem in the form of

$$\begin{aligned} \min & \sum_{j \in \mathcal{J}} c_j x_j + \mathbb{E}_\omega f(x, \omega) \\ \text{s.t.} & \sum_{j \in J} x_j \leq \nu \\ & \sum_{j \in \mathcal{J}_z} x_j \geq w_z, \quad \forall z \in \mathcal{Z} \\ & x \in \mathbb{B}^m, \end{aligned} \tag{4.9}$$

where the recourse problem is defined as

$$\begin{aligned}
&\min -\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} q_{ij}y_{ij} + \sum_{j\in\mathcal{J}} q_{j0}y_{j0} \\
&\text{s.t.} \sum_{i\in I} d_{ij}y_{ij} - y_{j0} \leq ux_j, \quad \forall j \in \mathcal{J} \\
&\sum_{j\in\mathcal{J}_z} y_{ij} = h^i(\omega), \quad \forall i \in \mathcal{I} \\
&y_{ij} \in \mathbb{B}, \quad \forall i \in \mathcal{I}, j \in \mathcal{J} \\
&y_{j0} \geq 0 \quad \forall j \in \mathcal{J},
\end{aligned} \tag{4.10}$$

where x_j for $j \in \mathcal{J}$ are the first-stage variables that are equal to 1 if a server is located at site j, 0 otherwise. y_{ij}^{ω} are the second-stage variables, equal to 1 if client i is served at location j under scenario ω, 0 otherwise. The first and second constraints in (4.9) respectively enforce the limit on the number of servers that can be located and the lower bound on the number of severs that are necessary within each zone. The recourse problem's objective is to maximize the revenue earned from serving the clients for a given server location and scenario. The first constraint in the recourse problem (4.10) enforces an upper bound equal to the capacity of a sever to the supply of a server j if that server is open and zero if the server is not open. The variable y_{j0} is added to ensure the relative complete recourse property for the problem. Finally, the last constraint guarantees that a client is only severed by one server.

We next report the result of applying the Generalized Benders' algorithm to the instances of the SSLP problem available in SIPLIB. We denote these instances by *sslp-m-n*($|\Omega|$) where m and n are respectively the number of potential servers and clients and $|\Omega|$ is the number of scenarios. Table 4.5 shows the properties of the deterministic equivalent of the problems. In this table, the *DEP* column described the deterministic equivalent problem and the *2nd Stage* column describes the recourse problem. The number of binary variables, integer variables and constraints of the problem are respectively shown by *bin, cons* and *int*. In the last two columns we report the solution time for the deterministic equivalent problem and the relative MIP gap at termination obtained from solving the

problem with SYMPHONY solver with the default settings. The experiments are run within a time limit of one hour.

Instance	DEP			2nd Stage			Time (s)	% Gap
	cons	bin	int	cons	bin	int		
sslp-5-25(25)	751	3130	125	30	130	5	2.23	0.0
sslp-5-25(50)	1501	6255	250	30	130	5	4.89	0.0
sslp-5-25(100)	3001	12505	500	30	130	5	14.37	0.0
sslp-10-50(50)	3001	25010	500	60	510	10	3600+	78.57
sslp-10-50(100)	6001	50010	1000	60	510	10	3600+	77.21
sslp-15-45(5)	301	3390	75	60	690	15	3600+	33.44
sslp-15-45(10)	601	6765	150	60	690	15	3600+	39.63
sslp-15-45(15)	901	10140	225	60	690	15	3600+	24.90

Table 4.5: The deterministic equivalent of SSLP instances

Table 4.6 shows the result of solving the SSLP instances with the Generalized Benders' algorithm under four settings. The first setting is the default setting where the subproblems are solved to optimality and optimal dual functions are collected after each solve and added to the master problem. This setting reflects the default behavior of the algorithm. Under the second setting, a limit on the number of nodes of the branch-and-bound tree used to solve the recourse subproblems is enforced. In our experiments, we set an initial limit of 10 nodes in the solution of the subproblems. The idea behind this setting is that by limiting the number of nodes in the solution tree, the number of dual functions collected from the tree becomes fewer which in turn, limits the size of the master problem. Since in any stage of solving the tree, we can obtain valid dual functions, the algorithm remains valid. However, since the collected dual problem is not necessarily optimal for the corresponding subproblem, the exact expected recourse value with the fixed first-stage solution may not be obtained. When solving the subproblem with the node limit, if the solver returns a feasible solution without finding an optimal solution, the feasible solution still provides an upper bound for the subproblem and therefore can be used to construct an upper bound for the expected recourse. This upper bound can be used to check whether the global upper bound of the problem can be updated, as we did in the default setting of the algorithm.

In the case where the solution of the subproblem with node limit is terminated with-

out finding any feasible solution, the subproblem is solved by enabling the parameter *find_first_feasible*. The returned feasible solution then can be used to update the upper bound. Regardless of whether a feasible solution is found or not, it is possible that after updating the master problem with the new dual functions, the solution to the master problem does not get updated. In that case, the limit on the maximum number of nodes in the solution tree to the subproblem is increased (by 10 nodes in our experiments) and the subproblems are resolved with the new node limit only in the subsequent iteration. The node limit then is fixed back to the initial limit for the following iterations.

Table 4.6 summarizes the results of applying the Generalized Bender's algorithm to SSLP problems under the default setting with no node limit and the second setting where an initial node limit of 10 is enforced when solving the recourse subproblems.

For each of the settings, the number of iterations of the algorithm at the time of termination is reported in the first column. The second column shows the size of the master problem at the time of the termination in the form of number of $q(n, m)$ where q is the number of nodes in the branch-and-bound tree used to solve the master problem, n is the number of columns and m is the number of rows of the master problem in the final iteration. The user time is reported in seconds in the *Time* column. Finally, the *% Gap* column is the relative gap between the best upper and lower bounds on the problem obtained at the time of termination.

Within the time limit of an hour, the algorithm solved the sslp-5-25 instances to optimality under both settings. The solution time for these instances were significantly more than the solution time spent in solving the deterministic equivalent. This increase in the solution time can be explained by the increase in the size of the problems. In the final master problem solved for the sslp-5-25 problem with 25 scenarios, the number of constraints are about three times more than the one of the deterministic equivalent, reported in Table 4.5. Given that our reformulation of the master problem does not include the subproblems' variables, the final master problem has fewer binary variables than the deterministic equivalent. However, the final iterations of the Generalized Benders' algorithm involve solving master problems of similar sizes to the master problem

solved in the final iteration. This alone accounts for a solution time that is multiple times larger than the time spent to solve the deterministic equivalent. The overhead of pre-processing each master problem, solving the subproblems and collecting and storing the dual functions are other additional steps affecting the increased time of the algorithm. In an attempt to control the size of the master problem and reduce the time spent in solving the recourse problems, we solve the same problems with enforcing a node limit to the solution of subproblems. The results in Table 4.6 show that this setting reduces the overall computational time and results in smaller master problem in the final iteration for all three sslp-5-25 instances. The iteration number remains the same under both settings, implying that the updated master problems in the iterations where the node limit was reached for one or more subproblems resulted in updated right-hand sides for the subproblems, therefore, no increase in the node limit was necessary in any iteration.

For the larger instances of sslp-10-15 and sslp-15-45, although the size of the master problems is kept small during the execution of the algorithm, the solver spends more time in solving the instances compared with the DEP. For these instances, the majority of the solution time is spent in solving the master problems. The number of nodes in the branch-and-bound tree at the time of termination for these instances explain the few number of iterations completed within the time limit of an hour. For all of the SSLP instances, the solution to the first master problem solved in the Generalized Bender's method, i.e., the lower bound to the problem, is a small number in the order of 10^{-6}. For the sslp-5-25 instances, the dual functions obtained in the first few iterations, when incorporated in the master problem, result in an increase of the lower bound so that the optimality gap is about 1%, which is then reduced to 0% in the subsequent iterations. For other SSLP instances, however, the few iterations completed within the time limit do not result in a significant increase in the lower bound. For example, in the first two iterations of the algorithm when solving sslp-10-15(50) instance, the lower bound increases from -877041.8 to -876975.8. As a result of this slow change in the lower bound, the optimality gap remains large within the time limit of an hour.

For the sslp-15-45 instances, the algorithm completed more iterations with the limit

on the number of the nodes. The additional iterations are a result of the ineffective constraints added to the master problem when the subproblems are not solved to optimality. As a result of the additional iterations, the size of the master problems are increased compared to the default setting with no limit on the nodes. While the relative optimality gaps remain large in all the three instances, the dual functions collected in the additional iteration performed in instances with 10 and 15 scenarios result in a smaller gap than the ones of the default setting. In contrast, the additional iterations completed in the instance with 5 scenarios not only resulted in a larger master problem obtained in more iterations, but also did not result in a smaller optimality gap. In this case, the dual functions collected from subproblems result in redundant constraints in the master problem which lead to no change in the lower bound to the problem. Subsequently, the increase in the node limit and the resolve of subproblems followed by changing the allowed number of nodes to the initial value in the following iterations resulted in a weaker dual function compared to the one obtained in fewer iterations of the algorithm completed when no node limit was placed in solving recourse subproblems.

In the third and fourth settings, we experiment with the two warm starting strategies we explained in Section 4.3. The summary of the performance of our algorithm under these settings is reported in Table 4.7. For the smaller SSLP instances, both of these settings resulted in larger computational times compared with the first two settings where no warm-starting was used. The main factor that contributed to the larger master problems in both settings is the larger number of leaf nodes in the warm-started branch-and-bound trees used to solver the recourse problems. In particular, when the $|\Omega|$ branch-and-bound trees are warm-started one for each scenario subproblems, for sslp-5-25 test instances the size of the final master problem grows to almost twice the size of the final master problem in the default setting. For the larger instances, the master problems are also larger than the ones resulted from the default settings. While for the two instances of sslp-15-45 with 10 and 15 scenarios an additional iteration is completed, the reduction in the gap is far from desirable and we have another case of the slow convergence of the algorithm. When warm-starting the scenario subproblems are all conducted on a single tree, the sizes of the

master problems solved are much larger than the previous three settings. Warm-starting all subproblems with a single tree results in a very large tree whose leaf nodes may contain strong duals only for a small subset of subproblems. Given that in each iteration the dual functions are collected from all the leaf nodes, a large number of the dual pieces encoded in the master problem are weak or redundant for a given scenario. However, in order to keep the dual functions valid, importing the dual information of all the tree nodes to the master problem is inevitable. Warm-starting a single tree, even though in theory contains all the pieces of the recourse problem's value function needed to solve the two-stage problem, results in prohibitively large master problems solving which is a challenge in practice.

Instance	No Limit on Number of Nodes				Node Limit on Recourse (10 nodes)			
	Iteration	Size	Time (s)	% Gap	Iteration	Size	Time (s)	%Gap
sslp-5-25(25)	18	48282 (1145, 2326)	472	0.0	18	47553 (1040, 2016)	323	0.0
sslp-5-25(50)	18	89184 (2270, 4215)	582	0.0	18	59868 (2150, 4121)	379	0.0
sslp-5-25(100)	18	23350 (4161, 8108)	732	0.0	18	48487 (4132, 8081)	710	0.0
sslp-10-50(50)	3	669527(409, 716)	3600	2748	3	638661 (369, 667)	3600	2748
sslp-10-50(100)	3	614089 (788, 1413)	3600	2851	3	610048 (706, 1299)	3600	2743
sslp-15-45(5)	8	42660 (235, 410)	3600	810	12	92910 (547, 813)	3600	1044
sslp-15-45(10)	3	622605 (208, 352)	3600	1182	4	530449 (247, 489)	3600	1089
sslp-15-45(15)	3	1386489 (286, 434)	3600	3499	4	1304527 (302, 482)	3600	2899

Table 4.6: Generalized Benders' algorithm applied to SSLP instances

Instance	Warm starting individual subproblems				Warm starting subproblems on one tree			
	Iteration	Size	Time (s)	% Gap	Iteration	Size	Time (s)	%Gap
sslp-5-25(25)	17	1877 (3578, 6936)	710	0.0	12	31912 (31791, 43396)	3600	10.11
sslp-5-25(50)	18	1482 (4753, 9256)	749	0.0	7	41852 (25955, 34989)	3600	17.9
sslp-5-25(100)	18	16078 (9286,17450)	1130	0.0	12	27961 (32657, 53829)	3600	3.37
sslp-10-50(50)	3	223694(614, 1076)	3600	2746	2	654503 (946, 1197)	3600	2746
sslp-10-50(100)	3	112481 (829, 1420)	3600	2758	2	8289 (5989, 7040)	3600	2848
sslp-15-45(5)	8	1236 (934, 1981)	3600	1023	4	7849 (1228, 2328)	3600	2560
sslp-15-45(10)	4	230109 (420, 878)	3600	1089	2	428462 (1024, 1202)	3600	2591
sslp-15-45(15)	4	110789(1324, 3533)	3600	2927	2	839219 (1002, 1185)	3600	5349

Table 4.7: Generalized Benders' algorithm with warm start applied to SSLP instances

The SIZES Instances

We next use the SIZES test set from SIPLIB for our computational experiments. This test set consists of three test instances originally developed by Jorjani et al. (1999). The SIZES problem is a production substitution problem whose goal is to minimize the cost of production of items with a finite number of sizes under demand uncertainty. In this problem, a larger sized item can be used to meet the demand of a smaller sized item. Let the indices $i \in \{1, 2, \ldots, N\}$ and $t \in \{1, 2, \ldots, T\}$ respectively correspond to the item sizes and time periods. As before, we denote the a scenarios with $\omega \in \Omega$. The data to the problem is

$C_{t\omega}$: production capacity in period t in scenario ω

$d_{jt\omega}$: demand for item size j in period t in scenario ω

α_i: production cost for item i

β: set up cost

r: cutting cost, and the decision variables are

$x_{ijt\omega}$: the amount of item i produced in period t in scenario ω to meet the demand of an item size $j < i$.

$y_{it\omega}$: the amount of item i produced in period t in scenario ω

$z_{it\omega}$: 1 if item i is produced in period t in scenario ω, 0 otherwise.

The stochastic integer optimization problem as proposed by Jorjani et al. (1999) is

formulated as

$$
\begin{aligned}
\min \sum_{\omega\in\Omega} p_\omega \sum_{t=1}^{T}\sum_{i=1}^{N}[(\alpha_i y_{it\omega} + \beta z_{it\omega}) + r\sum_{j<i} x_{ijt\omega}] & \\
\text{s.t. } y_{it\omega} \le M z_{it\omega} & \quad \forall i,t,\omega \\
\sum_{i=1}^{N} y_{it\omega} \le C_{t\omega} & \quad \forall t,\omega \\
\sum_{t'\le t}[\sum_{j\le i} x_{ijt'\omega} - y_{it\omega}] \le 0 & \quad \forall i,t,\omega \\
\sum_{i=1}^{N} x_{ijt\omega} \ge d_{jt\omega} & \quad \forall j,t,\omega \\
z_{it\omega} \in \mathbb{B} & \quad \forall i,t,\omega \\
y_{it\omega} \ge 0 & \quad \forall i,t,\omega \\
x_{ijt\omega} \ge 0 & \quad \forall i,j,t,\omega \\
x, y, z \in \mathcal{N} &
\end{aligned}
\tag{4.11}
$$

The first constraints of (4.11) ensures that a set up cost is enforced if a given item is produced in a period. The second and third constraints enforce the capacity restrictions and the balance of the inventory respectively. The fourth constraint ensures that the demand for each item in each period is met. A more detailed description of the problem is provided in (Jorjani et al., 1999). The SIZES instances are two-stage integer problems generated by letting $t = 1, 2$ in (4.11). We first provide the description of the instances and show the results of solving their deterministic equivalent in Table 4.8. The experiments are run within a time limit of one hour. The column descriptions in this table are similar to Table 4.5. The last column shows the MIP gap obtained at termination. For the deterministic equivalent problem of the SIZES10 instance, the gap is 4.43% after one hour. The solver terminates with the optimal solution with 0 gap for SIZES3 and SIZES4.

The result of our experiments with the generalized Benders' algorithm on the SIZES instances is summarized in Tables 4.9 and 4.10. For all of the instances, the maximum time limit of one hour is reached. The number of iterations of the algorithm at the time

of termination is reported in the first column. The second column follows the notation described earlier for Table 4.6. Like before, the *Time* column shows the user time in seconds and the *% Gap* column is the relative gap between the best upper and lower bounds on the problem obtained at the time of termination.

Like the SSLP case, for SIZES instances solving the deterministic equivalent clearly results in better solution time and a smaller relative optimality gap. In contrast to the SSLP instances, imposing node limits on the size of the branch-and-bound tree used to solve the recourse problems result in no lower bound improvement in the master problem, which in turn increases the number of iteration of the algorithm while providing any decrease in the duality gap. Both of the settings with warm starting options resulted in a better bound for the SIZES4 instance than the setting with node limit. However, the size of the final master problems with warm-starting options is larger than the default setting and the setting with node limit. This is consistent with our results for the SSLP problems.

Together, the Generalized Benders' algorithm did not result in better solution times than solving the deterministic equivalent problems for both of the test sets we examined. Nevertheless, the implementation of the algorithm helped us better understand the extent to which the value function of the recourse problem has to be explored to solve the two-stage optimization problem. In addition, we have a better understanding of the dual function collected from warm-started branch-and-bound trees when solving the recourse problems. The lower performance of the algorithm is a result of the size of the master problems that grow to be larger than the deterministic equivalent of the original two-stage problem. Warm-starting recourse subproblems also resulted in larger optimal tree sizes for scenario subproblems compared with the default setting of our algorithm, which in turn increased the size of the master problems solved. Even though the master problems encode the polyhedral pieces of the recourse value function that are necessary to obtain strong bounds for the recourse subproblems, the large number of constraints and binary variables used in the problem's formulation make the problem challenging to solve in each iteration of the algorithm. A major computational drawback of formulating master

problems in the form of (3.16) is that the set of additional valid inequalities from the leaf nodes of the optimal branch-and-bound tree of a subproblem have to be added or dropped from the master problem all together in order to maintain the validity of the piecewise polyhedral dual function of the recourse value function. Treating individual valid inequalities independently, as in the case of a usual MILP, result in a problem that is neither lower nor upper bounding to the original two-stage problem. Therefore, even when the set of valid inequalities from a subproblem contain weak or redundant valid inequalities, they cannot be dropped from the master problem. This is in contrast with the usual cutting-plane algorithm applied to a MILP. In particular, when the deterministic equivalent problem is solved with the branch-and-cut algorithm, the MILP solver can treat each valid inequality independently and reduce the size of the problem by keeping only stronger cuts. In the case of the master problem, however, the same polyhedral piece of the value function can appear in the dual function of multiple scenario subproblems. Nevertheless, such a valid inequality has to remain in the master problem, since without it the piecewise polyhedral function is neither a lower not an upper bound to the recourse value function.

	DEP			2nd Stage				
Instance	cons	cols	bin	cons	cols	bin	Time (s)	% Gap
SIZES3	124	300	40	31	107	10	41.4	0.0
SIZES5	186	450	60	31	107	10	2383	0.0
SIZES10	341	825	110	31	107	10	3600	4.43

Table 4.8: The deterministic equivalent of the SIZES instances

	No Node Number Limit			
Instance	Iteration	Size	Time (s)	% Gap
SIZES3	3	636782 (1560, 2987)	3600	0.12
SIZES5	3	149740 (2548, 4989)	3600	20.81
SIZES10	1	4845862 (1513, 2889)	3600	-
	Node Number Limit on Recourse			
SIZES3	4	20449 (1293, 2451)	3600	42.72
SIZES5	5	85910 (2349, 5744)	3600	23.13
SIZES10	2	133934 (1684, 3219)	3600	-

Table 4.9: Generalized Benders' algorithm applied to SIZES instances

	Warm starting individual subproblems			
Instance	Iteration	Size	Time (s)	% Gap
SIZES3	4	47764 (3273, 6264)	3600	0.80
SIZES5	3	122198 (3767, 7324)	3600	11.37
SIZES10	2	4254983 (3133, 6119)	3600	-
	Warm starting subproblems on one tree			
SIZES3	3	159689(2927, 5700)	3600	1.81
SIZES5	2	195582 (4216, 8382)	3600	-
SIZES10	2	656519 (6460, 12772)	3600	-

Table 4.10: Generalized Benders' algorithm with warm start on SIZES instances

Chapter 5

Conclusions and Future Research

In this dissertation, we study the structure of the general MILP value function and show that it can be represented by a countable set of right-hand sides, propose a value function based reformulation for the general two-stage stochastic mixed integer linear optimization problem and develop a Benders' decomposition algorithm to solve such problems. We also show the performance of the algorithm when applied to several test instances.

The value function of a MILP is key to sensitivity analysis for integer optimization as well as development of solution methods for various classes of optimization problems. The backbone of our work is to derive a discrete characterization of the MILP value function. We identify a countable set of right-hand sides which is sufficient to describe the discrete structure of the value function and use this set to propose an algorithm for the construction of MILP value function. This algorithm is finite when the set of right-hand sides over which the value function of the associated pure integer problem is finite is bounded.

We further study the connection between the MILP, PILP and LP value functions. In particular, we show that the MILP value function from the combination of a PILP value function and a single LP value function. We address the relationship between our representation and the classic Jeroslow formula for the MILP value function. We study the continuity and convexity properties of the value function, as well as the relationships between several critical sets of the right-hand sides such as the set of discontinuity and

non-differentiability points. As a result of our work, we now have a method to dynamically generate necessary points to describe a MILP value function. A subset of such points can be used to derive functions that bound the value function from above, while the full collection of them is sufficient to have a complete characterization of the value function. The dynamic generation of these points can be integrated with iterative methods to solve stochastic integer and bilevel integer optimization problems.

We propose a reformulation of the general two-stage stochastic mixed integer linear optimization problem based on a discrete representation of the value function and show how that leads to a generalization of Benders' algorithm. Our implementation of the algorithm utilizes dual functions obtained as by-products of the branch-and-bound procedure for solving the second-stage problem. Such branch-and-bound dual functions, if derived for each scenario in each iteration, yield a convergent version of Benders' method. Solving the subproblems from scratch and incorporating the large number of dual functions that would arise into the master problem, however, seems unlikely to result in a scalable algorithm. To address the challenge of this very difficult problem class, we propose to improve the basic method with the addition of a warm-starting strategy in which all subproblems are solved within a single branch-and-bound tree and in which we use a single continuously refined lower approximation of the value function within the master problem. By taking a global view of the construction of dual functions, we are able to derive stronger dual functions. As a corollary to our approach, we have further shown that there exists a single branch-and-bound tree from which we can derive the full value function of the second-stage problem, though it is unnecessary to do so in practice. We provide details on the implementation of this algorithm. In particular, we implement two warm-starting strategies for the recourse subproblems which results in constructing different approximations of the second-stage value function. We provide computational results on the performance of the algorithm and compare different strategies, including limiting the size of the branch-and-bound tree used to solve the recourse problems in addition to the warm-starting strategies.

The challenge that we have in practice with the generalized Benders' method is the

size of the master problem that grows very fast as a result of the strong dual functions of the recourse value function encoded in it. In fact, in our computational experiments, the master problems solved during the execution of the generalized Benders' algorithm grow to be larger than the corresponding deterministic equivalent problems. This is a consequence of the structure of the dual functions we use in formulating the master problem. To maintain the validity of these functions as dual functions, the dual information of all leaf nodes of the branch-and-bound tree should be kept each time a new scenario subproblem is solved. Nevertheless, many of the collected dual polyhedral functions contain the same dual information but cannot be dropped from the formulation since they are a necessary part of a piecewise polyhedral function. Therefore, individual pieces of the piece-wise polyhedral functions cannot be dropped. Dropping a dual function with many repetitive pieces all together can also potentially make the remaining functions in the master problem weak with respect to the right-hand side that led to the dropped dual function in the first place. Given that many individual pieces belonging to different dual functions have the same dual information (i.e., the gradient of the polyhedral function) and only differ in the intercept, one extension of our work is finding formulations of the master problem that takes advantage of this similarity by combining two or more piecewise polyhedral dual functions derived from distinct scenario subproblems such that the combined function remains strong at all the right hand-sides leading to the original dual functions. Doing this, the repeating pieces can be dropped to represent a unified function which in turn results in fewer binary variables in the formulation of the master problem.

Another extension of our work is to only keep the strong polyhedral pieces of the dual functions collected for a scenario subproblem and relax the remaining (potentially many) polyhedral pieces by replacing them with a much simpler function that is still lower bounding to the value function of the recourse problem. Such surrogate pieces can potentially be derived from the structure of the value function of the linear relaxation of the recourse problem or take more complicated forms. Then, the additional binary variables added to the master problem for representing the weak pieces can also get eliminated and replaced by far fewer binary variables needed to represent the simpler

polyhedral functions. In contrast, another extension of our work is to start with the value function of the linear relaxation of the recourse problem as the initial lower bounding approximation and strengthen it by the dual functions of only part of the leaf nodes of the branch-and-bound tree that correspond to strong dual polyhedral pieces. The challenge in such an extension is to how to combine these new polyhedral pieces with the LP value function such that the new function remains lower bounding to the MILP value function of the recourse problem.

Our computational experiments also show that solving the deterministic equivalent problems of the two-stage mixed integer test instances is faster than using the generalized Benders' algorithm to solve them. This is in contrast with the case of continuous two-stage optimization problems for which Benders' decomposition has been a very successful solution framework. This difference in performance is a direct consequence of the structure of the MILP value function which is tremendously more complicated than the one of an LP value function. In the continuous Benders' method, each individual dual function added to the master problem is simply a single valid inequality, just as in the cutting plane method. Managing the size of the master problem therefore requires the same cut pool management techniques used in the cutting plane or branch-and-cut algorithms. In the generalized Benders' algorithm, however, a collection of potentially many valid inequalities build a dual function, therefore, adding and dropping individual valid inequalities is not trivial. Managing cut pools in this case requires managing a collection of valid inequalities all together. Detangling these valid inequalities, which represent a piece-wise polyhedral function, to individual valid inequalities is not straightforward, as a single polyhedral function has to be a lower bounding function to the recourse value function which can be strong only at a very limited number of right-hand sides. However, reducing the number of intertwined valid inequalities in order to represent a simpler piece-wise polyhedral function by relaxation methods we explained earlier may result in faster solution times.

Bibliography

Ahmed, S., Tawarmalani, M., and Sahinidis, N. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377.

Avis, D. and Fukuda, K. (1992). A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*, 8(1):295–313.

Bachem, A. and Schrader, R. (1980). Minimal inequalities and subadditive duality. *SIAM Journal on Control and Optimization*, 18(4):437–443.

Bank, B., Guddat, J., Klatte, D., Kummer, B., and Tammer, K. (1983). *Non-linear parametric optimization*. Birkhäuser verlag.

Bard, J. F. (1998). *Practical bilevel optimization: algorithms and applications*, volume 30. Springer.

Bazaraa, M., Jarvis, J., Sherali, H., and Bazaraa, M. (1990). *Linear programming and network flows*, volume 2. Wiley Online Library.

Belotti, P. (2009). Couenne: a users manual. *Technical Report*.

Birge, J. and Louveaux, F. (1997). *Introduction to stochastic programming*. Springer Verlag.

Blair, C. (1995). A closed-form representation of mixed-integer program value functions. *Mathematical Programming*, 71(2):127–136.

BIBLIOGRAPHY

Blair, C. and Jeroslow, R. (1977). The value function of a mixed integer program: I. *Discrete Mathematics*, 19(2):121–138.

Blair, C. and Jeroslow, R. (1982). The value function of an integer program. *Mathematical Programming*, 23(1):237–273.

Blair, C. and Jeroslow, R. (1984). Constructive characterizations of the value-function of a mixed-integer program i. *Discrete Applied Mathematics*, 9(3):217–233.

Blair, C. E. (1978). Minimal inequalities for mixed integer programs. *Discrete Mathematics*, 24(2):147 – 151.

Blair, C. E. and Jeroslow, R. G. (1986). Computational complexity of some problems in parametric discrete programming. i. *Mathematics of Operations Research*, 11(2):pp. 241–260.

Carøe, C. and Schultz, R. (1998). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1):37–46.

Carøe, C. and Tind, J. (1997). A cutting-plane approach to mixed 0-1 stochastic integer programs. *European Journal of Operational Research*, 101(2):306–316.

Carøe, C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1):451–464.

Edmundson, H. (1957). Bounds on the expectation of a convex function of a random variable. Technical report, DTIC Document.

Gade, D., Küçükyavuz, S., and Sen, S. (2012). Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, pages 1–26.

Geoffrion, A. and Nauss, R. (1977). Parametric and post optimality analysis in integer linear programming. *Management Science*, 23(5):453–466.

Gomory, R. and Johnson, E. (1972a). Some continuous functions related to corner polyhedra, i. *Mathematical Programming*, 3(1):23–85.

Gomory, R. E. and Johnson, E. L. (1972b). Some continuous functions related to corner polyhedra, ii. *Mathematical Programming*, 3(1):359–389.

Güzelsoy, M. (2009). *Dual methods in mixed integer linear programming.* PhD thesis, Lehigh University.

Jeroslow, R. (1978). Cutting-plane theory: Algebraic methods. *Discrete mathematics*, 23(2):121–150.

Jeroslow, R. G. (1979). Minimal inequalities. *Mathematical Programming*, 17(1):1–15.

Johnson, E. L. (1973). Cyclic groups, cutting planes and shortest paths. *Mathematical programming*, pages 185–211.

Johnson, E. L. (1974). On the group problem for mixed integer programming. In *Approaches to Integer Programming*, pages 137–179. Springer.

Jorjani, S., Scott, C. H., and Woodruff, D. L. (1999). Selection of an optimal subset of sizes. *International journal of production research*, 37(16):3697–3710.

Kall, P. and Mayer, J. (2010). *Stochastic linear programming: models, theory, and computation.* Springer Verlag.

Klabjan, D. (2007). Subadditive approaches in integer programming. *European journal of operational research*, 183(2):525–545.

Klein, D. and Holm, S. (1979). Integer programming post-optimal analysis with cutting planes. *Management Science*, 25(1):64–72.

Kong, N., Schaefer, A., and Hunsaker, B. (2006). Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach. *Mathematical Programming*, 108(2):275–296.

Laatsch, R. G. et al. (1964). Extensions of subadditive functions. *Pacific J. Math*, 14:209–215.

Laporte, G. and Louveaux, F. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142.

Llewellyn, D. C. and Ryan, J. (1993). A primal dual integer programming algorithm. *Discrete Applied Mathematics*, 45(3):261–275.

Loukakis, E. and Muhlemann, A. (1984). Parameterisation algorithms for the integer linear programs in binary variables. *European Journal of Operational Research*, 17(1):104–115.

Madansky, A. (1959). Bounds on the expectation of a convex function of a multivariate random variable. *The Annals of Mathematical Statistics*, pages 743–746.

Mak, W.-K., Morton, D. P., and Wood, R. K. (1999). Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1):47–56.

Marsten, R. E. and Morin, T. L. (1977). Parametric integer programming: The right-hand-side case. *Annals of Discrete Mathematics*, 1:375–390.

Morton, D. P. (1996). An enhanced decomposition algorithm for multistage stochastic hydroelectric scheduling. *Annals of Operations Research*, 64(1):211–235.

Nauss, R. M. (1979). *Parametric integer programming*, volume 67. University of Missouri Press Columbia, Missouri.

Nemhauser, G. and Wolsey, L. (1988). *Integer and combinatorial optimization*, volume 18. Wiley New York.

Ntaimo, L. (2010). Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations research*, 58(1):229–243.

Ntaimo, L. and Sen, S. (2005). The million-variable march for stochastic combinatorial optimization. *Journal of Global Optimization*, 32(3):385–400.

Piper, C. J. and Zoltners, A. A. (1976). Some easy postoptimality analysis for zero-one programming. *Management Science*, 22(7):759–765.

Ralphs, T., Güzelsoy, M., and Mahajan, A. (2013). Symphony 5.5.0 users manual.

Ralphs, T. K. and Güzelsoy, M. (2005). The symphony callable library for mixed integer programming. In *The next wave in computing, optimization, and decision technologies*, pages 61–76. Springer.

Roodman, G. M. (1972). Postoptimality analysis in zero-one programming by implicit enumeration. *Naval Research Logistics Quarterly*, 19(3):435–447.

Roodman, G. M. (1974). Postoptimality analysis in integer programming by implicit enumeration: The mixed integer case. *Naval Research Logistics Quarterly*, 21(4):595–607.

Ruszczynski, A. and Shapiro, A. (2003). Stochastic programming models. *Handbooks in operations research and management science*, 10:1–64.

Ruszczyński, A. and Świtanowski, A. (1997). Accelerating the regularized decomposition method for two stage stochastic linear problems. *European Journal of Operational Research*, 101(2):328–342.

Schrage, L. and Wolsey, L. (1985). Sensitivity analysis for branch and bound integer programming. *Operations Research*, pages 1008–1023.

Schrijver, A. (1980). On cutting planes. *Annals of Discrete Mathematics*, 9:291–296.

Schultz, R., Stougie, L., and Van Der Vlerk, M. (1998). Solving stochastic programs with integer recourse by enumeration: A framework using Gröbner basis. *Mathematical Programming*, 83(1):229–252.

Sen, S. and Higle, J. (2005). The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20.

Sen, S. and Sherali, H. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223.

Shapiro, J. F. (1977). Sensitivity analysis in integer programming. *Annals of Discrete Mathematics*, 1:467–477.

Sherali, H. and Fraticelli, B. (2002). A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1):319–342.

Sherali, H. and Zhu, X. (2006). On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming*, 108(2):597–616.

Sherali, H. D. and Smith, J. C. (2009). Two-stage stochastic hierarchical multiple risk problems: models and algorithms. *Mathematical programming*, 120(2):403–427.

Trapp, A. C., Prokopyev, O. A., and Schaefer, A. J. (2013). On a level-set characterization of the value function of an integer program and its application to stochastic programming. *Operations Research*, 61(2):498–511.

Van Slyke, R. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, pages 638–663.

Wolsey, L. (1981a). Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming*, 20(1):173–195.

Wolsey, L. A. (1981b). The b-hull of an integer program. *Discrete Applied Mathematics*, 3(3):193–201.

Yuan, Y. and Sen, S. (2009). Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing*, 21(3):480–487.

Zhao, G. (2001). A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs. *Mathematical Programming*, 90(3):507–536.

Biography

Anahita Hassanzadeh was born in Mashhad, Iran in 1984. She received her undergraduate degree with honors in Industrial and Systems Engineering from Amirkabir University of Tehran (Tehran Polytechnic) in 2006. She continued her studies with a Masters in the same department in the same year and graduated in 2008 with highest honors. She started her Ph.D. in 2010 at the Department of Industrial and Systems Engineering of Lehigh University. She became a P. C. Rossin Doctoral Fellow in 2011. She interned at Oracle Inc. during summer 2012, working on mixed integer pricing optimization problems. She joined the Data Science team at The Climate Corporation in summer 2014 as a quantitative researcher with a focus on stochastic optimization. Her current focus is in the development of data-driven models and solution methods for stochastic yield optimization and spatio-temporal pattern recognition problems.